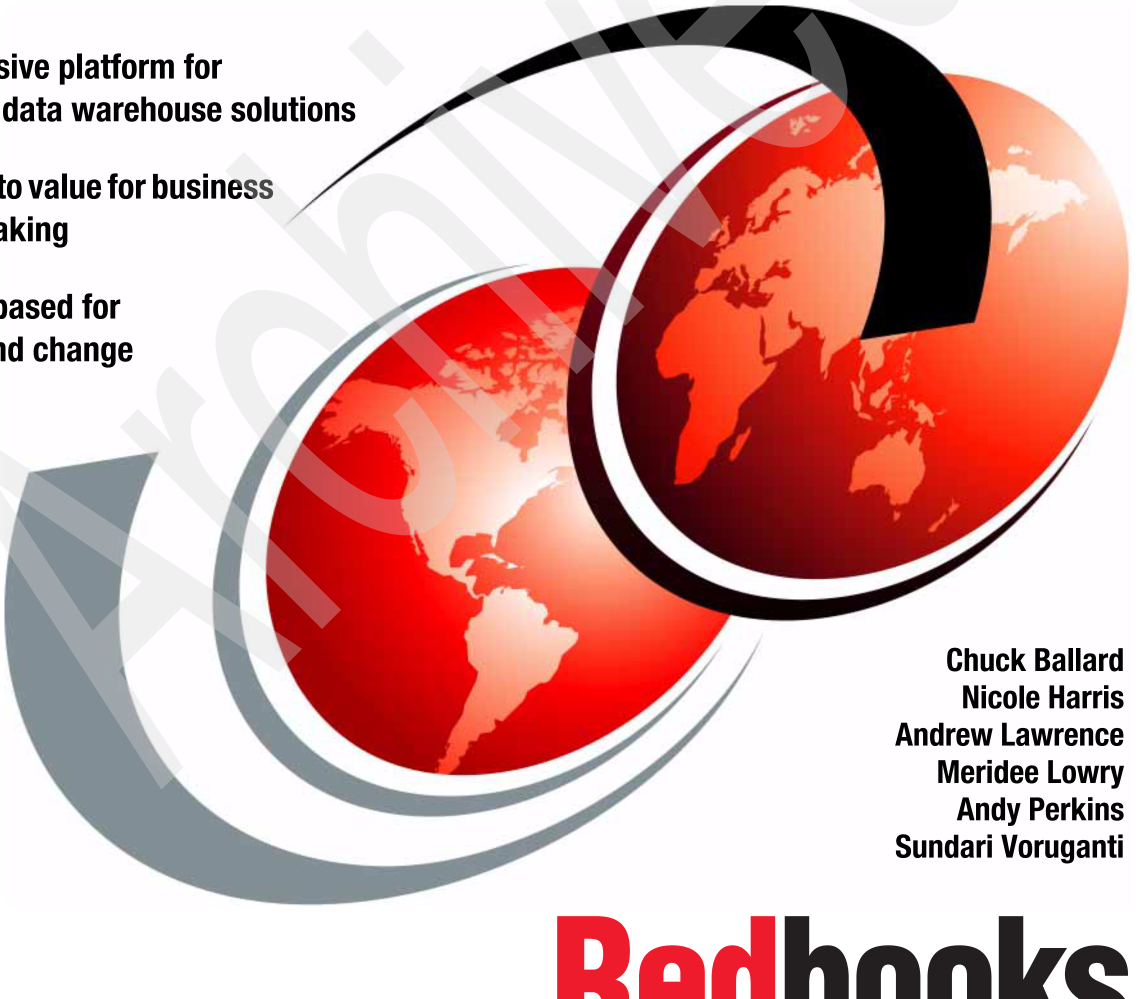
Information Management

IBM

InfoSphere Warehouse
A Robust Infrastructure for Business Intelligence

- Comprehensive platform for architected data warehouse solutions
- Faster time to value for business decision-making
- Standards-based for flexibility and change



Chuck Ballard
Nicole Harris
Andrew Lawrence
Meridee Lowry
Andy Perkins
Sundari Voruganti

Redbooks

ibm.com/redbooks

International Technical Support Organization

InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence

June 2010

SG24-7813-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

First Edition (June 2010)

This edition applies to Version 9.7 of IBM InfoSphere Warehouse (product number 5724-E34).

Contents

Notices

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the web at `http://www.ibm.com/legal/copytrade.shtml`

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
Alphablox®
Balanced Warehouse®
Blox®
ClearCase®
Cognos®
DataStage®
DB2®
Distributed Relational Database Architecture™
DRDA®
DS4000®
eServer™
HACMP™
HiperSockets™
IBM®
IMS™
Informix®
InfoSphere™
Intelligent Miner®
LanguageWare®
NUMA-Q®
Parallel Sysplex®
POWER6®
pSeries®
pureXML®
QMF™
Rational®
Red Brick®
Redbooks®
Redbooks (logo) ®
System z10®
System z9®
System z®
Tivoli®
WebSphere®
z/Architecture®
z/OS®
z10™
z9®
zSeries®

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cognos, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

ACS, Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication is primarily intended for use by IBM, IBM clients, and IBM business partners. In this book, we describe and demonstrate Version 9.7 of IBM InfoSphere™ Warehouse.

InfoSphere Warehouse is a comprehensive platform with all the functionality required for developing robust infrastructure for business intelligence solutions. InfoSphere Warehouse enables companies to access, analyze, and act on operational and historical information (whether structured or unstructured) to gain business insight for improved decision making. InfoSphere Warehouse solutions simplify the processes of developing and maintaining a data warehousing infrastructure and significantly enhance business value

The InfoSphere Warehouse platform provides a fully integrated environment built around IBM DB2® 9.7 server technology on Linux®, UNIX®, and Microsoft® Windows® platforms, as well as System z®. Common user interfaces support application development, data modeling, and mapping, SQL transformation, online application processing (OLAP) and data mining functionality from virtually all types of information.

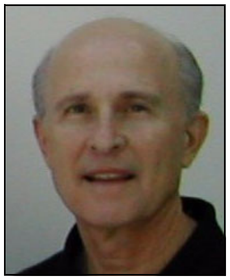
Composed of a component-based architecture, there are client and server functions, both making use of emerging IBM Software Group frameworks. It extends the DB2 data warehouse with design-side tooling and runtime infrastructure for OLAP, data mining, InLine Analytics, and intra-warehouse data movement and transformation, in a common platform.

Ultimately, companies can save money as a result of providing users with a more holistic view of the company, a consistent view of information across functions and an improved ability to analyze performance across divisions and activities. InfoSphere Warehouse includes monitoring and automation capabilities designed to help businesses proactively diagnose and resolve database issues to maintain optimum performance. In this manner, the platform helps companies apply business intelligence more broadly to improve operational business processes throughout the organization.

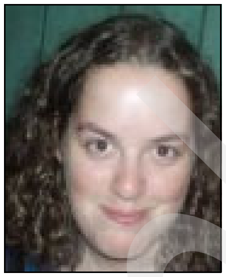
InfoSphere Warehouse is available as a stand-alone software solution or as part of a preconfigured, preintegrated, pretested InfoSphere Balanced Warehouse® solution. You can choose from the multiple versions available to satisfy the particular needs of your company.

The team who wrote this book

This Redbooks publication was produced by a team of business and technical specialists from around the world working with the International Technical Support Organization, in San Jose, California.



Chuck Ballard is a Project Manager at the International Technical Support organization, in San Jose, California. He has over 35 years experience, holding positions in the areas of Product Engineering, Sales, Marketing, Technical Support, and Management. His expertise is in the areas of database, data management, data warehousing, business intelligence, and process re-engineering. He has written extensively on these subjects, taught classes, and presented at conferences and seminars worldwide. Chuck has both a Bachelors degree and a Masters degree in Industrial Engineering from Purdue University.

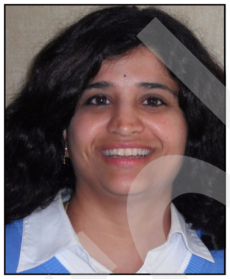


Nicole Harris is an InfoSphere Warehouse specialist for the Information Management SWG technical sales team in the UK and Ireland, with a particular focus on Data Mining. She holds a Bachelors in Zoology and a PhD in Plant Science from the University of Southampton, UK.

Andrew Lawrence is an Associate Partner with the IBM Business Analytics and Optimization consulting organization. He has over 20 years of business intelligence experience assisting clients with strategy, architecture, design, and implementation. Andy has worked on many large scale data warehouse projects with DB2 on UNIX and the mainframe, primarily for financial organizations. He is located in Phoenix, Arizona.

Meridee Lowry is a Consulting IT Specialist, Information Management Technical Sales Specialist in Software Sales, for IBM Sales & Distribution in Pittsburgh, PA.

Andy Perkins is a data warehousing specialist for the IBM Silicon Valley Lab Data Warehouse on the System z SWAT team, focusing on InfoSphere Warehouse on System z. He lives in Dallas, Texas, and has over 26 years of experience with customer data warehouse and BI projects, providing support in consulting, architecture design, data modeling, and implementation of data warehouses and analytical solutions. Over the years Andy has worked with the majority of the IBM Information Management products portfolio on platforms ranging from the PC to the System z.



Sundari Voruganti is an IT specialist working on the Data Warehouse on System z SWAT Team, and is located at the IBM Silicon Valley Laboratory in San Jose, California. She is currently focusing on InfoSphere Warehouse for System z, and provides presales support activities. Her areas of expertise include Information Integration, Replication and InfoSphere Warehouse on System z. Sundari has over 10 years experience as a Quality Assurance Engineer, and holds a Masters in Computer Science from the University of Alberta, in Edmonton, Canada.

Other Contributors:

A special thanks to:

Willie Favero for his written content, which contributed significantly to the development of this book. He is a seasoned veteran with IBM and an expert on System z and DB2 for z/OS®. His knowledge and experience on these systems has gained him a highly respected reputation, worldwide. Willie is a Senior Certified Consulting IT Software Specialist, and zChampion, located at the IBM Silicon Valley Lab in San Jose, CA.

Dan DeKimpe for his advice and guidance, and particularly for his technical review and feedback information. Dan is a software developer for Cubing

Services and Business Analytics, and is located at the IBM Silicon Valley Lab in San Jose, CA.

Thanks also to the following people for their contributions to this project:

From IBM Locations Worldwide
Jaime Anaya - Information Management Solutions Development for System z, Silicon Valley Lab, San Jose, CA.
Vikram S. Khatri - Certified Consulting I/T Specialist, SMPO DB2 Migration Team, Silicon Valley Lab, San Jose, CA.
Anuradha I Pariti - Software Engineer - Replication Development, Information Management, Silicon Valley Lab, San Jose, CA.
Warren Pettit - Information Management Instructor/Developer for Data Warehouse courses, Menlo Park, CA.
Patricia Shimer - Data Replication Software Test Specialist, Information Management, Silicon Valley Lab, San Jose, CA.
Burt L. Vialpando - Certified I/T Specialist, DB2 Migrations, Silicon Valley Lab, San Jose, CA.
Zach K Zakharian - Technical Support Specialist, Information Management Software Group, Silicon Valley Lab, San Jose, CA.

From the International Technical Support Organization, San Jose, CA
Mary Comianos - Publication Management
Ann Lund - Residency Administration
Emma Jacobs - Graphics
James Hoy- Editor

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Gaining business insight with InfoSphere Warehouse

In this chapter we discuss and describe the capabilities required for building a business intelligence (BI) architectural framework. Such a framework can enable an expanding number of business users to provide analytic services as part of their normal daily business processes. These services can be made easily accessible to users in a format with which they are familiar, and would require little to no additional training to take advantage of their use.

In particular, for IBM, this framework is provided by the IBM InfoSphere Warehouse offering. We describe that offering in detail in the remaining chapters of this IBM Redbooks publication.

Providing access to enterprise data and powerful analytics capabilities to more users, at all levels of the business can enable significant improvements in the efficiency and effectiveness of the business enterprise. It is these improvements that can enable business to lower costs, meet business performance measurements, and make a direct positive impact on profitability. With that impact the logical progression is to enable additional growth and gain market share. In a word, we might call that being successful.

1.1 Current business challenges

Over the last decade or so, there have been a few interesting trends in the business world. There was a time when the focus was on unrestricted growth of the business. This, of course, led to an extremely painful corrective period where expenses had to be brought in line and the business brought back to profitability. There seems now to be a movement where companies are returning to the fundamentals of sound business models.

There is also a renewed emphasis on growth and increasing the value of their business, but with a more balanced approach. That is, companies are looking for growth opportunities, but keeping a careful eye on costs. CEO's must have the ability to be responsive and flexible to meet changing business requirements. this is done by maximizing the efficiency of the entire operation in terms of people, processes, and resources.

Key to maximizing efficiency is the ability to make better informed business decisions more quickly, from the CEO to line-of-business employees. To obtain the required responsiveness and flexibility, decision making has to be pushed lower in the organization, and to a larger number of employees. Access to the right information at the right time by the right people is the challenge currently faced by the CIO.

Expanding decision making to the larger numbers and roles of employees requires a different approach for access to information than is currently used in most companies. In addition to working with the typical operational type of data found in the daily business processes, an expanded reach is needed that requires access to the type of data found in analytic tools more typically used by knowledge workers and data analysts. This is the future of BI, one in which all companies should be heading.

1.2 Information as a service

Companies are literally drowning in data that has been captured in various kinds of data repositories. The goal has been to capture data in a safe, reliable, and secure fashion. However, the struggle has always been to find a simple but powerful way to organize and filter the data to discover the business value. There are many technical and business reasons for the difficulty in organizing, accessing, and understanding company data. Consider the shear volume of data, structured and unstructured data, and data in particular silos kept for the exclusive use of a particular organization in the company. These can all be deterrents to providing the right data at the right time to the right people. There are solutions, and companies must employ them.

For example, there is great promise in the concept of delivering information as a service, called *Information On Demand*. This service requires that information be freed from the complexities of individual data stores or formats in a process called *virtualization*, which enables you to adapt business processes more rapidly while optimizing the underlying technology. You must also provide a unified view of the information, ensuring consistent formats and increased quality of data. This is accomplished through information integration. You must also accelerate the use of information by providing new, innovative ways to organize, maintain, and analyze it.

As seen in Figure 1-1, Information On Demand is all about delivering information to any tool, process, or person that can benefit from it. In this book we discuss how to deliver advanced, custom analytics as an embeddable service.

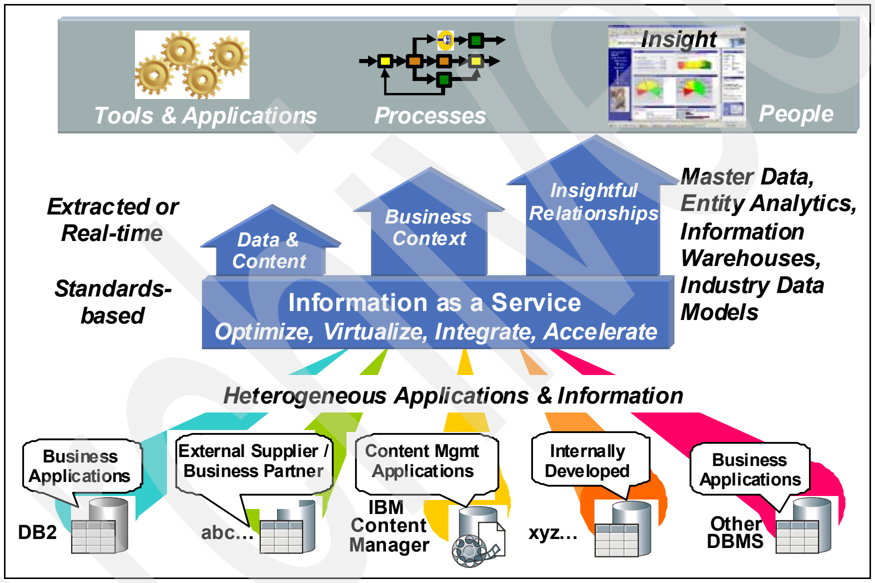


Figure 1-1 Information On Demand

1.3 Embedding analytics into business processes

The IT community is acquainted with the value proposition of the data warehouse as a vital foundation for any strategic approach to making use of information assets, by establishing consistent, predictable levels of data quality, breadth, and depth in a managed environment. However, the scope of the data warehouse has expanded beyond the traditional use as a reporting base. Advanced analytics in the form of Online Analytical Processing (OLAP) and data mining has greatly increased the value of the data warehouse to the corporation.

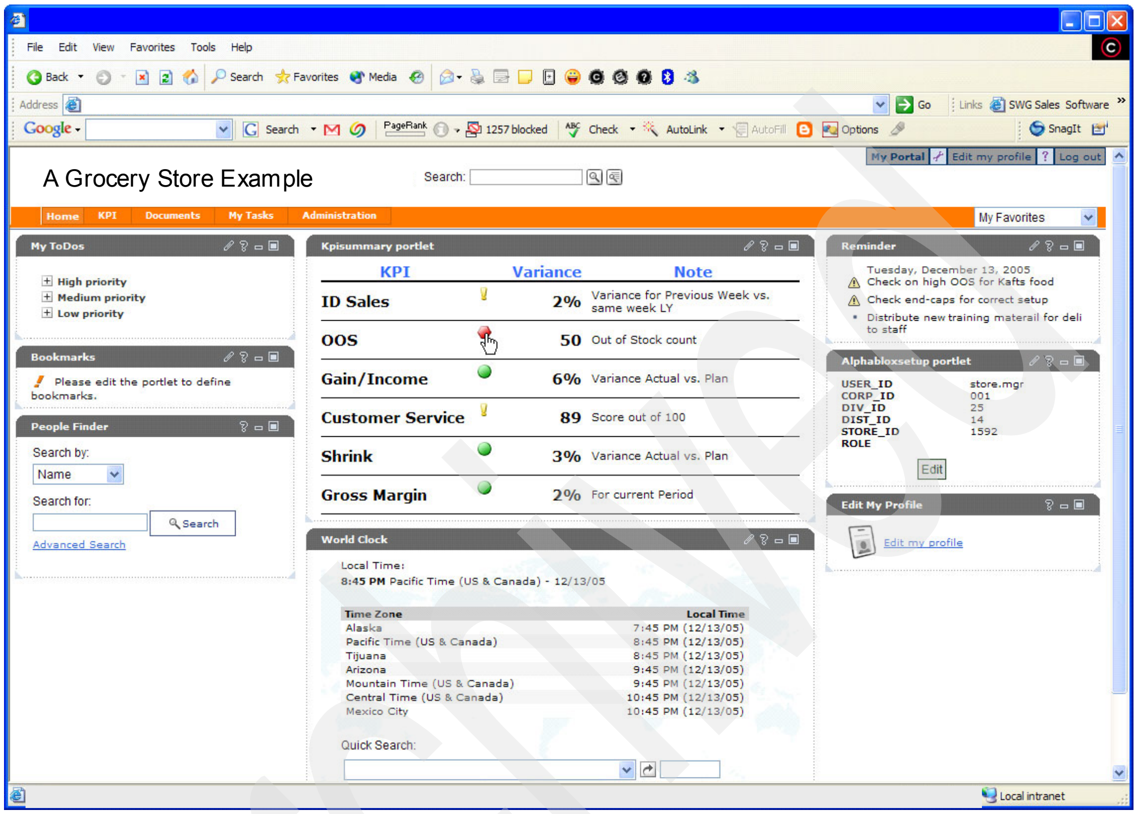
This advanced type of analytics has typically been performed by knowledge workers in the corporate office using specialized tools. This has provided value to the corporation, but as more decision making is pushed to the line-of-business workers, the requirement for delivering advanced analytical applications outside of the corporate office, and sometimes outside of the corporation, has become the competitive edge.

This means delivering advanced analytic capabilities to hundreds, or even thousands, of people who are not trained in the use of these OLAP and data mining tools. In addition to the expertise needed, these tools are expensive at a per seat cost and typically require a local client application to be installed. Paying for and managing this for hundreds or thousands of users, is prohibitive.

These new users are already familiar with using applications over the Web, typically through web browsers. Corporations have found delivering operational applications over the web to be effective and the line-of-business workers to be well versed in using web browsers. The goal is to deliver advanced analytic capabilities without the user knowing that they are performing advanced functions. They should not be cognizant that they are doing something different than they normally do in their job. When a worker logs into the corporate portal, there should just be objects on the window that represents things that need to be done, whether it be running payroll, ordering inventory, scheduling personnel, interacting with an OLAP database using interactive charts or even performing data mining through, as an example, a market basket analysis application.

This concept of integrating advanced analytics into every day business processes is commonly referred to as operational BI. It is critical to have the ability to embed analytic functionality into the daily routine of a line-of-business employee with a minimum amount of effort and interruption.

Figure 1-2 depicts a Web-based workbench example scenario for a grocery store manager.



A Grocery Store Example

Search:

Home | KPI | Documents | My Tasks | Administration | My Favorites

My ToDos

- High priority
- Medium priority
- Low priority

Bookmarks

Please edit the portlet to define bookmarks.

People Finder

Search by: Name

Search for:

Search

Advanced Search

Kpisummary portlet

KPI		Variance	Note
ID Sales	!	2%	Variance for Previous Week vs. same week LY
OOS	●	50	Out of Stock count
Gain/Income	●	6%	Variance Actual vs. Plan
Customer Service	!	89	Score out of 100
Shrink	●	3%	Variance Actual vs. Plan
Gross Margin	●	2%	For current Period

World Clock

Local Time:
8:45 PM Pacific Time (US & Canada) - 12/13/05

Time Zone	Local Time
Alaska	7:45 PM (12/13/05)
Pacific Time (US & Canada)	8:45 PM (12/13/05)
Tijuana	8:45 PM (12/13/05)
Arizona	9:45 PM (12/13/05)
Mountain Time (US & Canada)	9:45 PM (12/13/05)
Central Time (US & Canada)	10:45 PM (12/13/05)
Mexico City	10:45 PM (12/13/05)

Quick Search:

Reminder

Tuesday, December 13, 2005
- Check on high OOS for Kafts food
- Check end-caps for correct setup
- Distribute new training materail for deli to staff

Alphabloxsetup portlet

USER_ID	store.mgr
CORP_ID	001
DIV_ID	25
DIST_ID	14
STORE_ID	1592
ROLE	

Edit

Edit My Profile

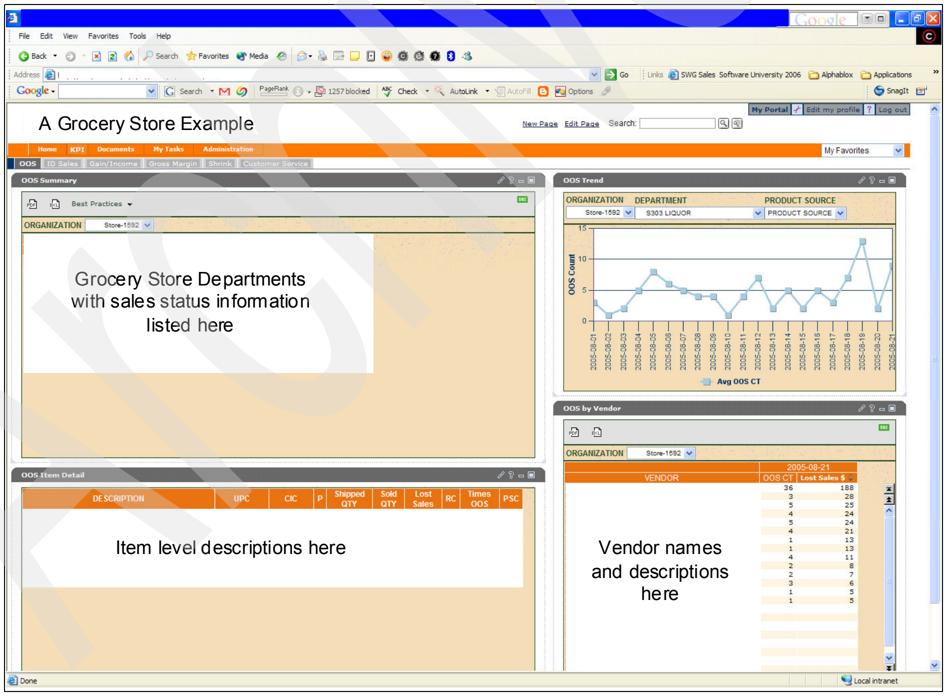
Edit my profile

Figure 1-2 Grocery store manager workbench embedded analytics example

Figure 1-3 also depicts a window that could be part of a Web-based workbench example scenario for a grocery store manager. However, both of these figures could be examples of applications that could be accessed and viewed from a common workbench for the store manager. Such a workbench could give the store manager access to all of the applications that are available and used, but in one integrated workbench.

Functionality can be delivered by traditional applications and by analytical applications, but the store manager need not care, as they are seamlessly integrated and embedded into the workbench.

For example, the store manager can use the workbench to analyze out of stock (OOS) situations that were highlighted by the key performance indicator (KPI) component on the home page. By clicking the OOS alert, the store manager is directed through a series of analytics to determine where the problem lies and how best to handle it, which might result in a stock reorder being generated. All of this can be done without the store manager noticing the difference between analytics and operational systems. He is interacting with functional objects on his workbench.



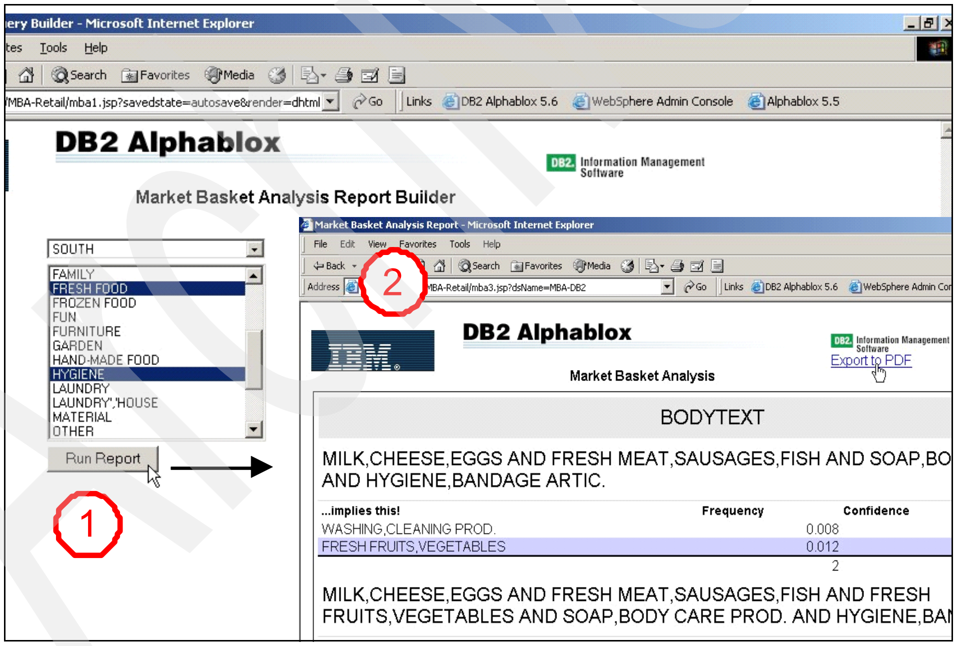
VENDOR	2005-08-21 OOS CT	Lost Sales $
	36	188
	3	28
	5	25
	4	24
	5	24
	4	21
	1	13
	1	13
	4	11
	2	8
	2	7
	3	6
	1	5
	1	5

Figure 1-3 Grocery store manager workbench embedded analytics example

Embedding analytics into a company intranet is not limited to OLAP analytics. For example, store managers need to understand what products their customers are buying in a specific trip to the store by analyzing what is in the basket. This can be accomplished with an associations analysis provided by data mining tools. These tools have traditionally been developed by statisticians at corporate headquarters.

However, in the business environment today, store managers also need to be much more responsive. To do so, they need a capability such as a Web-based application on their workbench with which to perform, for example, their own market basket analysis. Figure 1-4 shows an example of such an analysis application. It shows the report builder (1), and the example results of a market basket analysis (2). The store manager now has the type of information needed to manage and control the business better. All this is done without the store manager even realizing that what is being performed is really advanced analysis using data mining.

The store manager has all this information available as the result of sophisticated data mining, but it appears as a simple analytical analysis on the workbench.


ery Builder - Microsoft Internet Explorer
tes Tools Help
Search Favorites Media
/MBA-Retail/mba1.jsp?savedstate=autosave&render=dhtml
Go Links DB2 Alphablox 5.6 WebSphere Admin Console Alphablox 5.5
DB2 Alphablox
DB2 Information Management Software
Market Basket Analysis Report Builder
SOUTH
FAMILY
FRESH FOOD
FROZEN FOOD
FUN
FURNITURE
GARDEN
HAND-MADE FOOD
HYGIENE
LAUNDRY
LAUNDRY','HOUSE
MATERIAL
OTHER
Run Report
1
Market Basket Analysis Report - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address MBA-Retail/mba3.jsp?dsName=MBA-DB2
Go Links DB2 Alphablox 5.6 WebSphere Admin Co
2
IBM
DB2 Alphablox
DB2 Information Management Software
Export to PDF
Market Basket Analysis

BODYTEXT

MILK,CHEESE,EGGS AND FRESH MEAT,SAUSAGES,FISH AND SOAP,BO
AND HYGIENE,BANDAGE ARTIC.

...implies this!	Frequency	Confidence
WASHING,CLEANING PROD.		0.008
FRESH FRUITS,VEGETABLES		0.012
		2

MILK,CHEESE,EGGS AND FRESH MEAT,SAUSAGES,FISH AND FRESH FRUITS,VEGETABLES AND SOAP,BODY CARE PROD. AND HYGIENE,BAI

Figure 1-4 Embedded data mining application example

Work also continues to grow in the area of access to, and use of, unstructured data. Unstructured data represents the largest, most current, and fastest growing source of information that is available today. This information exists in many different sources (such as call center records, repair reports, product reviews, emails, and many others). The text analysis features of InfoSphere Warehouse can help uncover the hidden value in this unstructured data. In this chapter we discuss the general architecture and business opportunities of analyzing unstructured data with the text analysis capabilities of InfoSphere Warehouse.

With InfoSphere Warehouse, users can create business insight from unstructured information. Information can be extracted from text columns in the data warehouse (for example call center notes, free-text survey-fields, repair reports, patient records). This extracted information is then used in reports, multidimensional analysis, or as input for data mining.

Unstructured data must be analyzed to interpret, detect, and locate concepts of interest that are not explicitly tagged or annotated in the original document. For example, documents can include domain-specific information, such as named entities (such things as persons, organizations, facilities, and products).

Text data is usually grouped in a category (such as unstructured data) together with video tapes, images, audio tapes, and other data that is stored into different fields with little or no subdivisions. The text analysis functions of InfoSphere Warehouse focus on information extraction to create structured data that can be analyzed with InfoSphere Warehouse tools. You can apply text analysis or a combined analysis of structured and unstructured data in all industries and across many business functions.

The text analysis capability in InfoSphere Warehouse can be integrated with IBM Cognos® reporting, enabling business users to exploit the text analysis results. This is an exciting and ever-expanding area that opens huge amounts of additional data for analysis and provides access to information not previously accessible.

1.4 From data warehouse to information warehouse

Consider how to deliver information as a service from the perspective of the enterprise data warehouse. Information as a service extends the notion of the traditional data warehouse by providing layers that apply business context to the data, as well as better insights into that data. The result is the higher value notion of the information warehouse, which enables more efficient and effective decision making, and faster action to be taken to avoid problems that could impact business performance. This is depicted in Figure 1-5 on page 9.

The base of the information warehouse is an enterprise data warehouse that provides a single version, or view, of the business. It is implemented as a single, scalable, and consolidated data warehouse.

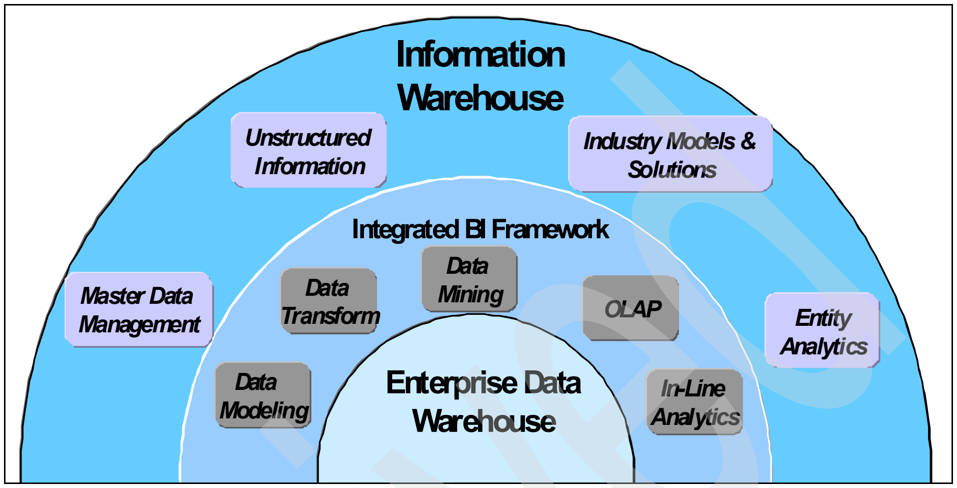


Figure 1-5 From data warehouse to information warehouse

With the foundation of the consolidated enterprise data warehouse in place, the next step is to begin enhancing the value of the data warehouse by incorporating capabilities (Such as intra-warehouse data movement and transformation, data mining, OLAP modeling and embedded analytics) once relegated to independent data marts. These critical horizontal capabilities typically have been developed alongside the data warehouse, but often in different areas of the larger organization. Their lack of integration with the data warehouse impacted their value and increased the overall cost of ownership.

Data mining is a great example of this. Data mining has tremendous potential for increasing the value of the warehouse. But it is a complex technology that has been loosely integrated with the data warehouse, and not accessible by administrators and users. Consequently, it has not been widely accepted and has not realized the real potential of data mining.

The integrated BI framework is about integrating capabilities (such as mining, providing support to improve the quality of data it is processing and support to make better and easier use of the results). In addition, the framework provides integrated tooling that makes it easier to incorporate this kind of function into the larger data warehouse/application environment and makes it more consumable.

The final layer of the information warehouse takes us from the traditional data warehouse to provide a higher level of value. This is achieved by incorporating richer sources of information (such as unstructured data), applying additional business context to data (such as with master data management), extracting additional insight from data (Entity Analytics) and tying it together in consumable solutions that solve specific business problems.

1.5 An integrated BI framework

To deliver analytic functionality in a service oriented environment requires building BI functionality into the data warehouse and making it accessible through open, standard interfaces as embeddable components in an integrated BI platform. To respond to the needs of the business requires a BI framework, or architecture, in place with an integrated data warehouse.

Components of a BI framework

A BI framework should consist of components and processes that allow you to respond to the analysis needs of the company in a timely fashion. If it takes too long to provide a new analytic function, the window of opportunity might close.

Such a framework must provide for the development and delivery of advanced analytic applications containing OLAP and data mining capabilities and a standard Web-based delivery mechanism. To accomplish this, base infrastructure capabilities are needed, such as:

- Moving and transforming data in the data warehouse for populating the data structures needed for OLAP and data mining
- Modeling and implementing OLAP, or cube, structures
- Creating data mining models and applying, or scoring, the models without having to move data out of, and back into, the data warehouse
- Delivering OLAP and data mining applications through the web in an embedded and customized approach so that these analytical applications seamlessly integrate into the existing Web-based application infrastructure

1.6 The DB2 Business Intelligence Solution Framework

The DB2 Business Intelligence Solution Framework enables you to extend the value of your enterprise data warehouse, and is based on the InfoSphere Warehouse architecture, depicted in Figure 1-6. The purpose of this framework is to provide the components necessary to build and support BI solutions. As examples, there are components for transforming and moving data into the data warehouse, modeling the data, creating OLAP models, creating and scoring data mining models in the database engine, and for developing and delivering embedded analytic application components into the existing corporate web infrastructure.

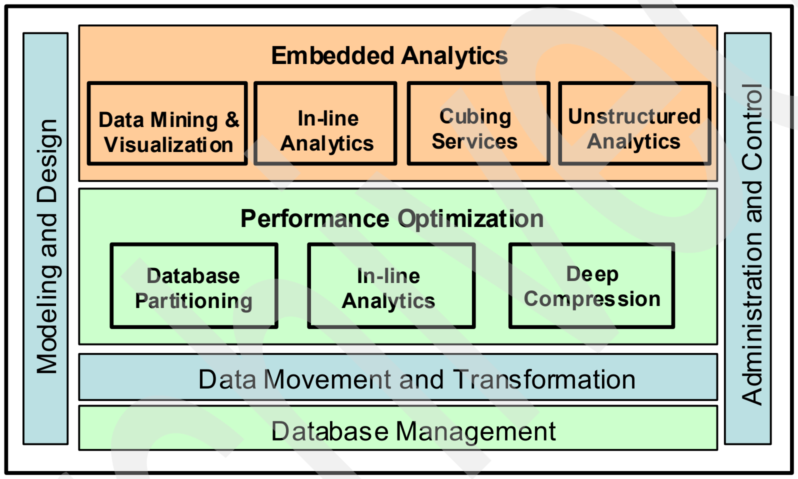


Figure 1-6 InfoSphere Warehouse architecture

InfoSphere Warehouse provides the components for building and delivering embedded Web-based analytic solutions. At the core is the DB2 Enterprise Server Edition (ESE) relational database engine, which provides a robust, scalable platform for the data warehouse on which the BI framework is built.

The integrated design studio provides the core components to perform the following tasks:

- Model data structures graphically
- Move and transform data in the data warehouse
- Implement OLAP
- Build and score data mining models
- Develop embedded analytic application components that expose the BI services available for embedding into the company Web-based infrastructure

The data modeling component enables the creation of physical data models and provides the base metadata to the other components. Physical data models might be developed, or reverse-engineered from existing databases. This is a subset of functionality from the IBM Rational® Data Architect data modeling product.

The data transformation component, called SQL Warehousing (SQW), provides the ability to develop SQL-based data movement and transformation flows in the data warehouse. It makes use of the SQL functionality of the DB2 relational database engine. Data flows and control flows are developed using graphical editors in Design Studio and makes use of the physical data model metadata.

The data mining component is an extension to the SQW component. It adds the ability to develop data mining models graphically, using mining flows to do data preparation, model creation, model visualization, model extraction, and model scoring. Model scoring can also be embedded into data flows to allow batch scoring in the data warehouse. There are also SQL and Java™ bean APIs for embedding data mining into applications.

To support slice and dice analytics, the OLAP component enables modeling and optimization of OLAP, or cube, structures. The modeling of OLAP structures, facts, dimensions, and hierarchies, for example, is performed as an extension to the physical data model and is typically used with star-schema-like structures. The OLAP component also provides wizards to optimize OLAP processing.

The in-line analytics component allows the development of visual analytic components that are embedded into, and delivered by, Web-based applications. For example, where Alphablox® is used, these components would be known as blox. These components can access and visualize data from OLAP structures, data mining, and relational tables.

Runtime environments are administered using a Web-based administration console. The administration console allows the deployment of InfoSphere Warehouse applications into the runtime environment, allows the scheduled execution of data movement flows, and administers OLAP and data mining functions. The Web-based analytic applications can also be deployed through the provided J2EE compliant WebSphere® Application Server.

This chapter has been a brief and high-level overview of the BI framework, along with the products and components to enable implementation. Additional details and examples in the remainder of this book help get you started developing and implementing your BI solutions environment.

2

Technical overview of InfoSphere Warehouse

In this chapter we provide an overview of InfoSphere Warehouse architecture, and its components. The remainder of this IBM Redbooks publication provides a detailed look at each of the various components.

Note: All the functions, features, and operators for InfoSphere Warehouse on the LUW platforms are not the same as on the System z platform. We try to point out the differences where appropriate in each chapter.

InfoSphere Warehouse represents the IBM offering for implementing integrated business intelligence (BI) solutions. The BI framework enables the transformation of the data warehouse from a static repository, primarily used for batch reporting, into an active end-to-end platform for BI solutions. InfoSphere Warehouse integrates design-time tooling and runtime infrastructure for Online Analytical Processing (OLAP) and data mining, and the delivery of embedded Inline analytics on a common platform based on the IBM DB2 Relational Database Server and the WebSphere Application Server.

InfoSphere Warehouse removes cost and complexity barriers, and enables delivery of powerful analytics to an enterprise. The integrated tooling enables a low total cost of ownership (TCO) and provides improved time-to-value for developing and delivering analytics enterprise-wide. InfoSphere Warehouse also avoids the requirement for multiple purchase justifications for BI tools and enables you to have the tools available when you need them.

2.1 InfoSphere Warehouse architecture

InfoSphere Warehouse is a suite of components that combines the strength of the DB2 database engine with a powerful business analytics infrastructure from IBM. InfoSphere Warehouse provides a comprehensive business analytics platform with the tools required to enable you to design, develop, and deploy analytical applications in your enterprise.

InfoSphere Warehouse can be used to build a complete data warehousing solution that includes a highly-scalable relational database engine, data access capabilities, business analytics, and user analysis tools. It integrates core components for data warehouse administration, data mining, OLAP, inline analytics, reporting, and workload management.

2.1.1 The component groups

InfoSphere Warehouse has a component-based architecture, complete with client and server components. Those components are arranged into three logical component groups, as shown in Figure 2-1 on page 15. These component groups are typically installed on three different computers, but can be installed on one or two, because they can operate in a multi-tier configuration. The component groups are as follows:

- Data server components

  This includes DB2, Workload Manager, Cubing Services, and data mining.
- Application server components

  This category includes WebSphere Application Server, InfoSphere Warehouse Administration Console, SQL Warehousing Tool, and IBM Alphablox and Miningblox.
- Client components

  The client components are Design Studio, IBM Data Server Client, Intelligent Miner® Visualizer, and Workload Manager Center.

In addition to these component products, the InfoSphere Warehouse documentation and tutorials can also be installed in any of the logical component groups.

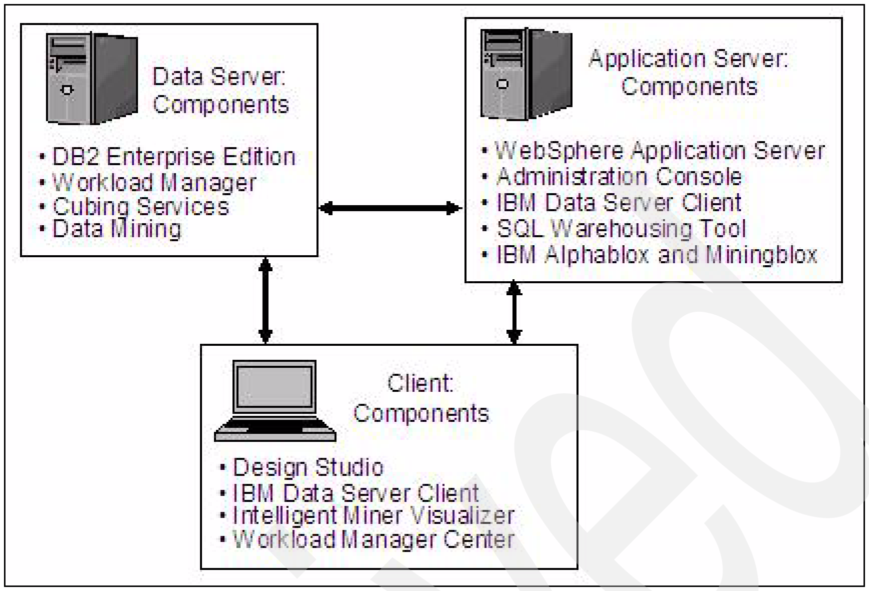


Figure 2-1 Logical component groups in the IBM InfoSphere Warehouse

The highly scalable and robust DB2 database servers are the foundation of the InfoSphere Warehouse. All the design and development is performed using Design Studio, an Eclipse-based development environment. The IBM WebSphere Application Server is used as the run-time environment for deployment and management of all InfoSphere Warehouse-based applications.

2.1.2 InfoSphere Warehouse components

The IBM InfoSphere Warehouse components are organized into the following major categories of the architecture:

- Database management system
- Data movement and transformation
- Performance optimization
- Embedded analytics
- Modeling and design
- Administration and control

The categories and components of the architecture are shown in Figure 2-2, and described in the remainder of this section.

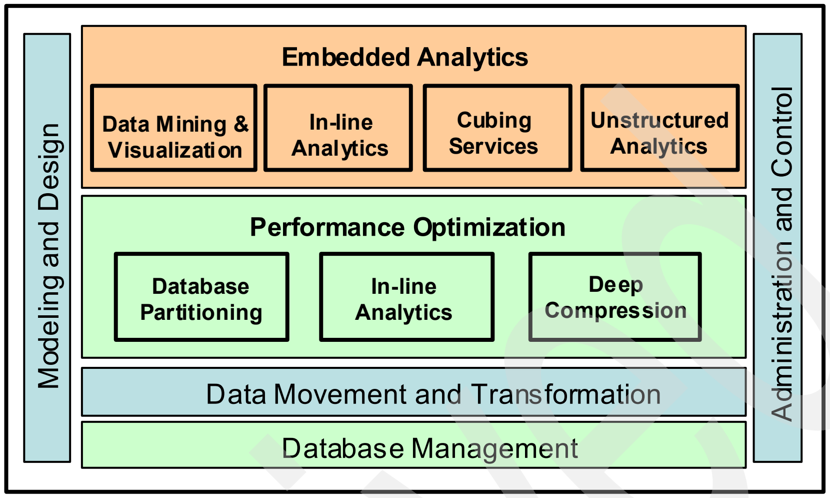


Figure 2-2 DB2 InfoSphere Warehouse functional component architecture

Database management system

The database management system is the foundation for the DB2 InfoSphere Warehouse. DB2 offers industry leading performance, scale, and reliability on your choice of platform from Linux to z/OS. It is optimized to deliver industry-leading performance across multiple workloads, while lowering administration, storage, development, and server costs.

DB2 has been available from IBM for quite a number of years, and has proven to be a highly functioning and highly reliable database management system. It is a leading database management system offering and is in use world-wide.

For more information about DB2, see the following web page:

http://www-01.ibm.com/software/data/db2/

Data movement and transformation

One of the key operating principles of the IBM InfoSphere Warehouse is to maintain simplicity. More importantly, the principle is to have integrated tooling to design, develop, and deploy data warehouse applications.

In the information environment, we typically find that organizations have several DBAs that spend a significant amount of time writing SQL scripts to maintain data in the data warehouse and data marts. They must create documentation for those SQL scripts, and keep it up to date and maintained over time. This is not only time consuming for them, but also for new staff to perform data warehouse maintenance. With InfoSphere Warehouse, the data movement and transformation for intra-warehouse data movement is accomplished using the SQL Warehousing tool (SQW).

SQW tooling is part of the Eclipse-based Design Studio. SQW enables DBAs to define data transformation and data movement tasks using the simple, intuitive, drag-and-drop interface. The SQW tool generates DB2-specific SQL based on visual operator flows that you model in the InfoSphere Warehouse Design Studio. The library of SQL operators covers the in-database data operations that are typically needed to move data between DB2 tables and to populate analytical structures (such as data mining models and multidimensional cubes). SQW complements and works well with ETL products from IBM and third parties that have specific integration points for IBM InfoSphere DataStage®.

There are two basic types of flows in SQW:

- Data flow

  A data flow moves and transforms data using SQL based operators to accomplish tasks (such as joins, filtering, splitting data, transforming data with column functions, and maintaining slowly changing dimensions).

- Control flow

  A control flow arranges data flows and other types of data processing into logical sequences of processing. A control flow can invoke OS scripts, SQL scripts, and ftp files, in addition to data flows.

Control flows are created using SQW are deployed and managed using the InfoSphere Warehouse Administration Console. The administration console allows for variables to be introduced in transformation jobs that can be modified at deployment or run time.

Performance optimization

The InfoSphere Warehouse has several features that can be used to enhance the performance of analytical applications, using data from the data warehouse.

More information about these features can be found in the following locations:

- For DB2 Linux, UNIX, and Windows

  Appendix A, “DB2 ESE for LUW: Support for data warehousing” on page 517

- For System z

  Appendix B, “DB2 for z/OS: Support for data warehousing” on page 571

Embedded analytics

IBM InfoSphere Warehouse has embedded analytics, in the form of OLAP and data mining, as part of the data warehouse infrastructure.

The InfoSphere Warehouse embedded analytics includes the following components:

- Data mining and visualization

  The IBM Intelligent Miner is embedded in the InfoSphere Warehouse. This allows for mining models to be deployed to the enterprise data warehouse and scoring to be performed against these mining models from in the data warehouse.

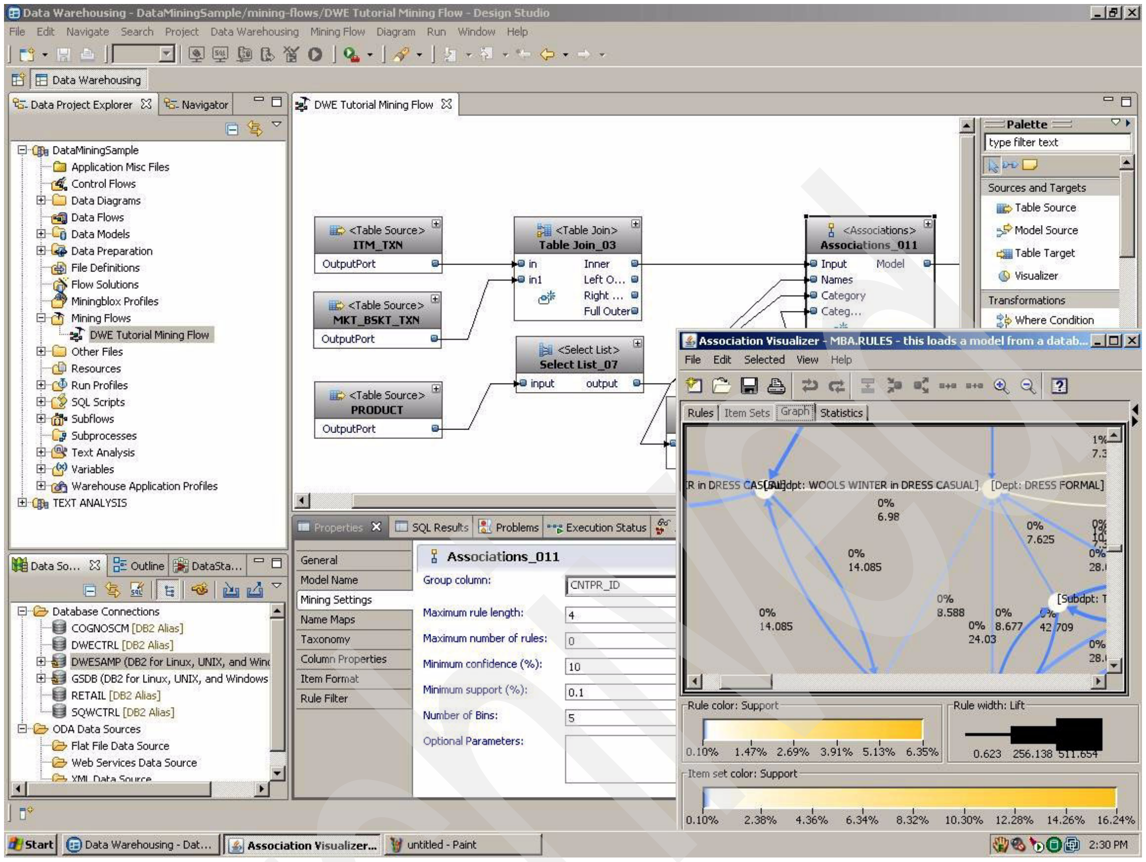
  Instead of taking the data to the mining tool, InfoSphere Warehouse brings the mining tool to the data. The creation of data mining models and the application of mining models happens in the DB2 database acting directly upon the data without removing it from the data warehouse.

  The data mining component of the InfoSphere Warehouse Design Studio (Design Studio) has functions to explore and visualize the data to increase the understanding of the data prior to mining. The graphical mining flow editor has operators for data preparation and for execution of the data mining algorithms. The graphical editor also helps to visualize the results of the mining and to extract information from the model for storing in a table for use by an application.

  Once a model is built to satisfaction, it can be applied to new data for the purpose of scoring or prediction. This can be integrated into a mining flow or a data flow for batch scoring directly in the database. Mining flows can also be integrated into SQW control flows.

  Modeling and scoring functions can be called through standard SQL functions, which enable data mining modeling and scoring functions to be embedded into applications for real-time data mining. For example, a market basket analysis application can easily be developed such that a store manager, just by making a few product selections, can see the results of data mining.

  Figure 2-3 on page 19 shows a mining flow open in the graphical mining flow editor, with data exploration visualizations and the data mining model visualizer. Data mining flows are also part of a data warehouse project along with data flows and mining flows. The figure must be magnified for readability. However, this is an example, so it is not necessary to read or understand that data. The purpose of the figure is to familiarize you with the format and structure of how the data types might appear when provided, along with their relationships.



Associations_011

Group column: CNTPR_ID

Maximum rule length: 4

Maximum number of rules: 0

Minimum confidence (%): 10

Minimum support (%): 0.1

Number of Bins: 5

Optional Parameters:

Figure 2-3 InfoSphere Warehouse data mining

- Inline analytics using InfoSphere Warehouse Alphablox

  InfoSphere Warehouse Alphablox (Alphablox) provides the ability to create custom, Web-based analytic applications rapidly. These applications can fit into the corporate infrastructure. They have the ability to reach users both inside and outside of the corporation. Applications built with Alphablox run in standard web browsers, enabling the execution of real-time, highly customizable multidimensional analysis in a web browser.

  InfoSphere Warehouse Alphablox accomplishes this by providing a set of pre-built components, or Blox®, that can be incorporated through a tag language into Java Server web pages or portlets. These blox are essentially calls to an application hosted by a J2EE-compliant application server. Figure 2-4 on page 20 shows a set of generic application Blox that help enable the creation of custom analytic solutions.

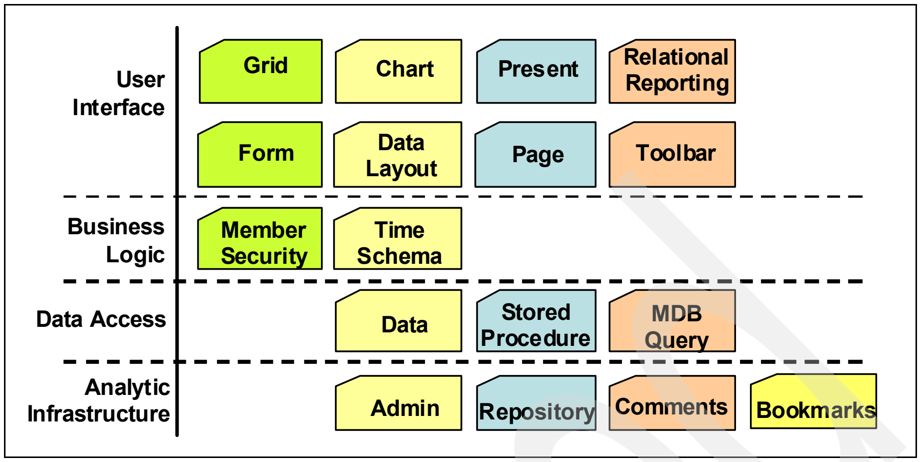


Figure 2-4 Alphablox application building blox

There are two components of the InfoSphere Warehouse Alphablox tooling. The Alphablox Cubing Engine (or cube server) uses the metadata created by the InfoSphere Warehouse Cubing Services (IWCS) component, and stored in the database, to instantiate a cube that can be queried by the embedded blox. Blox tags are created to make the call to the Alphablox Cubing Engine to request data. Alphablox blox can be embedded into web pages, or portlets, using a Java Server Page, or Portal development tool such as the Rational Application Developer. Alphablox components use the OLAP definitions to instantiate the cube and the Alphablox Cubing Engine translates MDX queries from the blox into SQL queries for accessing the data.

- InfoSphere Warehouse Cubing Services

  Cubing Services uses an MDX interface to access data in the data warehouse. This allows you to retain data in the warehouse and dynamically perform aggregations, enabling near real-time OLAP analysis. In addition to using the performance optimization features in the DB2 database engine, Cubing Services has another performance optimization layer, in the form of in-memory data caching.

  A star schema, or dimensional model, is typically used to store data for analytical needs. There is a fact table that contains all of the measurable information and a set of dimension tables that specify the various ways in which the facts can be viewed, including various hierarchical relationships in a dimension. However, to the relational database engine, these are just a set of tables in the database with no special meaning.

OLAP tools are usually built around the knowledge and relationships that are special in a dimensional model. Because the relational database itself has no special knowledge of the relationships inherent in the dimensional model, OLAP tools have to provide mechanisms in the tool to supply the additional meaning.

The InfoSphere Warehouse OLAP tooling extends the relational database model to include the dimensional model relationships. This metadata is stored once in the relational database and can be used by other IBM partner BI tools.

The InfoSphere Warehouse Design Studio OLAP component extends the physical data model to include the dimensional model and is stored in cube definitions. Cube definitions include information about the facts, the dimensions and the hierarchies in the dimensional model. This dimensional, or OLAP, metadata is available for use by BI tools and is used by InfoSphere Warehouse Alphablox. Because the shared common metadata includes aggregation formulas and calculations, you benefit from greater consistency of analytical results across the enterprise.

Once a BI tool has the dimensional knowledge about the underlying star schema structure, it can use that information to form SQL queries to retrieve the data. Depending on the granularity of the data and the granularity of the SQL query the SQL might cause quite a bit of work to be done by DB2 to join, sort, and aggregate data to the requested level. To optimize the performance of OLAP type queries, the InfoSphere Warehouse Cubing Services component also provides an optimization wizard to make model-based recommendations to build performance structures called Materialized Query Tables (MQTs). These tables pre-aggregate the most commonly requested data slices to be used by the DB2 optimizer and transparently reroute queries to the pre-aggregated data, which greatly accelerates access to the data.

Figure 2-5 shows a star schema data model, as viewed in InfoSphere Warehouse Design Studio. It depicts the cube metadata extension to the physical data model and the OLAP cube as deployed to a physical database. The purpose of Figure 2-5 is to give you an understanding of the type of data presented and its placement.

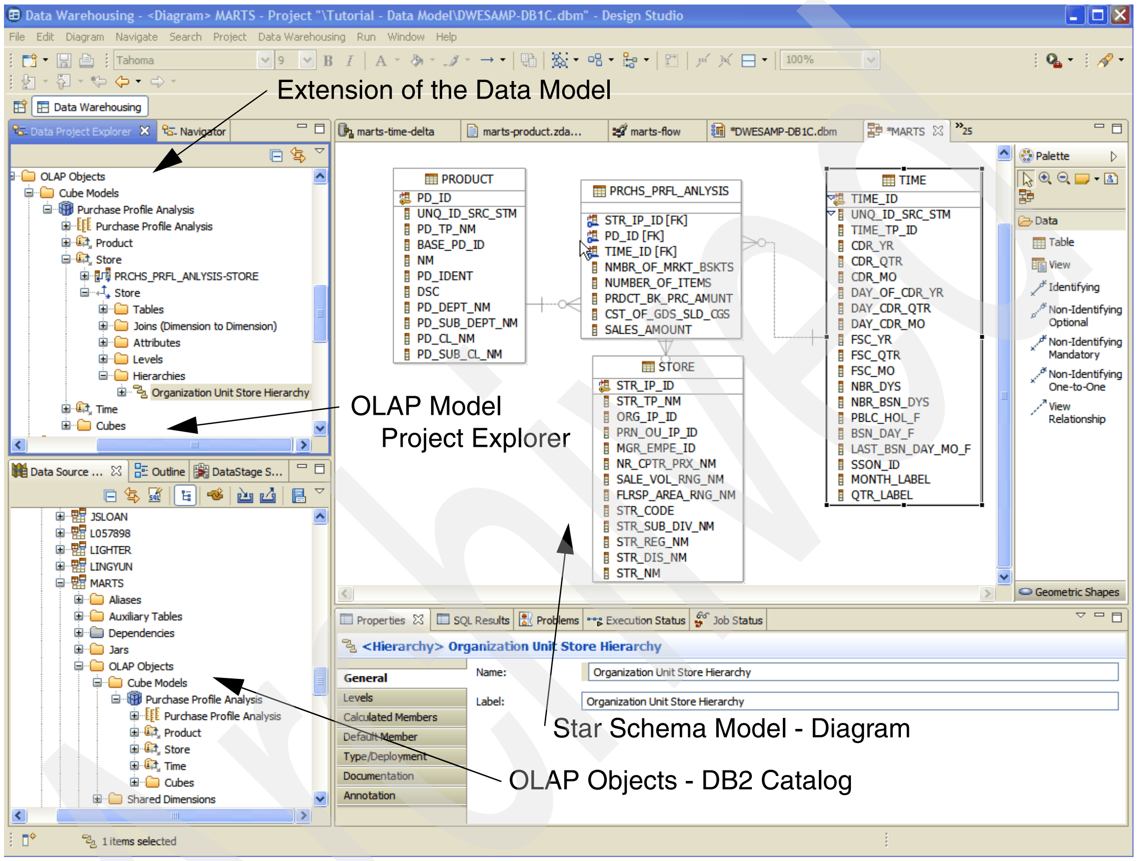


Figure 2-5 InfoSphere Warehouse Cubing Services

- Unstructured text analytics

  Practically every organization has data stored in both structured and unstructured format. The unstructured text analytics component enables you to analyze unstructured text data and to use it alongside structured data to gain better business insight.

Modeling and design

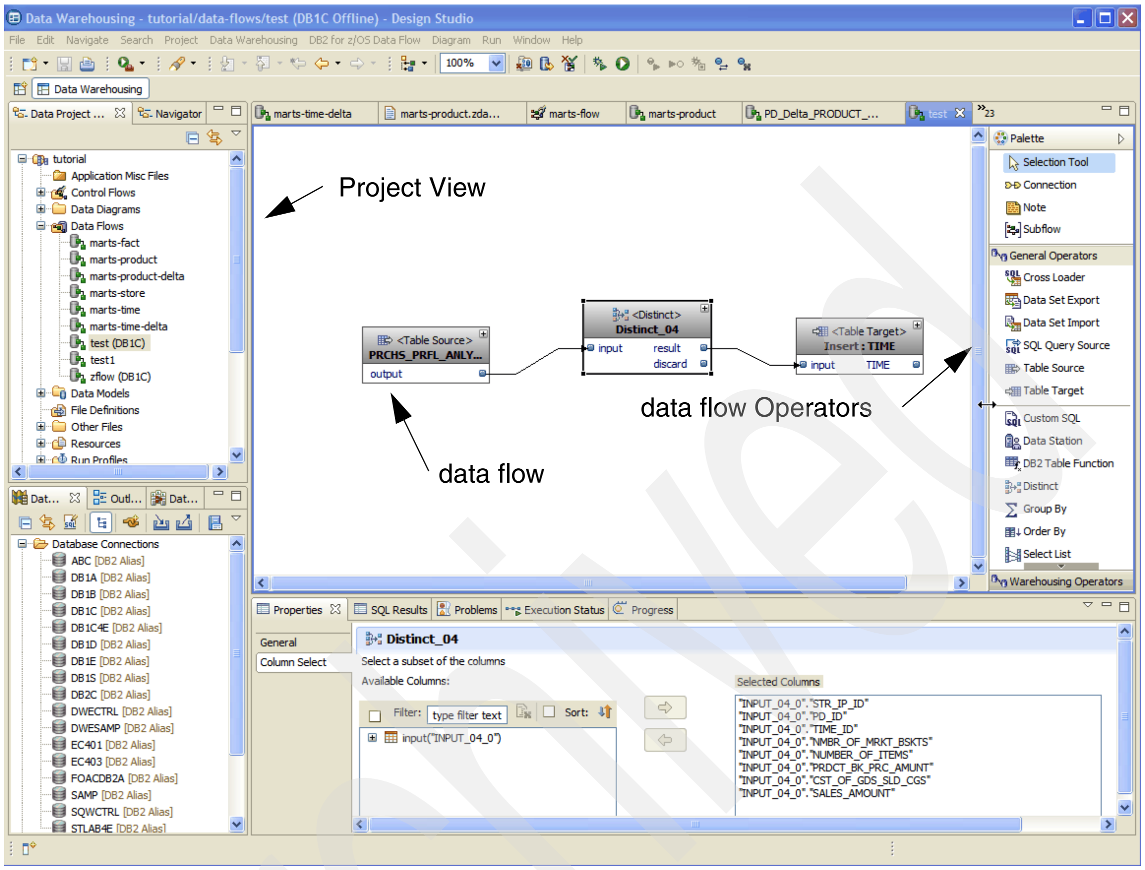
All the modeling and design for the InfoSphere Warehouse is done using an Eclipse-based integrated development environment (IDE) interface called Design Studio. This IDE allows for design and development of data models, OLAP models, data mining models, and intra-warehouse data movement tasks.

Design Studio includes the following tools and features:

- Integrated physical data modeling, based on InfoSphere Data Architect
- SQL Warehousing Tool for data flow and control flow design including integration points with InfoSphere and DataStage ETL systems
- Data visualization tools for data mining and data exploration
- Tools for designing OLAP metadata, MQTs, and cube models

Each tool in InfoSphere Warehouse Design Studio has the same look, feel, and mode of operation. There are standard explorers and views used by all of the component tools. This decreases the learning curve as users move from tool to tool. Figure 2-6 on page 24 shows the InfoSphere Warehouse Design Studio with the Project Explorer, the Database Explorer, standard view pages, and a graphical editor.

The InfoSphere Warehouse Design Studio is based on the popular Eclipse Workbench. Eclipse is a platform of frameworks and tools that make it easy and cost effective to build and deploy software. Eclipse provides an extensible architecture based on the concept of plug-ins and extension points. This allows InfoSphere Warehouse Design Studio to take advantage of code reuse, integration, and many other development functions provided by Eclipse.



```
"INPUT_04_0"."STR_IP_ID"
"INPUT_04_0"."PD_ID"
"INPUT_04_0"."TIME_ID"
"INPUT_04_0"."NMBR_OF_MRKT_BSKTS"
"INPUT_04_0"."NUMBER_OF_ITEMS"
"INPUT_04_0"."PRDCT_BK_PRC_AMUNT"
"INPUT_04_0"."CST_OF_GDS_SLD_CGS"
"INPUT_04_0"."SALES_AMOUNT"
```

Figure 2-6 InfoSphere Warehouse Design Studio

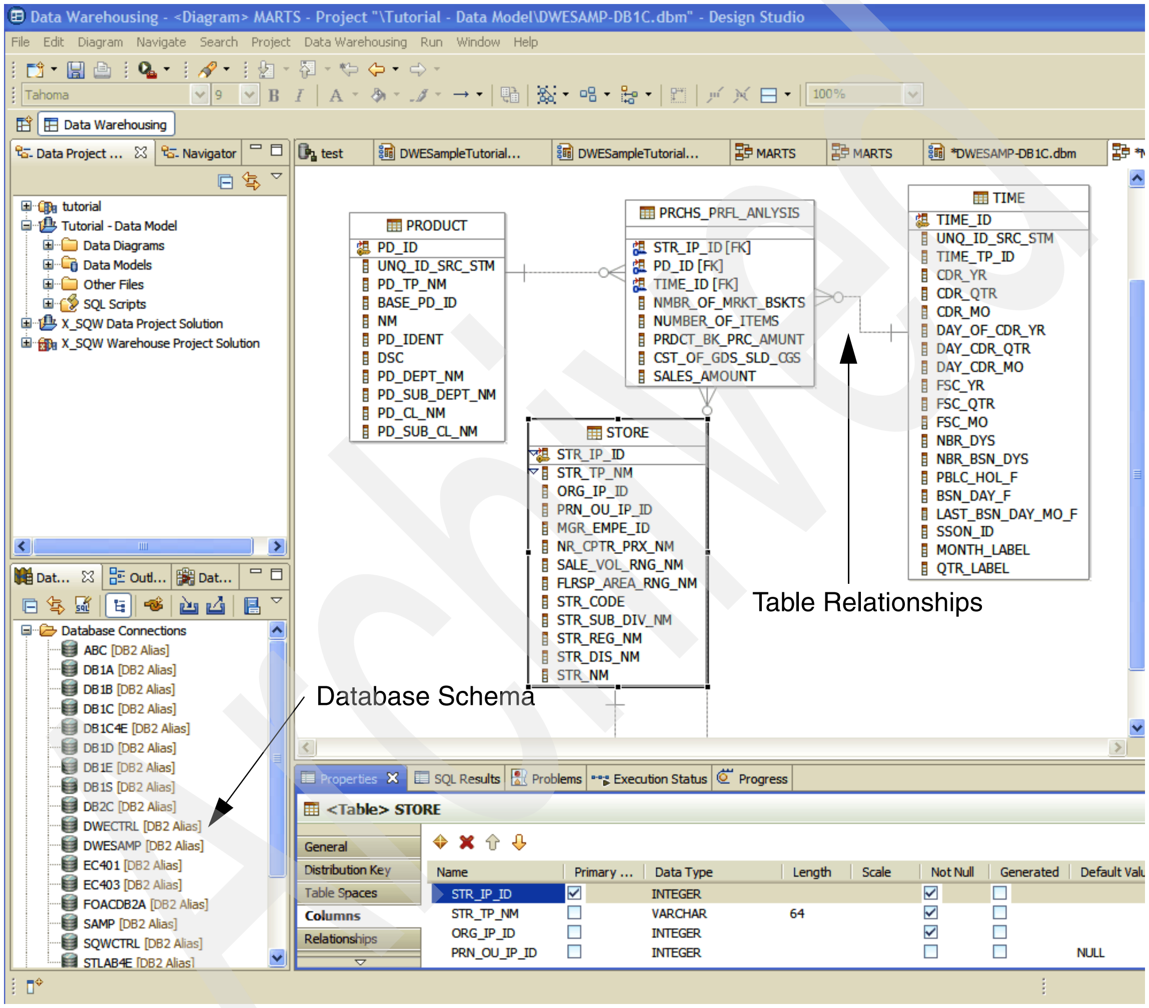
Physical data modeling

The physical modeling component is the subset of the InfoSphere Data Architect tool used to develop and maintain physical data models. Application developers and data warehouse architects use this component to work with physical data models for source and target databases and staging tables.

A physical data model is essentially the metadata representing the physical characteristics of a database. The data model can be newly developed using the data model graphical editor or can be reverse-engineered from an existing database. From the physical data model, you can perform the following tasks:

- Create the physical objects in a database
- Compare a data model to the database (and generate just the delta changes)
- Analyze the model for errors such as naming violations
- Perform impact analysis

The physical model is also used to provide metadata information to the other functional components of InfoSphere Warehouse, such as the SQL Warehousing Tool. Figure 2-7 shows a physical data model open in the Data Project Explorer, a graphical editor view of the physical data model, and the properties view of a selected table. It is not necessary to read or understand the data model, the figure is to provide a depiction of it for familiarity.



<Table> STORE

Name	Primary ...	Data Type	Length	Scale	Not Null	Generated	Default Valu
STR_IP_ID	☑	INTEGER			☑	☐	
STR_TP_NM	☐	VARCHAR	64		☑	☐	
ORG_IP_ID	☐	INTEGER			☑	☐	
PRN_OU_IP_ID	☐	INTEGER			☐	☐	NULL

Figure 2-7 InfoSphere Warehouse physical data model

Administration and control

All InfoSphere Warehouse-based applications are administered using a centralized Web-based interface, the InfoSphere Warehouse Administration Console. This Web-based interface runs on the IBM WebSphere Application Server that is included with the InfoSphere Warehouse suite. Figure 2-8 depicts this environment.

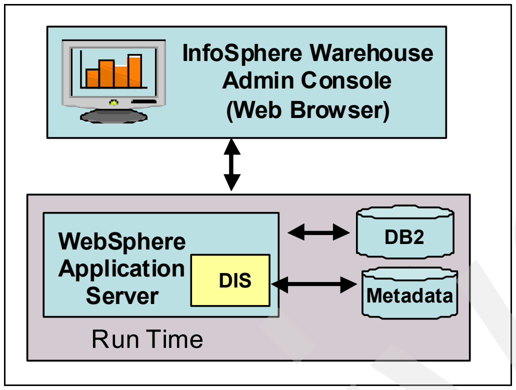


Figure 2-8 InfoSphere Warehouse runtime architecture

The InfoSphere Warehouse Administration Console is used to configure and manage the InfoSphere Warehouse runtime environment. It provides a single tool for administration functions across the functional components to perform the following tasks:

- Create and manage database and system resources and manage SQW processes.
- Run and monitor data warehouse applications and view deployment histories and execution statistics.
- Manage Cube Server, import cube models, explore cube models and cubes, and run the OLAP metadata optimization advisor.
- Enable a database for mining, load, and import and export mining models.
- Configure and use the Workload Manager.

The SQW runtime server, called Data Integration Service (DIS), manages the execution of SQW processes and remote database connections. Processes might be executed on-demand or scheduled through the InfoSphere Warehouse Administration Console. DIS gathers the required information from the metadata database and runs the task using the DB2 execution database, a DataStage server, or perhaps the local operating system as appropriate. DIS monitors the execution of the task and gathers runtime and completion information and stores this information in the metadata database.

2.2 InfoSphere Warehouse topology

InfoSphere Warehouse components are grouped into four categories for the purpose of installation:

- Clients

  The client category includes the InfoSphere Warehouse administration client, InfoSphere Warehouse Design Studio, the workload manager and the data mining visualization programs. Clients are supported on Windows 32-bit and Linux systems.

- Database servers

  This category includes the DB2 database server with DPF support plus the database functions to support InfoSphere Warehouse Intelligent Miner, InfoSphere Warehouse Cubing Services and the workload manager. The database server is supported on AIX®, various Linux platforms, Windows Server 2003 and System z.

- Application servers

  The application server category includes the WebSphere Application Server, DB2 Alphablox server components and InfoSphere Warehouse Administration Console server components.

- Documentation

  This category includes the pdf and online versions of the manuals and can be installed with any of the other categories.

For more details, and current information about InfoSphere Warehouse Installation, see the *InfoSphere Warehouse Installation Guide*, GC18-9800.

The InfoSphere Warehouse components can be installed on multiple machines in a number of topologies. There are three common topologies that are typical, and are depicted in Figure 2-9 on page 28.

- One Tier

  On this tier, the InfoSphere Warehouse Client, InfoSphere Warehouse Database Server, and the InfoSphere Warehouse Application Server are all on one system. This would only be used for development and testing purposes and only on a Windows platform.

- Two Tier

  On this tier, the InfoSphere Warehouse Database Server and the InfoSphere Warehouse Application Server are on one system with InfoSphere Warehouse Clients on separate systems. This could suffice for a test system or for smaller installations. The database and application s could be any of the supported Windows, Linux, or AIX platforms.

- Three Tier

  In any large installation, the InfoSphere Warehouse Database Server, the InfoSphere Warehouse Application Server and the InfoSphere Warehouse Clients should all be installed on separate servers. A DB2 client, at a minimum, is required to connect to database servers. It is a good suggestion that a DB2 server be installed for local access to the runtime metadata databases. The application server is supported on AIX, Linux, and Windows Server 2003.

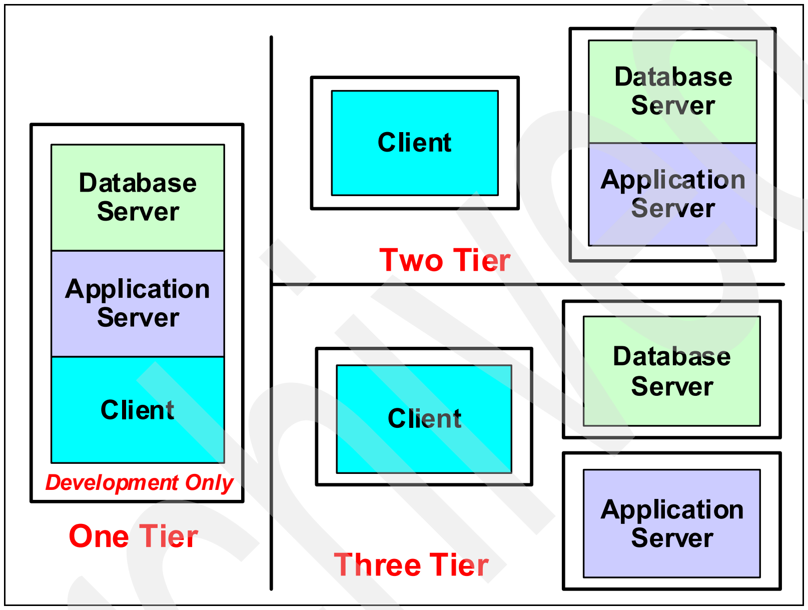


Figure 2-9 Common InfoSphere Warehouse installation topologies

3

InfoSphere Warehouse Design Studio

The InfoSphere Warehouse Design Studio (also referred to as Design Studio, or Design Studio) provides a platform and a set of integrated tools for developing Business Intelligence (BI) solutions. You can use these tools to build, populate, and maintain tables and other structures for data mining and Online Analytical Processing (OLAP) analysis in the context of a DB2 data warehouse.

Design Studio includes the following tools and features:

- Integrated physical data modeling, based on InfoSphere Data Architect
- SQL Warehousing Tool for data flow and control flow design
- Data mining, exploration, and visualization tools
- Tools for designing OLAP metadata, MQTs, and cube models
- Tools for developing AlphaBlox analytical applications
- Integration points with InfoSphere DataStage ETL systems

By integrating these tools, Design Studio offers a fast time-to-value and managed cost for warehouse-based analytics. In this chapter we cover the following topics:

- The Eclipse Platform
- The Design Studio Workbench
- Exploring data
- Designing physical database models
- Designing OLAP objects

- Designing and deploying SQL Warehousing data and control flows
- Designing and deploying data mining flows
- Designing and deploying AlphaBlox applications

Design Studio supports the BI solutions development life cycle.

3.1 The BI solutions development life cycle

When developing BI solutions, you typically follow a common sequence of steps or functions to design, build, and populate the data structures that are required. These functions are typically performed by people in specific roles. The following is a list of a few examples:

- Data architect

  The data architect models the database schemas that are needed to support the analytical solution.

- Warehouse administrator

  The warehouse administrator performs tasks such as creating the tables and ETL or data movement processes or flows to populate the data structures.

- OLAP developer

  The OLAP developer models and creates the OLAP metadata.

- Application developer

  The application developer builds and delivers front-end analytical applications to the business owners in the corporation, using analysis tools such as IBM DB2 AlphaBlox.

- Data steward

  The data steward administers the entire data warehousing environment.

We call this entire sequence the BI solutions development life cycle (shown in Figure 3-1). This life cycle is an iterative process, enhancing the analytic capabilities of the data warehouse by extending the models to incorporate additional business data to be analyzed.

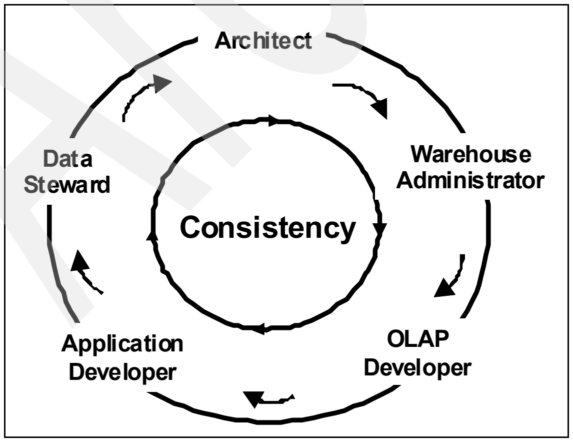


Figure 3-1 BI solutions development life cycle

Typically, these steps are often executed by different people using different tools, which might even be from different software vendors. The use of unintegrated tools makes transitioning between each phase in the life cycle difficult, and requires a great deal of coordination and communication. These factors can have the negative impact of extending the development cycle.

Design Studio delivers a great deal of consistency to the BI solution development process by integrating the tooling that supports that life cycle. This simplifies the transitions between tasks and helps BI teams deliver solutions to their users more quickly and easily.

By integrating your information assets, and applying real-time analytics, to turn your information into intelligence, your data warehousing environment can enable your company to deliver information on demand, that is, selective, transparent access to distributed data sources.

In the remainder of this chapter we discuss Design Studio in more detail. Design Studio is based on the Eclipse platform, which is a powerful development environment. We first provide background on Eclipse, and then delve into Design Studio user interface and common tasks.

3.1.1 The Eclipse platform

Eclipse is an open source community of companies that focus on providing a universal framework for tools integration. The Eclipse consortium was founded in late 2001, and has grown to over 80 members, including many industry-leading software vendors. The Eclipse platform provides a powerful framework and the common graphical user interface (GUI) and infrastructure required to integrate multiple tools easily. The platform is extended by installing plug-ins to provide specific features. InfoSphere Warehouse Design Studio is based upon the open source Eclipse platform.

The basic architecture of Eclipse provides numerous services that tool developers would have to write if they did not use the Eclipse platform. Eclipse has a rich infrastructure (see Figure 3-2 on page 33), including components such as a runtime environment, a generic user interface, and a help system.

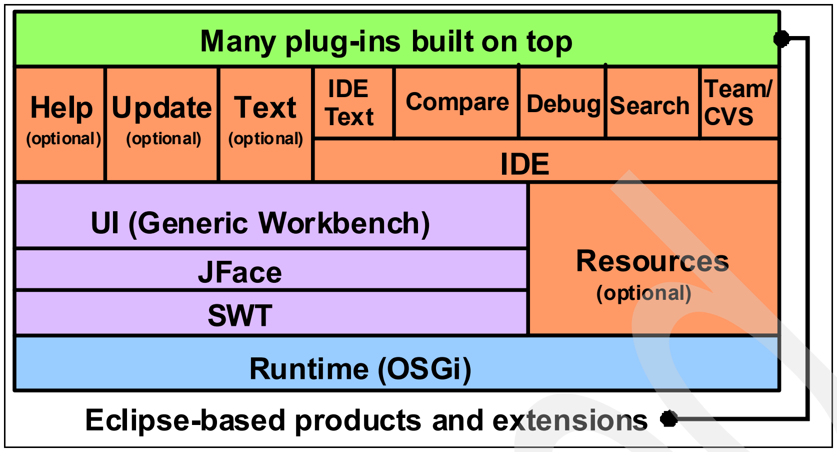


Figure 3-2 Eclipse basic platform

Tool vendors that use Eclipse are able to develop their products quickly. It enables them to focus on their core competency because they only need to build the features in their specialty. The additional capabilities that the tool vendors provide are delivered as plug-ins, which are installed into an existing Eclipse environment. Each of the capabilities that Design Studio provides are packaged together and are installed as plug-ins on top of the basic Eclipse platform.

Note: You do not need to download and install Eclipse before installing InfoSphere Warehouse Design Studio. The Eclipse base product is packaged with Design Studio and is installed with Design Studio.

Users of Eclipse-based tools enjoy many benefits:

- A rich user experience that is common across all Eclipse-based products (such as InfoSphere Warehouse Design Studio, WebSphere development tools, and the suite of Rational tools).
- A wide array of instructional resources on the Internet that explain how to extend the Eclipse platform or write tools for it.
- A broad selection of third-party tools that have been developed and are available to be installed into InfoSphere Warehouse Design Studio.

For more information about Eclipse and its community, go to the following web page:

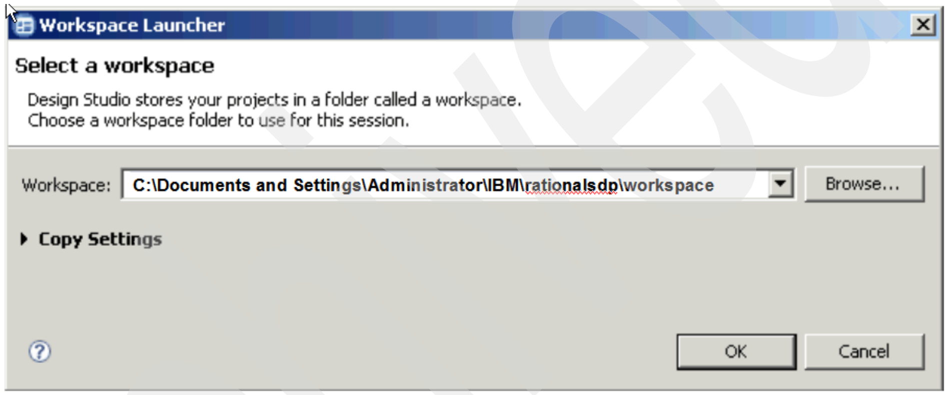
http://www.eclipse.org

3.2 Introduction to Design Studio

In this section we provide an introduction to InfoSphere Warehouse Design Studio. The Workbench is the interface you use to access and use the capabilities of Design Studio. But before you can begin using the Workbench, there are basic concepts with which you must become familiar.

3.2.1 The workspace

Every time InfoSphere Warehouse Design Studio is launched, you are prompted to provide a path to the workspace, as shown in Figure 3-3. A workspace is a collection of resources and is the central repository for your data files.



Design Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Documents and Settings\Administrator\IBM\rationalsdp\workspace

Figure 3-3 Prompt for the workspace location

InfoSphere Warehouse Design Studio, like other Eclipse-based tools, helps manage various resources. These resources take the form of projects, folders, and files.

A project is a container used to organize resources pertaining to a specific subject area. The workspace resources are displayed in a tree structure with projects containing folders and files being at the highest level. However, projects do not contain other projects.

You can specify different workspaces for different projects, but only one workspace is active per running instance of Design Studio. To change workspaces, choose **File** → **Switch Workspace**. A workspace can hold multiple projects.

If you specify a local directory for your workspace, back up that directory on a regular basis. Another option is to use teaming software such as Concurrent Versions System or Rational ClearCase®. See 3.3.5, “Team component” on page 52, for more information.

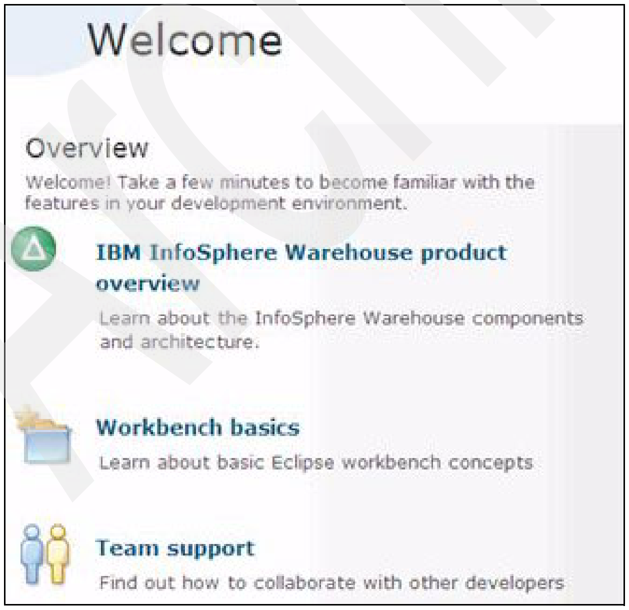
3.2.2 Projects and the local file system

When you create a new project, you find a new subdirectory on disk, located under the workspace directory specified at start up. In the project directory is a special file with a `.project` extension. The `.project` file holds metadata about the project, including information that can be viewed by selecting the Properties view in Design Studio. Inside the project subdirectory you see all of the files and folders that have been created as part of the project. The file names and content are the same, whether accessed from the file system or through Design Studio. You also see a folder with a `.metadata` extension, located in the workspace directory, at the same level as the projects that are part of that workspace. The `.metadata` directory holds platform-specific information, (including workspace structure information). The contents of this directory should never be altered or manipulated outside of the Design Studio API.

The project type controls or determines the kinds of objects that are available. In the InfoSphere Warehouse Design Studio, the data design project (OLAP) type allows you to work with physical data models and OLAP objects. The data warehousing project provides entries such as SQL warehousing objects, including data flows, control flows, and mining flows.

3.2.3 The Welcome page

The first time that Design Studio is started in a new workspace, the Welcome page is displayed. A partial view of that page is depicted in Figure 3-4 for familiarity.


Welcome

Overview

Welcome! Take a few minutes to become familiar with the features in your development environment.

IBM InfoSphere Warehouse product overview

Learn about the InfoSphere Warehouse components and architecture.

Workbench basics

Learn about basic Eclipse workbench concepts

Team support

Find out how to collaborate with other developers

Figure 3-4 The Welcome page for Design Studio

The Welcome page contains links to general information, the help system, recorded demos, and tutorials about InfoSphere Warehouse and Design Studio. InfoSphere Warehouse also provides sample projects that you can work through to become familiar with the workbench, and the steps involved in creating data models, data and control flows, and data mining flows.

Click the Workbench link (the arrow icon displayed at the top of the window) or close the Welcome page to launch the workbench. After you have launched the workbench, you can display the Welcome page by selecting **Help** → **Welcome**.

3.3 The Design Studio Workbench

The term *workbench* refers to the desktop development environment, and delivers the mechanism for navigating the functions provided by the various Design Studio plug-ins. The workbench offers one or more windows, which contain one or more perspectives, views, and editors that allow you to manipulate resources in your project. The default workbench for InfoSphere Warehouse Design Studio contains the following elements:

- Opened Perspective
- Menu Bar
- Project Explorer
- Editors
- Palette for Editors
- Stacked Views
- State Bar
- Data Sources Explorer
- Properties View

These elements are described in more detail in subsequent sections of this chapter. However, be aware that what you see in your own workbench environment might vary, based upon which element has focus.

3.3.1 Perspectives

Perspectives define an initial layout of views and editors in a workbench window. Perspectives provide the functionality required to accomplish a particular task or work with a particular resource.

Perspectives also control what options might appear in menus and task bars. They can be customized or modified and saved for reuse by selecting **Window** → **Save Perspective As**. If you have rearranged views or closed views, you can reset the perspective by choosing **Window** → **Reset Perspective**.

There are a number of perspectives in the InfoSphere Warehouse Design Studio, but the following perspectives are worked with most often:

- Data warehousing

  This is the default perspective for InfoSphere Warehouse Design Studio. It includes functions that are tailored for building information warehouses and enabling warehouse-based analytics such as OLAP and data mining.

- Data

  The data perspective provides physical data modeling functions, such as the ability to reverse engineer from existing data structures, to compare data objects, and to analyze models against a set of enterprise rules and standards.

- Team synchronizing

  These perspectives are used for source repository management functions such as synchronization and version control.

To change to a new perspective, select **Window** → **Open Perspective** or click **Open Perspective** on the shortcut bar on the left of the Workbench window. The current perspective is always reflected on the title bar of the Design Studio workbench window. An icon is also added to the shortcut bar to enable quick switching between open perspectives.

3.3.2 Editors

In Design Studio, there are different editors available for different types of files. Text and SQL editors are provided for resources such as SQL scripts, and diagram editors are available for resources such as data models. An editor is a visual component that you use to edit or browse a resource. Modifications that you make in an editor are not always automatically saved. Therefore, it is a good practice to save your changes. Tabs in the editor area reflect the names of the resources that are open for editing. In the tab, an asterisk (*) by the resource name indicates that there are unsaved changes to that resource. The border area on the left margin of the editing window might contain icons that indicate errors and warnings.

The editors that are available in the Data Warehousing perspective depend on the type of object with which you are working. The editors usually include a customized palette located to the right of the canvas.

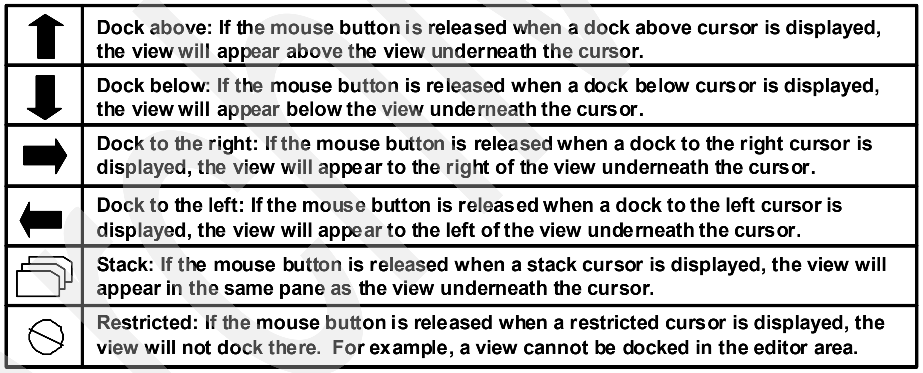
The editors that are available in Design Studio to work with data warehouse objects include:

- Physical data model editor
- SQL Scripts editor
- Data flow editor
- Control flow editor
- Data mining flow editor

3.3.3 Views

A view is a component that you use to navigate a hierarchy of information, open an editor, or display properties for the active editor. Modifications that you make in a view are saved immediately.

Perspectives, which are combinations of views and editors, might be arranged on the window to your liking. Views might be docked and stacked by grabbing the view's title bar and dragging from one area of the UI to another. As the view is dragged in the workbench window, notice a drop cursor (Figure 3-5) is displayed, reflecting where the view will be docked.



⬆	**Dock above: If the mouse button is released when a dock above cursor is displayed, the view will appear above the view underneath the cursor.**
⬇	**Dock below: If the mouse button is released when a dock below cursor is displayed, the view will appear below the view underneath the cursor.**
➡	**Dock to the right: If the mouse button is released when a dock to the right cursor is displayed, the view will appear to the right of the view underneath the cursor.**
⬅	**Dock to the left: If the mouse button is released when a dock to the left cursor is displayed, the view will appear to the left of the view underneath the cursor.**
⧉	**Stack: If the mouse button is released when a stack cursor is displayed, the view will appear in the same pane as the view underneath the cursor.**
⃠	**Restricted: If the mouse button is released when a restricted cursor is displayed, the view will not dock there. For example, a view cannot be docked in the editor area.**

Figure 3-5 Drop cursor behavior

Views might also be closed by clicking the **X** located on the right side of its title bar. When closed, a view might be displayed again by clicking the menu option **Window** → **Show View** and selecting the view you want to display. To maximize a view or editor, double-click its title bar. Double-clicking the title bar of a maximized view or editor returns it to its original size.

Several views are available in the data warehousing perspective, but those in the following sections are the ones you work with most often.

Data Project Explorer view

By default, this view opens in the upper left area of the Design Studio window. We have highlighted it with a box around it in Figure 3-6. We also show the contents of that box on the right side of the figure. The Data Project Explorer shows a logical representation of the currently opened projects. The representation is logical because the name of the folders and resources represented here need not match the real files and directories on the file system. For example, an SQLW data flow is physically made up of two files, but is represented in the Project Explorer as a single node.

With this view you can navigate your projects and the objects in your projects. You work with this view most often to make changes to your objects. Many actions can be triggered from the Project Explorer by right-clicking an element in the tree.

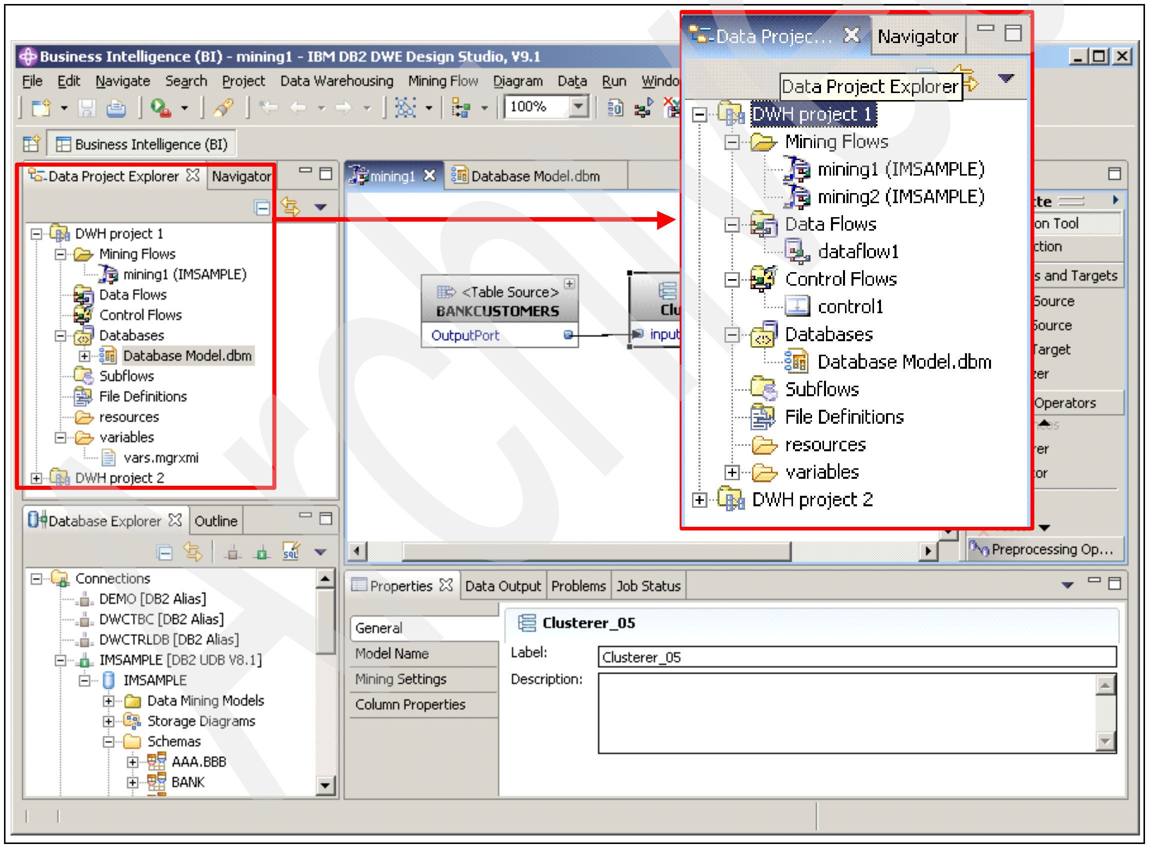


Figure 3-6 Data Project Explorer view

Navigator view

The Navigator shows a physical representation of the files and directories of the opened projects on the file system. While you typically work with the logical Data Project Explorer, there are functions that can only be performed in the Resource Navigator. For example, this view enables the ability to copy files or project elements such as data flows from one project to another.

Data Source Explorer view

By default, the Data Source Explorer opens in the lower left area of Design Studio, highlighted in Figure 3-7. This view enables you to connect to and explore a database. You can make the changes to the database that you have the proper authority and privileges to complete.

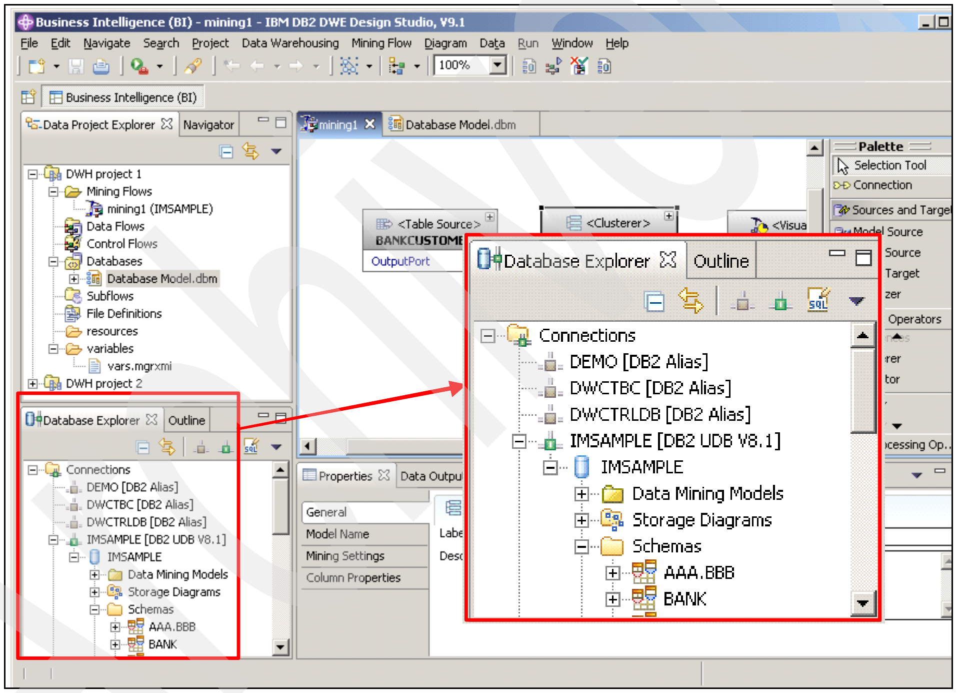


Figure 3-7 Data Source Explorer view

The Data Source Explorer view is used to complete the following tasks:

- Create JDBC connections to databases.
- Explore database content including schemata, table relationships and content, and value distributions.
- Generate storage overview diagrams from databases, and overview diagrams of schemas using either Information Engineering (IE) notation or Unified Modeling Language (UML) notation. Diagrams are a helpful way to visualize data warehousing projects.
- Reverse engineer databases.
- Compare database objects.
- Analyze a database or a schema to ensure that it meets certain specifications.

  Model analysis helps to ensure model integrity and helps to improve model quality by providing design suggestions and best practices.
- Analyze the impact of changes to models.

  You can also use the Impact Analysis features to find dependencies. For example, if you want to copy a schema from the Data Source Explorer to the Data Project Explorer, you can find dependencies on the schema to ensure that all references are resolved. You can analyze data objects in the Data Source Explorer, the Data Project Explorer, or in the data diagram.
- Create new database objects.
- Drag database objects to the Data Project Explorer.

Outline view

The Outline view shows an overview of the structural elements of the file that is currently open for editing. The contents of this view depend upon the nature of the edited file. When flows or diagrams are edited, the outline offers two representations:

- Tree representation

  In this representation, objects composing the flow can easily be selected.
- Graphical representation

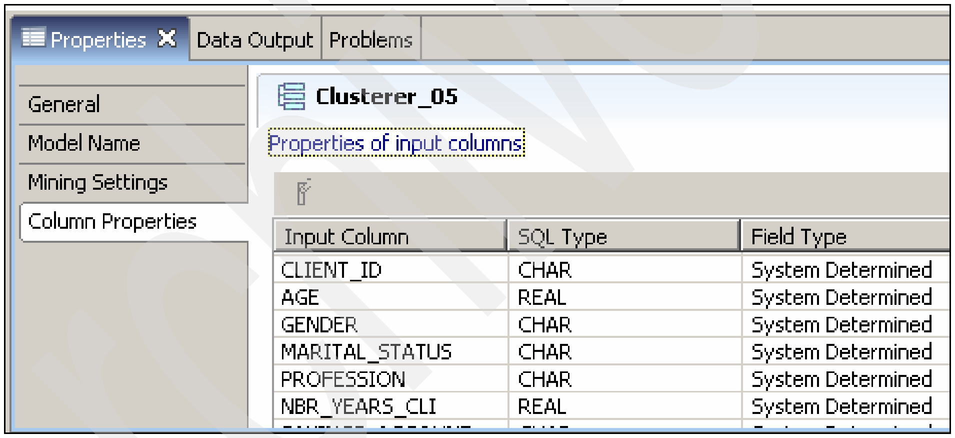
  This representation shows the entire flow or the entire entity relationship diagram.

The Outline view is particularly helpful when you are working with large flows or diagrams. You can use the Outline view to find your location in a large diagram. The Outline view shows the entire diagram with a small gray box representing the viewing area. You can drag the gray box to display the portion of the diagram on which you want to work. The Outline view is stacked with the Database Explorer view, so you need to click its identifying tab to bring it to the forefront.

Properties view

The Properties view, shown in Figure 3-8, is taken from the Outline View that is stacked with the Database Explorer View. It allows you to view and modify the properties of the current selected object. The edit capabilities of this view are dependent on the type of object that is currently selected. This is one of the most important views when designing flows or database objects. The properties of the newly created objects are primarily edited in this view.

Tip: When objects are selected in the Data Source Explorer view, the Properties view is read-only. To edit the properties of an object you must select it from the Data Project Explorer view.



Input Column	SQL Type	Field Type
CLIENT_ID	CHAR	System Determined
AGE	REAL	System Determined
GENDER	CHAR	System Determined
MARITAL_STATUS	CHAR	System Determined
PROFESSION	CHAR	System Determined
NBR_YEARS_CLI	REAL	System Determined

Figure 3-8 The Properties view in InfoSphere Warehouse Design Studio

SQL Results view

The SQL Results view is used to see messages, parameters, and results of the objects that you are working with. The SQL Results view displays the results of various actions when they are executed on a database. Use this view to inspect the contents of a table through the Sample Contents feature, execute a data mining flow, or execute an SQL or DDL script or perform any operation on a database. The SQL Results view is stacked with the Properties view.

The SQL Results view is divided into two parts. The left part of the view contains a read-only table with four columns:

- Status (indicates the state of the associated action)
- Operation (indicates what kind of action occurred)
- The data of the action
- The database connection of the associated action

The sort order of this view can be changed by clicking the header of any of the columns.

The right side of the Data Output view contains two tabs, as depicted in Figure 3-9:

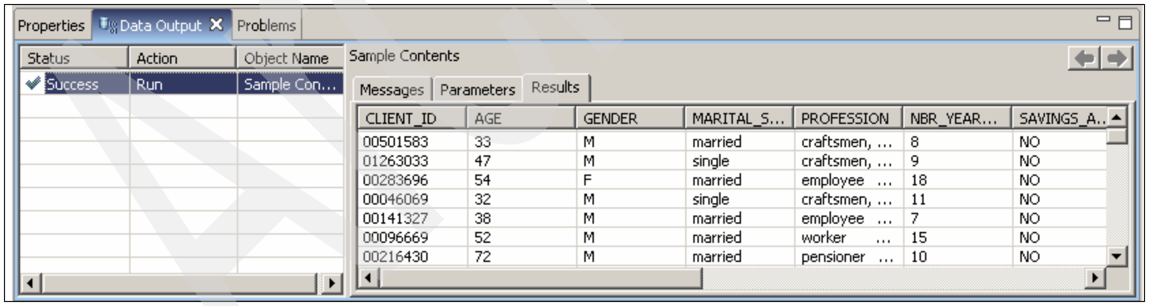
- Status

  This tab shows messages generated while the statement was being run. If there are errors in the SQL statement, an error message appears on this page. The SQL source of the statement being run is shown in this view also. See your database product's SQL documentation to check the validity of the structure of the SQL statement. Edit the statement by making the changes in the SQL builder or SQL editor as necessary, and run the statement again.

  For INSERT, UPDATE, and DELETE statements, a message is displayed on this page if the statement runs successfully. If an INSERT, UPDATE, or DELETE statement runs successfully, the database is modified.

- Results

  This tab contains any results that were generated from running the statement (for example, a table of sales data). The Results tab is selected by default.



Status	Action	Object Name
✓ Success	Run	Sample Con...

CLIENT_ID	AGE	GENDER	MARITAL_S...	PROFESSION	NBR_YEAR...	SAVINGS_A...
00501583	33	M	married	craftsmen, ...	8	NO
01263033	47	M	single	craftsmen, ...	9	NO
00283696	54	F	married	employee ...	18	NO
00046069	32	M	single	craftsmen, ...	11	NO
00141327	38	M	married	employee ...	7	NO
00096669	52	M	married	worker ...	15	NO
00216430	72	M	married	pensioner ...	10	NO

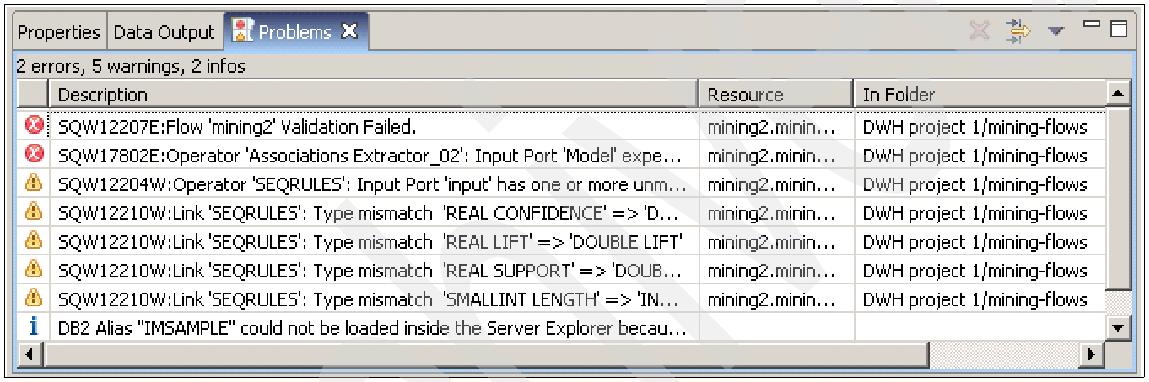
Figure 3-9 The SQL Results view

Problems view

While working with the resources in your data warehousing projects, certain editors might log problems, errors, or warnings for the Problems view, as shown in Figure 3-10. The Problems view is stacked with the Properties view and the SQL Results view. The Problems view shows three levels of severity:

- Errors
- Warnings
- Information

These messages can be sorted by their severity, the resource name, or the problem description. Messages can also be filtered by severity or resource. From the problem list, you can double-click an error, which launches the editor for the resource containing that error, with the corresponding object highlighted.



2 errors, 5 warnings, 2 infos

	Description	Resource	In Folder
⊗	SQW12207E:Flow 'mining2' Validation Failed.	mining2.minin...	DWH project 1/mining-flows
⊗	SQW17802E:Operator 'Associations Extractor_02': Input Port 'Model' expe...	mining2.minin...	DWH project 1/mining-flows
⚠	SQW12204W:Operator 'SEQRULES': Input Port 'input' has one or more unm...	mining2.minin...	DWH project 1/mining-flows
⚠	SQW12210W:Link 'SEQRULES': Type mismatch 'REAL CONFIDENCE' => 'D...	mining2.minin...	DWH project 1/mining-flows
⚠	SQW12210W:Link 'SEQRULES': Type mismatch 'REAL LIFT' => 'DOUBLE LIFT'	mining2.minin...	DWH project 1/mining-flows
⚠	SQW12210W:Link 'SEQRULES': Type mismatch 'REAL SUPPORT' => 'DOUB...	mining2.minin...	DWH project 1/mining-flows
⚠	SQW12210W:Link 'SEQRULES': Type mismatch 'SMALLINT LENGTH' => 'IN...	mining2.minin...	DWH project 1/mining-flows
i	DB2 Alias "IMSAMPLE" could not be loaded inside the Server Explorer becau...		

Figure 3-10 The Problem view

3.3.4 Common tasks

Now that you are familiar with the Design Studio user interface, you are ready to learn how to use Design Studio with your data warehousing environment. In the remainder of this chapter we review the major tasks performed in Design Studio.

Creating projects

Data warehouse projects are containers for the development of data warehouse applications in Design Studio. Before you can use Design Studio to perform any of the processes in the BI Solutions Development life cycle, you must create a project. The data warehouse project contains artifacts (such as your data schemas, SQL control flows, and mining flows).

To create a project, from the main menu select **File** → **New** → **Project** → **Data Warehousing**. There are two types of projects provided by Design Studio:

- Data design projects (OLAP)

  Design database physical models, including OLAP models. You can engineer or reverse-engineer database models.

- Data warehousing projects

  Design SQL Warehouse and data mining flows. This project type also provides the mechanisms to generate data warehouse application packages to deploy on the Information Warehouse server. Data warehouse projects can also contain your physical data models.

Data warehouse projects sometimes depend on metadata (for example, a physical data model) that is stored in data design projects. You can link a data warehouse project to a data design project with the following steps:

1. Right-click the data warehouse project and select **Project References** from the context menu.

   Note: If you delete a linked physical data model (`.dbm` file) from a data warehouse project, you are deleting the original model file, not just the link to that model. Instead, remove the project reference to that data model.

2. Select from the list of available data design projects.

When a warehouse project refers to a data design project, you have access to the metadata in that project when you are designing data flows and mining flows. Linked resources are indicated in Project Explorer. For example, in the Data Models folder of a data warehousing project, we have a resource named `GoSalesDW.dbm` (linked from Warehouse 97OLAP Lab).

Important: Do not rename or copy and paste a data warehouse project. These actions have unpredictable results and are likely to cause problems with the objects inside the project, such as subflows and application profiles.

You can import data flows and other objects into a data warehouse project by selecting **File** → **Import** → **File system**. Be sure to identify the correct folder name for the object that you want to import. For example, if you are importing a data flow, specify the data-flows folder as the target directory in your project. If you do not select the correct folder, the results of the import operation are unpredictable.

Adding database connections

Every time Design Studio is launched, the Data Source Explorer displays all local DB2 databases, any remote DB2 databases that have been cataloged on the IW client machine, and any previously defined non-DB2 database connections. Before you can browse or explore a new database, you must define a database connection to the data source.

Note: The database connections that you define in the Data Source Explorer view are used during the design or development phase of your data warehouse project. They define a connection from your Design Studio client to the database server. When you are ready to deploy your project, you must define a run-time database connection through the administration console. For information about how to do this see **"Configuring database connections" on page 370**.

The database connections are represented in the Data Source Explorer view by a node. The connections are initially disconnected (no [+] sign on their left side). You can connect to a database by right-clicking its connection node and choosing **Connect**. If the user ID and password have not been saved in this connection profile, you are prompted for a user ID and password. After the connection has been made, you can expand the tree to explore its schemata, tables, and other objects.

If you want to add another remote database or non-DB2 database to your Database Explorer view, right-click the **Connections** node and select **New Connection**. A dialog box collects the connection information (such as database vendor and version, database name, port number, JDBC driver class location, and user ID and password). You can also specify filters for the database connection if it is appropriate to do so. After this information is supplied, the database is added to the Connections list. For more information about configuring database connections see the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.datatools.accommon.doc/db_connections.html

If you are adding connections on System z, see the following web page:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp

Browsing databases and exploring data

Once you are successfully connected to a database in the Data Source Explorer, there are many ways to explore your data in Design Studio. You can explore schemata, table relationships, table content, or value distributions in your data.

Note: A few of the features in this section are not available on System z. As examples, the data mining features in the "Exploring value duistributions" sub-section, and options in the “Editing loading and extracting data” sub-section.

This is often the first step in creating a mining project. By exploring your database objects, you can select the appropriate data for mining analyses.

- Exploring database schemata

  To expand a database you want to explore, double-click its icon or click the [+] to the left of the database name. You can view items (such as storage diagrams, schemas, and tables) and view relationships between tables and columns. After selecting a table, use the Properties view to display this information.

- Viewing sample content

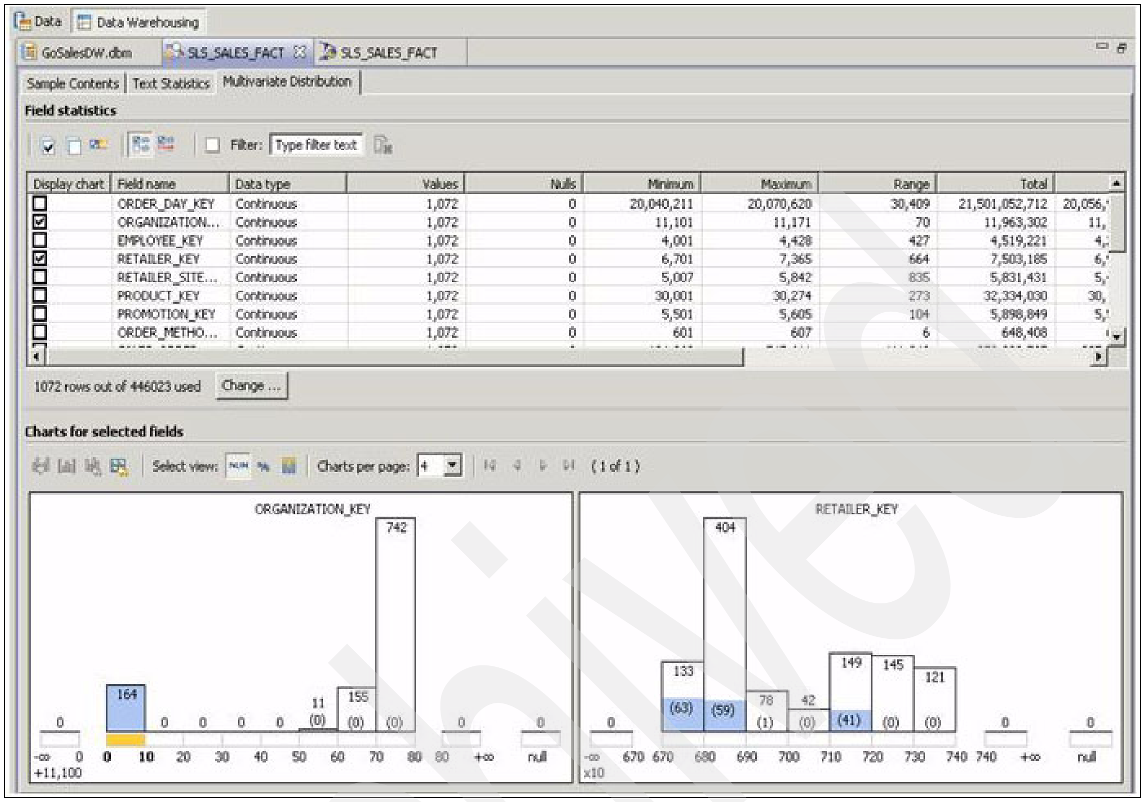
  Inspect the content of single tables by expanding the explorer tree and viewing the schemata and tables. Right-click a table and choose **Data** → **Sample Contents**. The SQL Results view displays a sampling of the data. The number of rows selected for the sample is controlled by the output preference setting, found under the data category, as described in “Customizing the environment” on page 49.

- Exploring value distributions

  If you have enabled the database for data mining, you can look at data value distributions for a particular table. This feature also shows other statistical information. Right-click the table name and choose **Distribution and Statistics**. Then choose **Multivariate**, **Univariate**, or **Bivariate**.

  The multivariate option allows you to explore value distributions and relationships between the columns of a table. For each field in the table, a chart is displayed that shows the distribution of the values in the chart. Multiple fields can be compared, as shown in Figure 3-11 on page 48, by selecting the field name in the statistics table that is displayed above the chart.

  For more information about the value distribution options see Chapter 7, “Data mining” on page 243.



Display chart	Field name	Data type	Values	Nulls	Minimum	Maximum	Range	Total
☐	ORDER_DAY_KEY	Continuous	1,072	0	20,040,211	20,070,620	30,409	21,501,052,712
☑	ORGANIZATION...	Continuous	1,072	0	11,101	11,171	70	11,963,302
☐	EMPLOYEE_KEY	Continuous	1,072	0	4,001	4,428	427	4,519,221
☑	RETAILER_KEY	Continuous	1,072	0	6,701	7,365	664	7,503,185
☐	RETAILER_SITE...	Continuous	1,072	0	5,007	5,842	835	5,831,431
☐	PRODUCT_KEY	Continuous	1,072	0	30,001	30,274	273	32,334,030
☐	PROMOTION_KEY	Continuous	1,072	0	5,501	5,605	104	5,898,849
☐	ORDER_METHO...	Continuous	1,072	0	601	607	6	648,408

1072 rows out of 446023 used Change ...

Charts for selected fields

Select view: Charts per page: 4 (1 of 1)

Figure 3-11 Multivariate value distributions

- Editing, loading, and extracting data

  In addition to sampling data contents, there are other actions that are available from the Data context menu. To get to the Data context menu, expand a database tree in the Data Source Explorer view, select a table, right-click, and chose **Data**. The following actions are available from the context menu:

  – Edit

    With the proper database authority, edit table contents in an editor.

  – Load

    Load the table with data from a flat file.

  – Extract

    Write table contents to a flat file through the Extract option.

– Extract as XML

  Write table contents to an MXML file.

– Load from XML

  Load the table with data from an XML file.

Working with databases offline

Database connection information can be saved locally to allow database objects to be viewed from Design Studio without having an active connection to the database. Not all features are available without an active connection. Capabilities that can be performed include viewing database objects and their properties and creating and viewing overview diagrams.

To work with a database connection offline, perform the following steps:

1. While connected, refresh your database connection.
2. Right-click the active database connection and select **Save Offline**. The connection information is saved to your local Design Studio client.
3. Right-click the same active database connection and choose **Disconnect**.
4. Right-click the same connection again and choose **Work Offline**. The locally saved database information is retrieved.
5. To return to the active connection, right-click and choose **Disconnect**. Right-click the connection and choose **Reconnect**.
6. If you want to work disconnected again, refresh your local connection by re-performing these steps to be sure that you have any changes that were made to the database since you were last connected.

Customizing the environment

You can modify many aspects of the behavior and appearance of your Design Studio workbench. For example, you can change the fonts and colors used, control the placement of tabs, and configure the behavior of plug-ins installed in your Eclipse environment. To customize your Design Studio environment, from the menu select **Window** → **Preferences**. The preferences dialog box allows you to set preferences for tools you have installed in Design Studio.

Note: Not all the data mining preferences that are available in LUW are available in System z.

The tree on the left of the dialog box displays all of the categories that can be customized. The following list details important preferences to review:

- General

  Configure the general appearance and behavior of the workbench. This is where you can set fonts and colors, configure keyboard shortcuts, and configure startup and shutdown behavior.

- Data Management

  Configure the preferences for data modeling and data exploration features. You can perform the following tasks:

  - Control the number of rows returned when the option to sample contents runs.
  - Set the notation type used in the model diagrams.
  - Configure the naming standards for physical tables, columns, indexes, and various constraints.

- Data Warehousing

  Set the preferences for the SQLW and data mining features (such as the colors used for links, operators, and operator ports).

- Modeling

  Define the appearance of the data model objects including constraints, comments, and notes. This is where validation rules for constraints are enabled and disabled.

- Team

  Configure CVS or any other version control system solution you are using.

Modeling data structures

As part of the process of developing your BI solution, you design and modify physical database models and define schemas and storage specifications. To get started, open the data design project in which you want to work, and launch the New Data Model Wizard by right-clicking the Data Models folder and choosing **New** → **Physical Data Model**. Choose the destination project folder, provide a name for your data model, and continue to follow the prompts from the wizard. Models can be created from scratch or reverse engineered from existing DDL or databases. Other tasks that are performed as part of this phase in the development life cycle are as follows:

- Designing tablespaces and containers for DB2 databases.
- Comparing data objects and models with each other or with other objects.
- Analyzing designs to confirm that standards are being met.
- Deploying the model.

For more information about physical data modeling, see Chapter 4, “Developing the physical data model” on page 53.

Creating OLAP models

An OLAP model reflects the dimensionality of data that is typically found in a star schema-like structure to reflect objects such as facts, measures, dimensions, and hierarchies. This metadata is important to BI tools. You can create this extra metadata by defining OLAP models and cubes as part of the physical data model. Design Studio also has wizards to optimize access to relational OLAP cubes in DB2. For more information about OLAP modeling see Chapter 6, “InfoSphere Warehouse Cubing Services” on page 169.

Designing data flows and control flows

Data flows and control flows are housed in the data warehouse project type. Data flows model data transformation steps that move data from a relational table or file, perform processing, and insert relational or file targets. From the data flow models, SQL-based code is generated to accomplish the tasks defined in the flow.

A control flow determines the processing sequence of data flows, mining flows, and other types of common data processing tasks (such as executing OS scripts, DB2 scripts, FTP, and email). It allows the construction of success/failure paths and iterative loops.

Creating data mining models

Data mining functions are SQL-based tasks for the creation and application of data mining algorithms and execute in the DB2 database engine, and so do not require data to be extracted from the database. Data mining models can be created using data mining flows, which allow the graphical development of data mining data preparation, the creation of data mining models, visualization of data models, and the application (or scoring) of the model.

Developing AlphaBlox Reports using BloxBuilder

BloxBuilder is an AlphaBlox interface that can be used to develop analytic applications without requiring Java or JSP development skills. The BloxBuilder interface is accessed through the BloxBuilder perspective. For more information about developing reports with Blox Builder, see IBM Redbooks publication *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582. Information about installing BloxBuilder can be found at the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.abx.ins.doc/abxinst51.htm

3.3.5 Team component

If you have several developers working on your data warehousing projects in Design Studio, you might want to set up a version control system. A version control system keeps track of changes to files and allows developers who might be physically separated to collaborate on design projects. Design Studio offers integrated support for two version control solutions, CVS and IBM ClearCase. The integration points allow you to perform version control tasks directly from your database project explorer.

For more information about these solutions, see the following web pages:

- IBM ClearCase

  http://www-01.ibm.com/software/awdtools/clearcase/

- CVS

  http://www.cvshome.org

4

Developing the physical data model

In 3.1, “The BI solutions development life cycle” on page 31, we explain the concept of the business intelligence (BI) solutions development life cycle. The first step in the life cycle is to design a physical model of the data structures needed to support the BI solution.

The data infrastructure for any business is built around data models. There are three types of data models:

- Conceptual data model

  This is a high level business view of the data. It identifies the data entities used in the business and their relationship to each other. This aids in the understanding and development of the logical and physical data models.

- Logical data model

  This is where the data model actually gets defined. This is where the data entities are identified in detail along with any existing relationships between them. These are normalized entities that are described by specific attributes and data types, and candidate keys that are used for direct access to data.

- Physical data model

  This is the physical implementation of the logical model in a specific database management system. It identifies the implementation details in terms of such things as the configuration, keys, indexes selected, and constraints, and the means by which it is physically stored in the selected database structures.

  This is a major step in that life cycle. To help with this process, the InfoSphere Warehouse Design Studio includes a subset of the functionality provided in the Rational Data Architect (RDA) product.

In this chapter we describe the physical data modeling capabilities in the InfoSphere Warehouse Design Studio. InfoSphere Warehouse includes the physical data modeling and corresponding SQL generation capabilities to implement changes to the physical model.

4.1 Physical data models

Physical data models define the internal schema of the data sources in your warehouse environment. Data models outline data tables, the columns in those tables, and the relationships between tables. Design Studio includes the components needed to create a physical model and generate the SQL appropriate for your implementation target. Physical models are constrained to the concept of the target. For example, InfoSphere Warehouse physical models are constrained to the relational model. You can only model objects that are supported by the target database. For example, the ability to model MQTs only applies to DB2 targets.

The physical models that you create in Design Studio can be used to create brand new schemas or to update schemas that already exist in the target database.

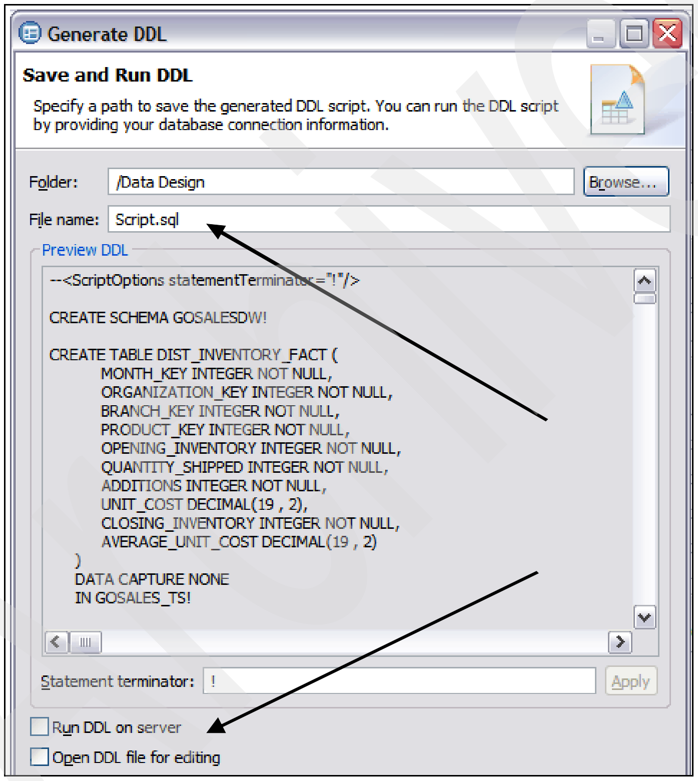
Using Design Studio, in your physical data models, you can define the following data elements for the target database:

- Databases (one per data model)
- Tables
- Views
- Primary keys and foreign keys
- Indexes
- Stored procedures and functions
- DB2 Tablespaces and buffer pools

The physical data models that you create and update in Design Studio are implemented as entity relationship diagrams. The models are visually represented using either Information Engineering (IE) notation, or Unified Modeling Language (UML) notation. Before you begin your modeling work, configure Design Studio to use the notation that you prefer. Configure Design Studio by clicking **Data** → **Diagram**, which takes you to the configuration screen where you can select the desired notation. For more information about configuring and customizing this setting, see 3.3.4, “Common tasks” on page 44.

4.1.1 Physical model structure

Physical models are displayed in the Data Project Explorer view of Design Studio. They are identified by a `.dbm` extension at the end of the model name. All physical models share a common structure, regardless of whether all objects are present in the model. In Figure 4-1 we show the components of a physical data model. There is one database per physical model. in the database, you can have OLAP objects, SQL statements, schemas, buffer pools, partition groups, tablespaces, users, groups, and roles. Each schema has a number of tables and a logical folder for diagrams. If you expand the table in the Data Project Explorer view, you see definitions for columns, including primary key definitions, constraint specifications, and indexes.



Specify a path to save the generated DDL script. You can run the DDL script by providing your database connection information.

Folder: /Data Design

File name: Script.sql

```
--<ScriptOptions statementTerminator="!"/>

CREATE SCHEMA GOSALESDW!

CREATE TABLE DIST_INVENTORY_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
        BRANCH_KEY INTEGER NOT NULL,
        PRODUCT_KEY INTEGER NOT NULL,
        OPENING_INVENTORY INTEGER NOT NULL,
        QUANTITY_SHIPPED INTEGER NOT NULL,
        ADDITIONS INTEGER NOT NULL,
        UNIT_COST DECIMAL(19 , 2),
        CLOSING_INVENTORY INTEGER NOT NULL,
        AVERAGE_UNIT_COST DECIMAL(19 , 2)
    )
    DATA CAPTURE NONE
    IN GOSALES_TS!
```

Statement terminator: !

- [ ] Run DDL on server
- [ ] Open DDL file for editing

Figure 4-1 Physical model structure

4.2 Creating the physical data model

There are several different ways to create physical data models with Design Studio. For example, models can be created:

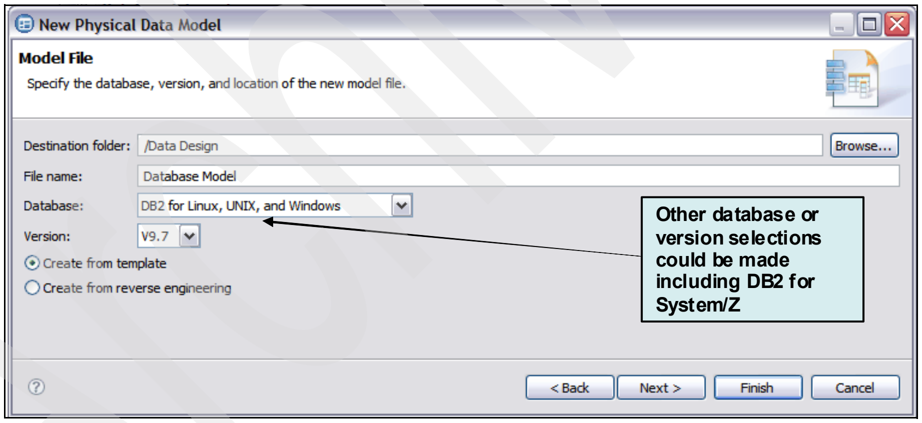
- From an empty template
- From a database using the Physical Model wizard
- From DDL using the Physical Model wizard
- Dragging and dropping from the Database Explorer view to the project in the Data Project Explorer View

4.2.1 Creating the physical data model from a template

You can create a new physical model from the empty template provided with Design Studio. To create a model this way perform the following steps:

1. Highlight the Data Models folder in the project, right-click, and choose **New** → **Physical Data Model**.

   This launches the New Physical Data Model wizard shown in Figure 4-2. This wizard can also be launched from the main Design Studio menu by selecting **File** → **New** → **Physical Data Model**.


New Physical Data Model

Model File

Specify the database, version, and location of the new model file.

Destination folder: /Data Design

File name: Database Model

Database: DB2 for Linux, UNIX, and Windows

Version: V9.7

(●) Create from template

( ) Create from reverse engineering

Other database or version selections could be made including DB2 for System/Z

Figure 4-2 Physical Data Model wizard

2. Select the **Create from template** radio button, as shown in Figure 4-2.

   Design Studio includes a modeling template called Empty Physical Model Template, which is shown on the next page of the wizard.

3. Choose the template and click **Finish**.

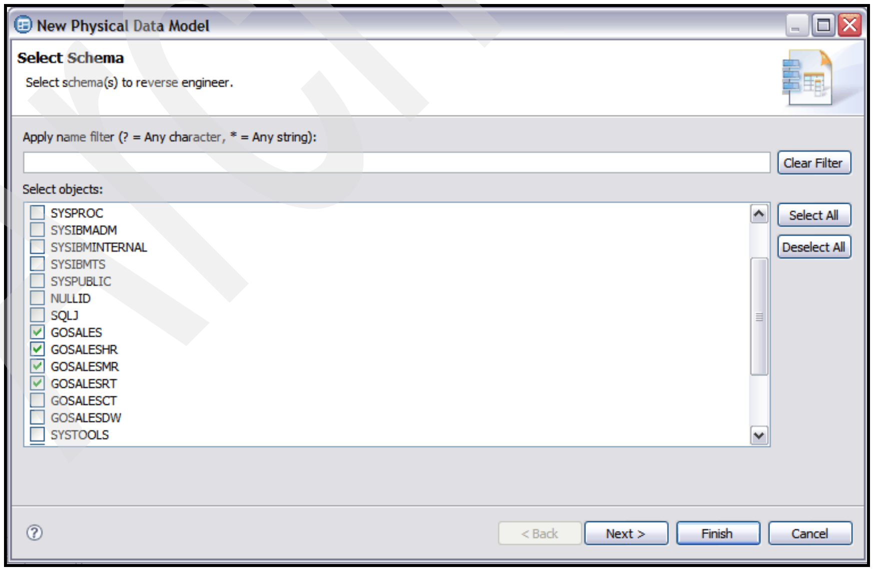
From the blank design template you can use the visual diagram editor with its palette to design the model or you can use the Data Project Explorer to add objects to the model. For more information see 4.3.2, “Using the Diagram Editor” on page 63.

4.2.2 Creating the physical data model from an existing database

Another approach to creating physical data models is to reverse engineer from a database connection or database definition. The steps to reverse engineer start out similarly to the template approach:

1. Launch the New Physical Model wizard by selecting **File** → **New** → **Physical Data Model** from the main menu or by right-clicking either the project or the Data Models folder in the Data Project Explorer view and choosing **New** → **Physical Data Model**.
2. Provide the details such as destination folder, model name, and database type and version.
3. Select the **Create from reverse engineering** radio button and click **Next**.
4. The next window provides the choice to reverse engineer from either a database or a DDL script. For purposes of this procedure, choose the database option.

   For instructions on reverse engineering from a DDL script, see 4.2.3, “Creating the physical data model from Data Definition Language (DDL)” on page 60.
5. Enter the requested database connection information. You can either use an existing connection or define a new one. After providing the database connection information, including user ID and password, you are presented with a list of schemas that can be reverse engineered.
6. Select a schema, as depicted in Figure 4-3.



New Physical Data Model

Select Schema

Select schema(s) to reverse engineer.

Apply name filter (? = Any character, * = Any string):

Select objects:

- [ ] SYSPROC
- [ ] SYSIBMADM
- [ ] SYSIBMINTERNAL
- [ ] SYSIBMTS
- [ ] SYSPUBLIC
- [ ] NULLID
- [ ] SQLJ
- [✓] GOSALES
- [✓] GOSALESHR
- [✓] GOSALESMR
- [✓] GOSALESRT
- [ ] GOSALESCT
- [ ] GOSALESDW
- [ ] SYSTOOLS

Figure 4-3 Selecting schemas

Note: If there is a filter defined as part of the database connection, the filter is applied every time the database connection is used. Therefore, this impacts the list of schemas that you see in the New Physical Data Model wizard. To verify whether filters are in use, review the objects associated with the database connection in the Database Explorer view. If the schema folder is labeled `Schemas[Filtered]`, then filters are enabled. To modify or review the filters, select the Schema folder in the Database Explorer, right-click, and choose **Filter**. You can then make any changes necessary.

As illustrated in Figure 4-4 the schema object list at this step in the wizard might also be filtered dynamically. This filter specifies the qualifiers to be included in the list.

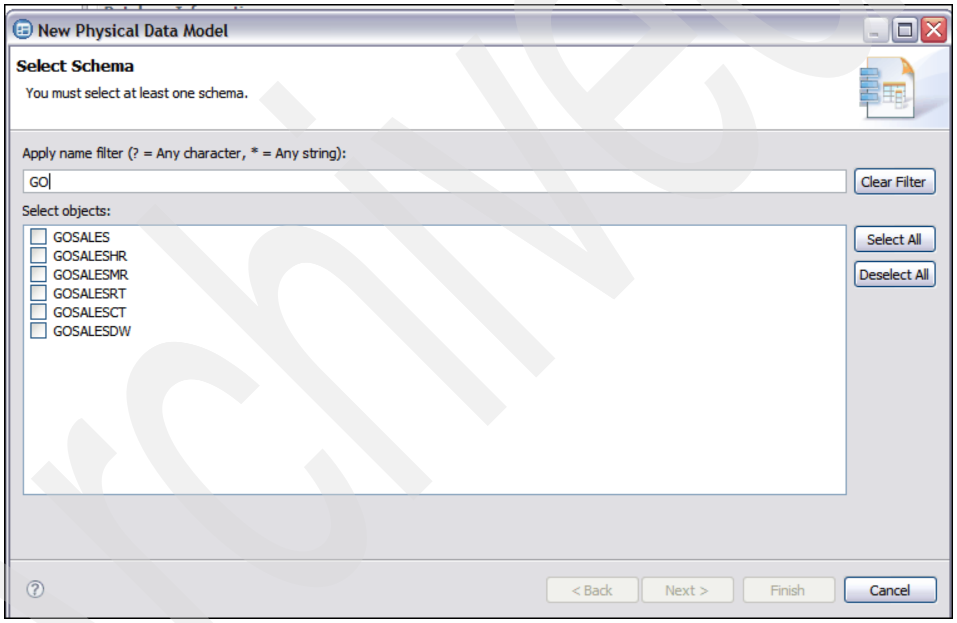


Figure 4-4 Filtering schemas

After selecting the schemas you want to use in the model, click **Next** to be presented with a list of the database elements you would like to include.

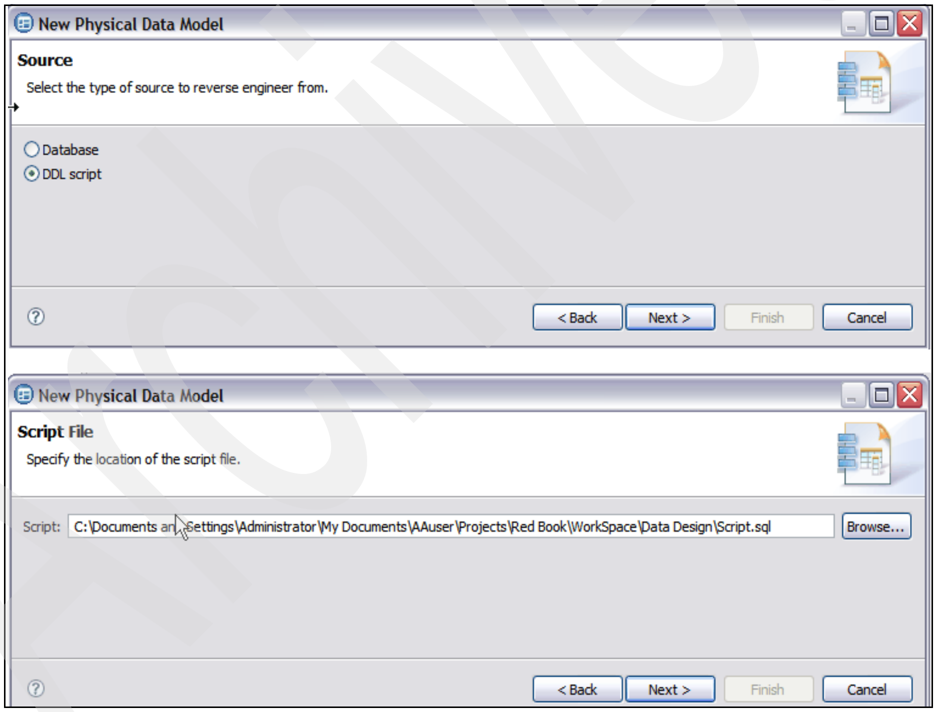
7. Check all database object types of interest and click **Next**.

   The final window of the wizard provides options to create an overview diagram and to infer implicit relationships. If you do not select these options, the components might be added to your model later.

4.2.3 Creating the physical data model from Data Definition Language (DDL)

Another of the options available in the Physical Model wizard is to reverse engineer from Database Definition Language (DDL). To do this, perform the following steps:

1. Launch the wizard by selecting **File** → **New** → **Physical Data Model** from the main menu. You can also right-click either your project or the Data Models folder in the Data Project Explorer view, and choose **New** → **Physical Data Model**.
2. Provide the details such as destination folder, model name, and database type and version.
3. Select the **Create from reverse engineering** radio button and click **Next**.
4. Choose the **DDL script** radio button as the source (Figure 4-5), and identify the location of the DDL file at the next prompt.



New Physical Data Model

Source

Select the type of source to reverse engineer from.

( ) Database
(●) DDL script

New Physical Data Model

Script File

Specify the location of the script file.

Script: C:\Documents an Settings\Administrator\My Documents\AAuser\Projects\Red Book\WorkSpace\Data Design\Script.sql

Figure 4-5 Reverse engineering from existing DDLto create physical data model

Similar to the steps to reverse engineer from a database connection, on the option window you can choose to have an overview diagram created and implicit relationships inferred. If you do not choose these options at this time, those components might be created later from the Data Project Explorer.

4.2.4 Creating the physical data model from the Database Explorer

Another method for creating a new physical model from an existing database is to drag the objects from the Database Explorer to the Data Project Explorer. To use this method, perform the following steps:

1. Verify that there is an active connection to the source database in the Database Explorer view.
2. Expand the database connection and select the objects to be reverse engineered. The level of granularity available ranges from database to table. Drag the selected objects to the Data Project Explorer and onto an existing project, as shown in Figure 4-6. The resulting physical model name is the name of the database connection. A number is added to the end of the model name for uniqueness, if necessary.

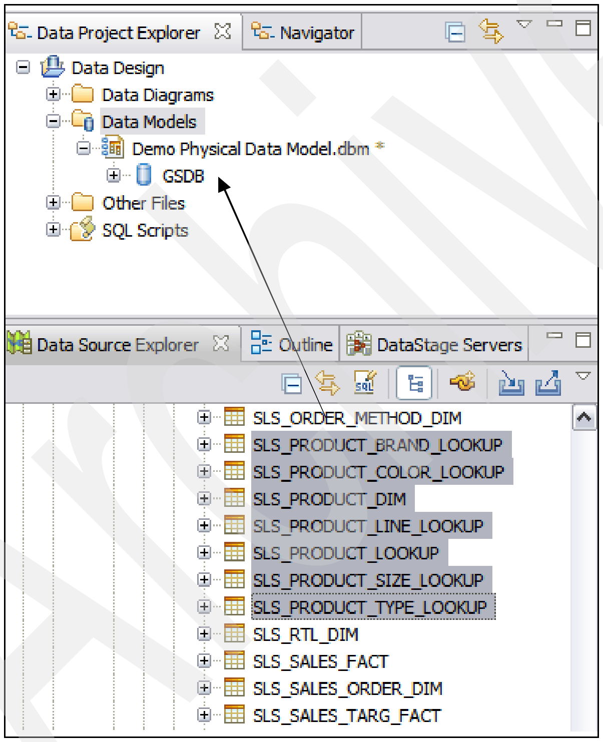


Figure 4-6 Dragging from the Database Explorer to create physical data model

4.3 Working with diagrams

Design Studio uses entity relationship diagrams to provide a visual representation of the physical data models in your projects. Entity relationship diagrams are a useful mechanism for understanding the data elements in your projects and communicating that information to others. in Design Studio, you can have multiple diagrams per schema in the projects. It is often helpful to organize the data models into multiple subject areas, especially when these are large and complex.

In addition to the value that the diagrams bring to the projects by simplifying and improving the understanding of complex data environments, the diagrams can also be used to edit and modify the physical data model.

4.3.1 Creating a diagram

Design Studio provides several ways to add entity relationship diagrams to projects.

If you have chosen to reverse engineer the data model, you can have the wizard create an overview diagram for you. The wizard provides prompts that allow you to specify what elements you want to have included in the diagram.

You can still use diagrams in the data projects, even if you do not create the data models through the reverse engineering approach. New diagrams might be created from any Diagrams folder in the Project Explorer. Select the Diagrams folder, right-click, and choose **New Overview Diagram**. You are prompted to select which elements from the current schema you want to include in the diagram.

You can also create a blank diagram, rather than including existing schema elements. To create a blank diagram right-click the Diagrams folder in the Project Explorer and choose **New Blank Diagram**.

4.3.2 Using the Diagram Editor

An overall view of the Diagram Editor is shown in Figure 4-7. The two main components to the Diagram Editor are the drawing area, or canvas, and the palette. Elements might be added to the diagram from either the palette or the Data Project Explorer. To use the Data Project Explorer, drag elements from the Data Models folder onto the diagram canvas.

The shapes representing tables and elements can be moved to make them more readable by highlighting either a rectangle or a line and dragging it to a new position.

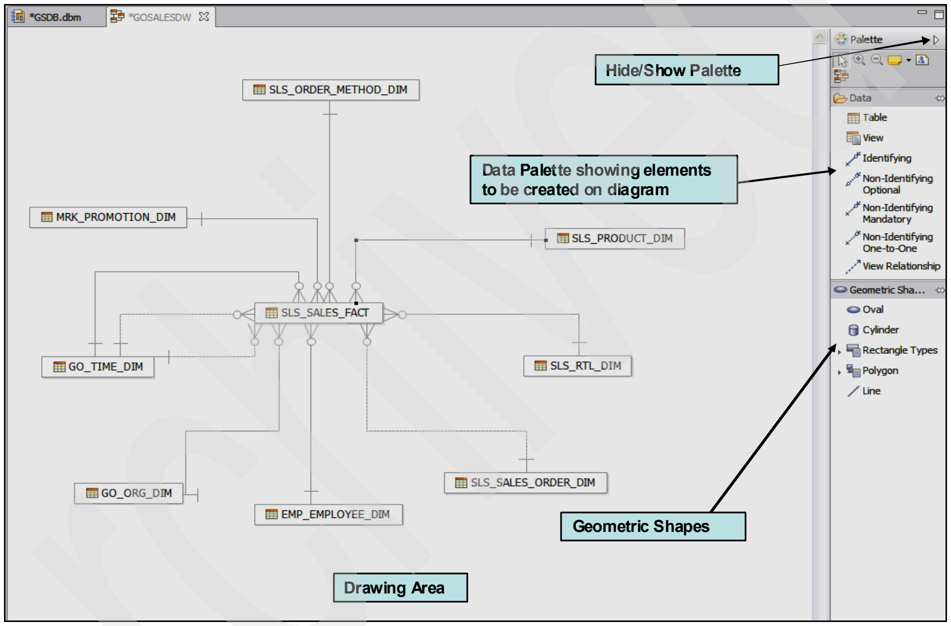


Figure 4-7 Diagram Editor view

Items from the palette might be placed on the drawing area by clicking an element to select it and moving the mouse to the drawing area and clicking. Elements that are added to the diagram in this manner are provided with a default name, but you can change the name to something more meaningful, as in Figure 4-8. When elements are added to the canvas, as in Figure 4-8, they are also added to the Data Project Explorer.

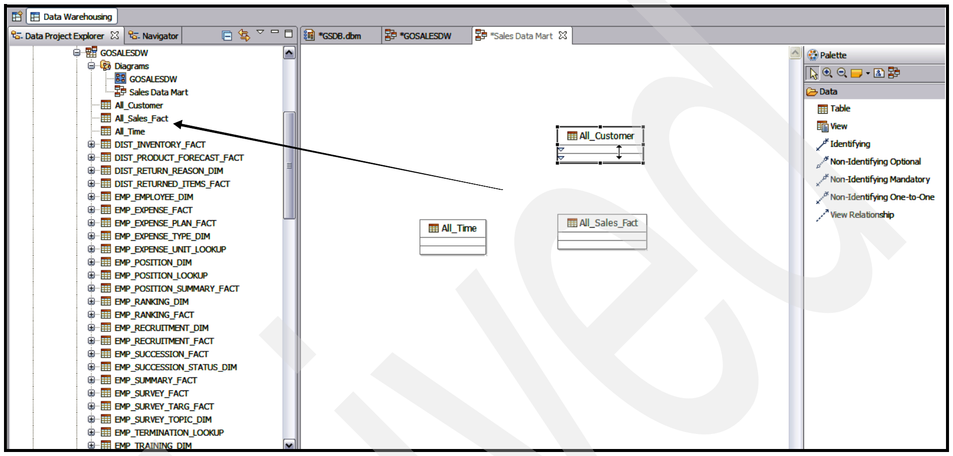


Figure 4-8 Adding palette elements to the diagram

Once the diagram contains tables, then columns might be added from the visual diagram. When you select a model element, action bars appear, providing context-aware pop-ups. Options available through the action bar include the ability to add a new key, column, index, or trigger. The palette can be used to establish identifying and non-identifying relationships between the various tables that make up each diagram. As you are using this approach, you can also use the Properties view to further define the objects that you are creating.

4.4 Editing physical data models

Design Studio provides two approaches to editing physical data models. Recall that the physical data models that you create are visually implemented as an entity relationship diagram. After you have created a diagram, you can make changes to the physical model by editing the diagram. See 4.3.2, “Using the Diagram Editor” on page 63 for more details about this approach. Another approach is to use the options available in the Data Project Explorer to modify or edit the physical model.

4.4.1 Using the Data Project Explorer

The Data Project Explorer provides two ways to modify the physical data models:

- Data model objects that have already been defined might be dragged onto the diagram canvas.
- Object properties might also be modified in the Data Project Explorer by selecting them and using the Properties view to modify various settings.

You can also use the Data Project Explorer to create new objects in the models, using the following steps:

1. Select the database in the Data Project Explorer, right-click, select **Add Data Object**, then schema, bufferpool, tablespaces, or dependencies.
2. Select the schema in the Data Project Explorer, right-click, select **Add Data Object**, then table, view, function, or stored procedure.
3. Select the table in the Data Project Explorer, right-click, select **Add Data Object**, then column check constraint, unique constraint, foreign key, index, trigger, or dependency.

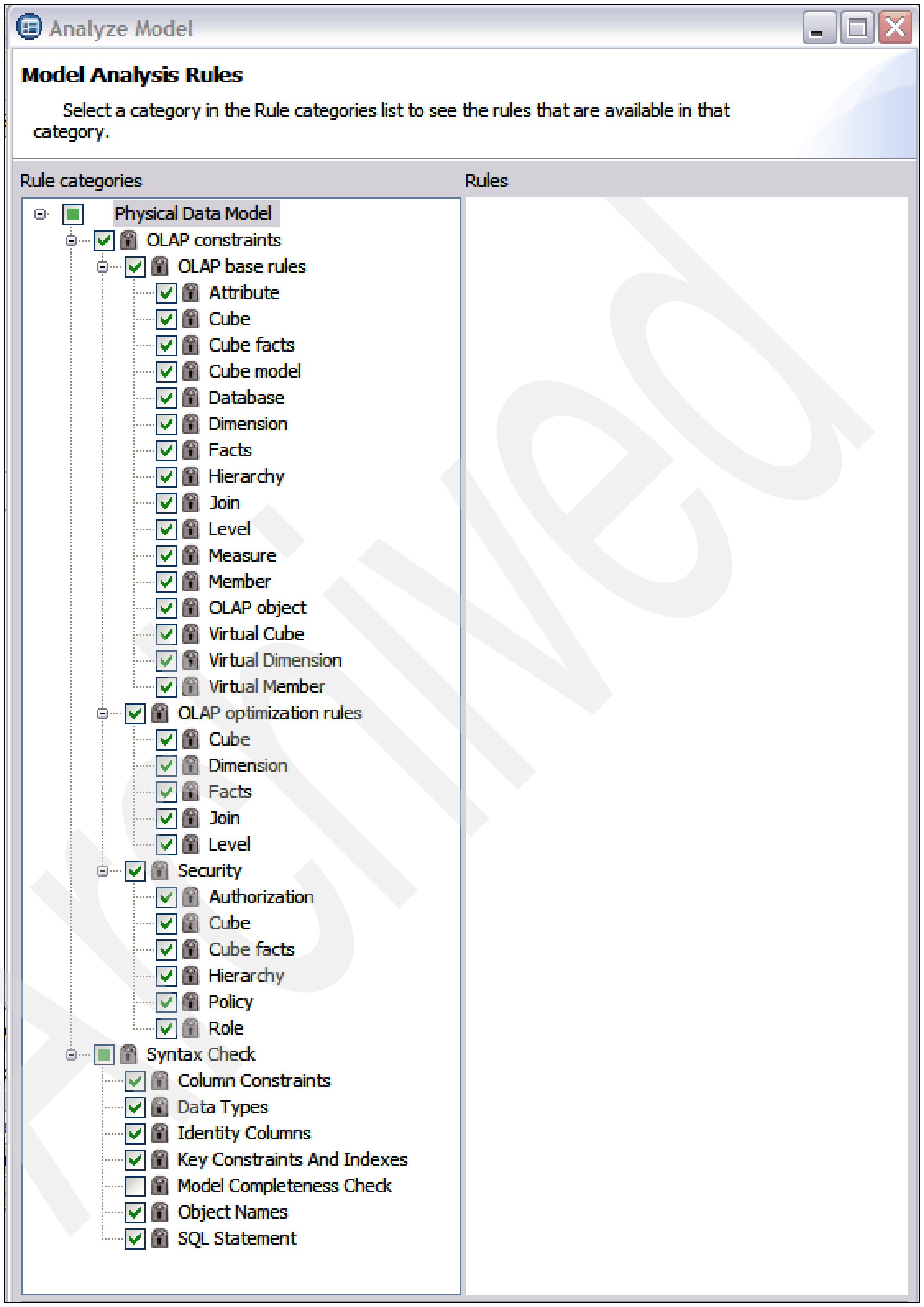
4.5 Model analysis

Design Studio includes a rule-based model analysis tool that allows you to evaluate adherence of the data model to design and optimization best practices. The model analysis utility provides a mechanism to improve the integrity and quality of your physical models, and helps to assure that the physical model is valid. Validate physical data models before they are deployed to be sure that there are not any problems in the design.

Model analysis might be performed against databases or schemas. To perform model analysis, perform the following steps:

1. Launch the wizard
2. Select the database or schema you are interested in analyzing, right-click, and choose **Analyze Model**.

   Models might be analyzed against the categories in Figure 4-9 on page 66. in each of these categories, there are several rules that you can choose to include in the analysis, such as key constraints, object naming standards, and several rules related to OLAP best practices.
3. Select the set of rules that are of interest and click **Finish** to start the analysis.



Analyze Model

Model Analysis Rules

Select a category in the Rule categories list to see the rules that are available in that category.

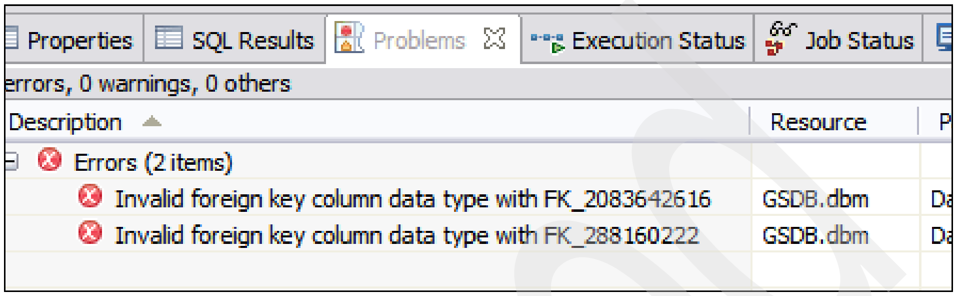
Rule categories

Rules

- Physical Data Model
  - ✓ OLAP constraints
    - ✓ OLAP base rules
      - ✓ Attribute
      - ✓ Cube
      - ✓ Cube facts
      - ✓ Cube model
      - ✓ Database
      - ✓ Dimension
      - ✓ Facts
      - ✓ Hierarchy
      - ✓ Join
      - ✓ Level
      - ✓ Measure
      - ✓ Member
      - ✓ OLAP object
      - ✓ Virtual Cube
      - ✓ Virtual Dimension
      - ✓ Virtual Member
    - ✓ OLAP optimization rules
      - ✓ Cube
      - ✓ Dimension
      - ✓ Facts
      - ✓ Join
      - ✓ Level
    - ✓ Security
      - ✓ Authorization
      - ✓ Cube
      - ✓ Cube facts
      - ✓ Hierarchy
      - ✓ Policy
      - ✓ Role
  - Syntax Check
    - ✓ Column Constraints
    - ✓ Data Types
    - ✓ Identity Columns
    - ✓ Key Constraints And Indexes
    - ☐ Model Completeness Check
    - ✓ Object Names
    - ✓ SQL Statement

Figure 4-9 Model Analysis

The results of the Model Analysis utility are displayed in the Problems view. An example of the output is shown in Figure 4-10. Double-click an item in the Problems view to navigate to the relevant model objects in the Project Explorer. Take corrective action and re-run the model analysis to verify that all problems were corrected.



errors, 0 warnings, 0 others

Description	Resource	P
Errors (2 items)		
Invalid foreign key column data type with FK_2083642616	GSDB.dbm	Da
Invalid foreign key column data type with FK_288160222	GSDB.dbm	Da

Figure 4-10 Model analysis output

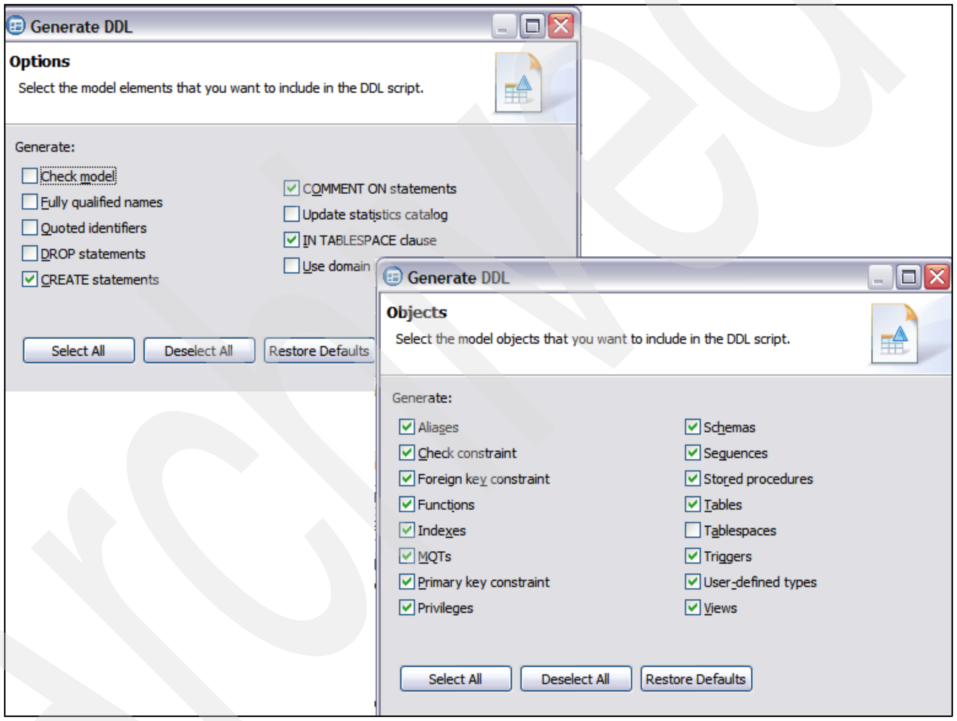
Correct any issues that the model analysis identifies until all errors are cleared. After the model has been successfully validated, you are ready to deploy it.

4.6 Deploying the data model

After the physical model has been designed and validated, you are ready to deploy the model into your environment. The deployment process results in the schemas, tables, and other objects that were modeled being created in the target database. After the physical model has been deployed, you can begin populating the structures with data.

4.6.1 Using Design Studio to deploy a physical data model

Physical data models might be deployed from Design Studio through the Generate DDL wizard, which generates context-driven DDL. When you generate the DDL code, you can choose a database, schema, user, table, bufferpool, or table space as the root for the code generation. To launch the DDL wizard, select the desired data element, right-click, and choose **Generate DDL**. Alternatively, to launch the wizard from the menu, with the data element selected, click **Data** → **Generate DDL**. Respond through the prompts of the wizard, indicating what model elements and objects you would like to include in the DDL script. See Figure 4-11 for an example of the Generate DDL wizard.



Options

Select the model elements that you want to include in the DDL script.

Generate:

- [ ] Check model
- [ ] Fully qualified names
- [ ] Quoted identifiers
- [ ] DROP statements
- [✓] CREATE statements
- [✓] COMMENT ON statements
- [ ] Update statistics catalog
- [✓] IN TABLESPACE clause
- [ ] Use domain

Objects

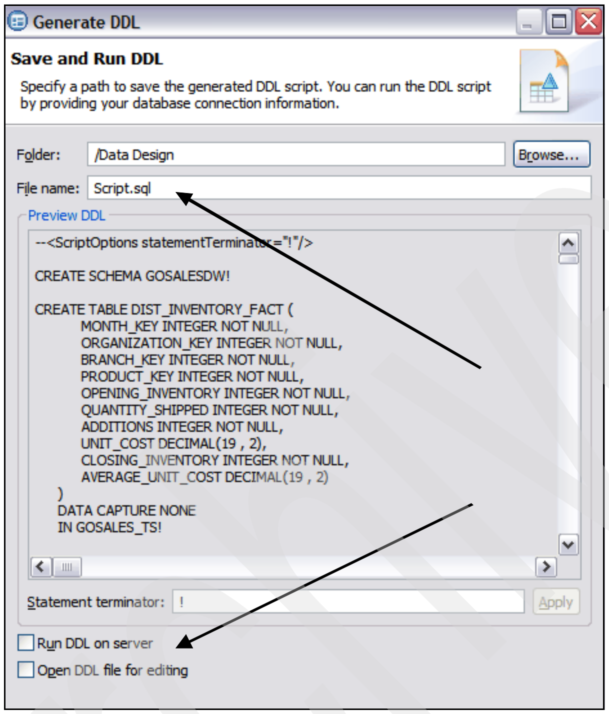
Select the model objects that you want to include in the DDL script.

Generate:

- [✓] Aliases
- [✓] Check constraint
- [✓] Foreign key constraint
- [✓] Functions
- [✓] Indexes
- [✓] MQTs
- [✓] Primary key constraint
- [✓] Privileges
- [✓] Schemas
- [✓] Sequences
- [✓] Stored procedures
- [✓] Tables
- [ ] Tablespaces
- [✓] Triggers
- [✓] User-defined types
- [✓] Views

Figure 4-11 Generating DDL

Once you have selected the objects you want to deploy, you will be presented with a summary of the DDL script, with options to save the DDL file or execute the DDL script on the server. These options are shown in Figure 4-12. If you choose to run the DDL on the server, you are prompted to select a database connection or create a new database connection. The DDL script is saved in the SQL Scripts logical folder in your data design project.


Generate DDL
Save and Run DDL
Specify a path to save the generated DDL script. You can run the DDL script by providing your database connection information.
Folder:
Browse...
File name:
Preview DDL
Statement terminator:
Apply
Run DDL on server
Open DDL file for editing

Folder: /Data Design

File name: Script.sql

```
--<ScriptOptions statementTerminator="!"/>

CREATE SCHEMA GOSALESDW!

CREATE TABLE DIST_INVENTORY_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
        BRANCH_KEY INTEGER NOT NULL,
        PRODUCT_KEY INTEGER NOT NULL,
        OPENING_INVENTORY INTEGER NOT NULL,
        QUANTITY_SHIPPED INTEGER NOT NULL,
        ADDITIONS INTEGER NOT NULL,
        UNIT_COST DECIMAL(19 , 2),
        CLOSING_INVENTORY INTEGER NOT NULL,
        AVERAGE_UNIT_COST DECIMAL(19 , 2)
    )
    DATA CAPTURE NONE
    IN GOSALES_TS!
```

Statement terminator: !

Figure 4-12 Options for saving and running DDL

The DDL scripts that are located in the SQL Scripts folder might be saved for later execution, and they might also be modified before being executed. To edit the DDL, select the file in the SQL Scripts folder, right-click, and choose **Open With → SQL Editor**. This causes the file to be opened with a text editor. When you are ready to run the DDL, select the file, right-click, and choose **Run SQL**. You can review the results in the Data Output view, which includes status information, messages, parameters, and results.

4.6.2 Using the administration console to deploy a physical model

Restriction: This method of deploying a physical model is only available for DB2 for Linux, UNIX and Windows databases.

The InfoSphere Warehouse Administration Console might also be used to deploy physical models. For more information about using this approach to model deployment see 9.2.2, “Deployment using the administration console” on page 389.

4.7 Maintaining model accuracy

It is important to manage changes to the models that might happen over time. As business requirements change, there is often an adjustment that needs to be made to the physical data models. Design Studio has the ability to help you manage that type of change. It is helpful to compare the differences between two physical data models, and also compare the difference between a physical data model and the database where it was deployed. After changes are identified, it is important to synchronize the two models being compared. It is also important to analyze the impact of a change to the model before the change is implemented.

4.7.1 Comparing objects in the physical data model

Design Studio provides object-based compare and synchronization capabilities as a mechanism to assist and simplify the task of maintaining model accuracy. From the Database Explorer, you can select a database object, right-click, and choose **Compare With** → **Another Data Object**. You are prompted to select the object to use for comparison. Another way to achieve the same comparison is to highlight two data objects, right-click, and choose **Compare With** → **Each Other**. In the Project Explorer, there is a third comparison available if the source of the model was created by reverse engineering, If that is the case, select an object, right-click, and choose **Compare** → **Original Source**. This option is only available from the Project Explorer, not the Database Explorer.

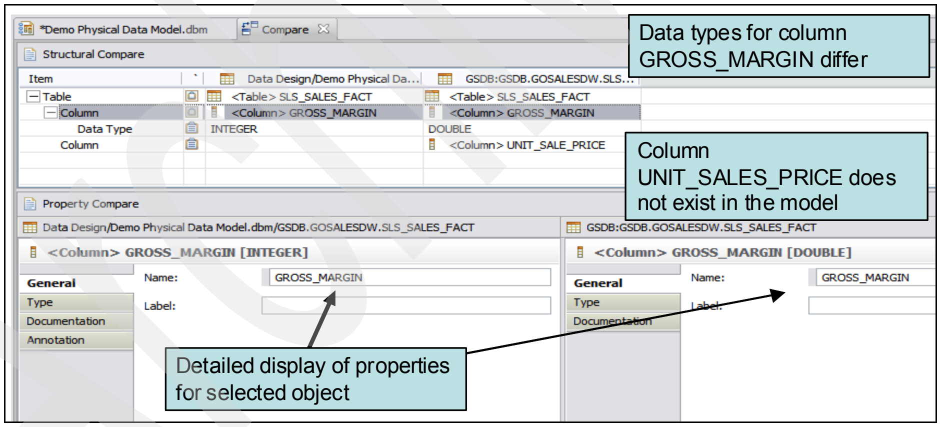
These options under the **Compare With** tool are helpful if you need to compare physical objects with each other (for example, to compare objects from a test database to objects in a production database). Or you can compare the baseline database objects from the Database Explorer with changed objects from your project before deploying changes.

4.7.2 Visualizing differences between objects

The results of the object comparison are displayed in two views, which are connected. The upper view is called *Structural Compare*. The Structural Compare shows the differences in a tree format. The first column is a tree. Each entry in the tree represents a part of the model where a difference was found. The second column in the Structural Compare output shows the first object as input to the Compare tool, and the third column shows the second object to which it was being compared. A copy of the differences might be saved by clicking **Export** in the upper right portion of the Structural Compare view. The file that results from this option is an XML file.

If two different items that appear on different rows in the comparison output need to be compared to each other, you can use the **Pair** and **Separate** buttons to facilitate their comparison and align the comparison results. This is helpful when the column names are different, but the objects represent the same data.

As you work down through the tree in the Structural Compare, differences are highlighted in the lower view, which is called the *Property Compare*. This shows the Properties view. An example of the output from the Compare Editor is shown in Figure 4-13.


*Demo Physical Data Model.dbm
Compare
Structural Compare
Data types for column GROSS_MARGIN differ
Column UNIT_SALES_PRICE does not exist in the model
Property Compare
General
Type
Documentation
Annotation
Detailed display of properties for selected object

Item	`	Data Design/Demo Physical Da…	GSDB:GSDB.GOSALESDW.SLS…
Table		<Table> SLS_SALES_FACT	<Table> SLS_SALES_FACT
Column		<Column> GROSS_MARGIN	<Column> GROSS_MARGIN
Data Type		INTEGER	DOUBLE
Column			<Column> UNIT_SALE_PRICE

Data Design/Demo Physical Data Model.dbm/GSDB.GOSALESDW.SLS_SALES_FACT

<Column> GROSS_MARGIN [INTEGER]

Name: GROSS_MARGIN

Label:

GSDB:GSDB.GOSALESDW.SLS_SALES_FACT

<Column> GROSS_MARGIN [DOUBLE]

Name: GROSS_MARGIN

Label:

Figure 4-13 The Compare Editor

4.7.3 Synchronization of differences

As you view the differences between the objects you have compared, you can use the **Copy From** buttons, which are located on the right side of the toolbar that separates the Structural Compare and Property Compare views. These buttons allow you to implement changes from one model or object to another. You can copy changes from left to right, or from right to left. As you use the **Copy From** buttons, the information displayed in the Structural Compare is updated to reflect the change.

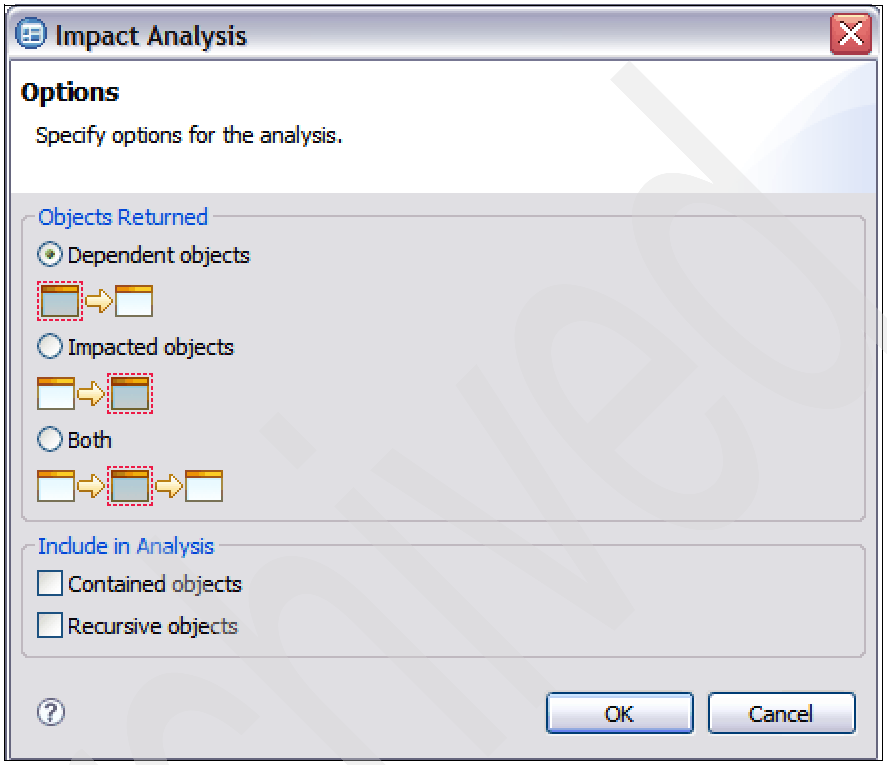
Another way to implement changes to bring your models in synchronization with each other is to edit the values that are displayed in the Property Compare view. Any changes that are made might be undone or redone by clicking **Edit** → **Undo** and **Edit** → **Redo**.

Once you have reviewed all of the differences and have decided what needs to be done to bring your models in sync with each other, you are ready to generate delta DDL. This DDL can be saved for later execution or might be executed directly from Design Studio.

4.7.4 Impact analysis

Design Studio also provides a mechanism for performing impact analysis. It is beneficial to understand the implications of changes to models before they are implemented. The impact analysis utility shows all of the dependencies for the selected object. The results are visually displayed, with a dependency diagram and a Model Report view added to the Output pane of Design Studio.

The impact analysis discovery can be run selectively. To launch the utility, highlight an object, right-click, and choose **Analyze Impact**. Choose the appropriate options, as shown in Figure 4-14.


Impact Analysis

Options

Specify options for the analysis.

Objects Returned

[✓ Dependent objects]

Impacted objects

Both

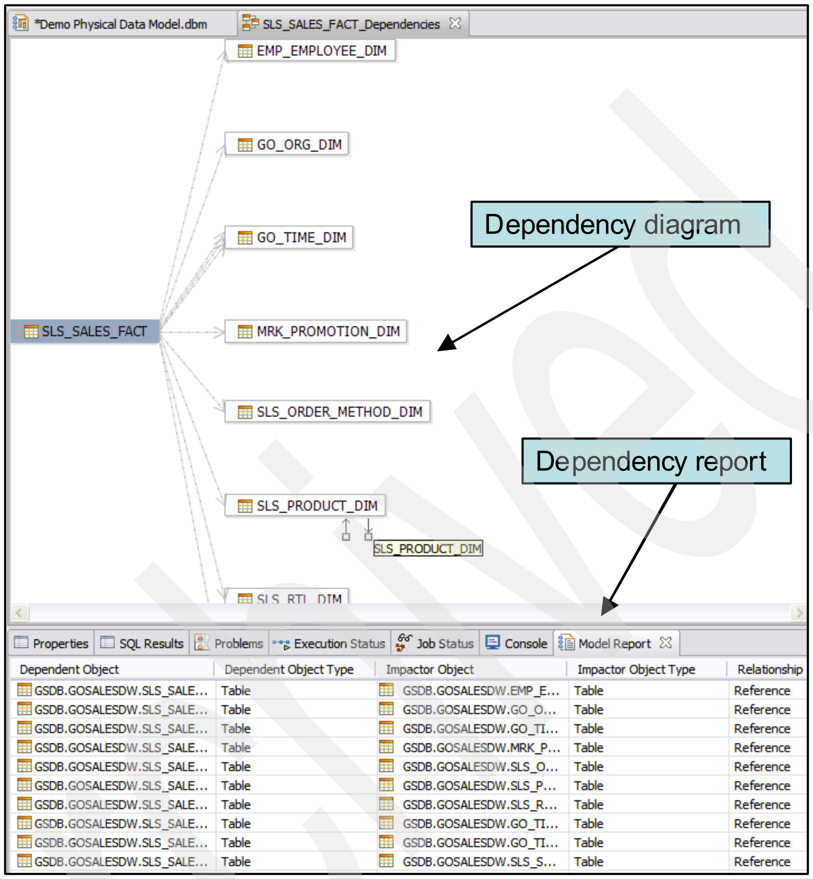
Include in Analysis

☐ Contained objects

☐ Recursive objects

Figure 4-14 Impact Analysis Options

The results of the impact analysis are shown in a dependency diagram, as shown in Figure 4-15.



Dependent Object	Dependent Object Type	Impactor Object	Impactor Object Type	Relationship
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.EMP_E...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.GO_O...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.GO_TI...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.MRK_P...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.SLS_O...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.SLS_P...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.SLS_R...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.GO_TI...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.GO_TI...	Table	Reference
GSDB.GOSALESDW.SLS_SALE...	Table	GSDB.GOSALESDW.SLS_S...	Table	Reference

Figure 4-15 Impact Analysis Report

4.8 Summary

In this chapter we have shown how to create and edit physical models. We have also explored common practices around model validation and analysis. After these steps have been concluded, you will have deployed your physical model to the database target. You are now ready to begin populating those structures with data. These procedures are outlined in Chapter 5, “Data movement and transformation” on page 75.

5

Data movement and transformation

In this chapter we discuss the data movement and transformation capabilities of InfoSphere Warehouse. The SQL Warehousing Tool (SQW) is the component that provides these services.

The information in this chapter gives you an understanding of the development and runtime architectures. It shows how to develop data flows and control flows, how to promote from development to test or production, and how to execute and monitor in the runtime environment.

5.1 SQW overview

The basic function of SQW is to manage and move data into and around the data warehouse while transforming it for various purposes. SQW provides these services by making use of the power of the DB2 relational database engine and the SQL language, which classifies SQW in the ELT (Extract-Load-Transform) category of data movement and transformation tools. SQW also provides sequencing and flow control functions and functions to integrate non-database processing.

There are two versions of InfoSphere Warehouse, one for the distributed platform LUW (Linux, UNIX and Windows) and one for the System z platform. While the basics of SQW are the same across the platforms, there are inherent differences due to the target databases, the target platform, and the timing of releases between the two products. In this chapter we discuss SQW in a general way, but point out where there are differences in the two platforms and provide examples from both.

5.1.1 Architecture

SQW consists of separate tooling for the development environment and the runtime environment. See Figure 5-1 on page 77. The development tooling is the InfoSphere Warehouse Design Studio. This is where the data movement and transformation routines are developed and tested. After successfully testing in the development environment against development databases, the routines are deployed to the runtime environment where they can be scheduled for execution using the production databases. The Web-based administration console is used to manage the runtime environment, including deployment, scheduling, and execution, and monitoring functions.

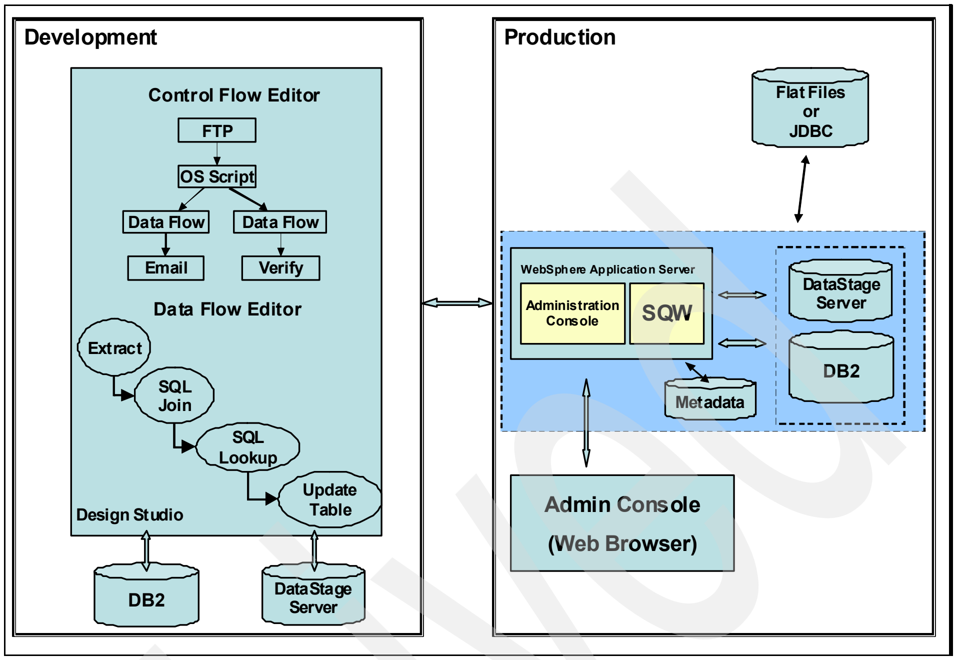


Figure 5-1 SQW development and runtime architecture

5.1.2 Development environment

The development tool, as seen in Figure 5-1, is an Eclipse-based member of the IBM Data Studio family of products called the InfoSphere Warehouse Design Studio. See Chapter 5, "Data movement and transformation" on page 75 for more information about the InfoSphere Design Studio.

A physical data model of the source and target relational databases are required to develop data movement routines. The physical data model provides metadata about the structure of the source and target tables. See Chapter 3, "InfoSphere Warehouse Design Studio" on page 29 for more information about developing the physical data model.

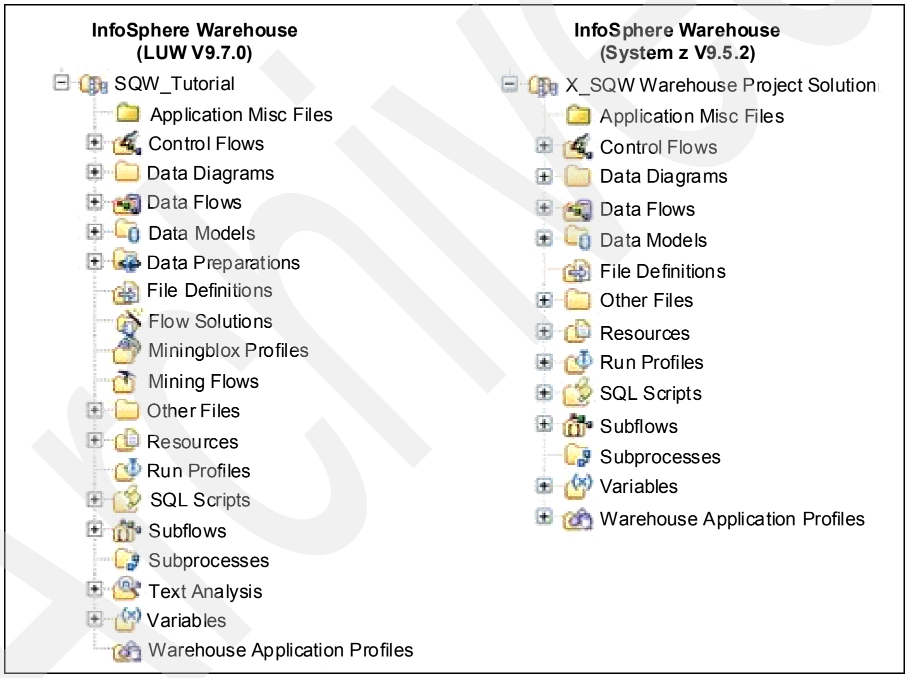
Data warehouse project

SQW routines, called flows, are organized into a data warehouse project. The project contains all of the developments artifacts for the project. There are also references to one or more data projects that contain the physical data models needed for the project.

Tip: The physical data model can actually be contained in the data warehouse project. However, this limits its use to the data warehouse project only. Using a data project for the physical data model allows the data model to be referenced by multiple data warehouse projects.

Figure 5-2 shows the structure of a data warehouse project. The project is organized into folders that contain the various types of development artifacts. The following folders are used most, as these contain the primary development artifacts:

- 1) Data Flows
- 2) Control Flows
- 3) Subprocesses
- 4) Subflows



InfoSphere Warehouse (LUW V9.7.0)

```
SQW_Tutorial
    Application Misc Files
    Control Flows
    Data Diagrams
    Data Flows
    Data Models
    Data Preparations
    File Definitions
    Flow Solutions
    Miningblox Profiles
    Mining Flows
    Other Files
    Resources
    Run Profiles
    SQL Scripts
    Subflows
    Subprocesses
    Text Analysis
    Variables
    Warehouse Application Profiles
```

InfoSphere Warehouse (System z V9.5.2)

```
X_SQW Warehouse Project Solution
    Application Misc Files
    Control Flows
    Data Diagrams
    Data Flows
    Data Models
    File Definitions
    Other Files
    Resources
    Run Profiles
    SQL Scripts
    Subflows
    Subprocesses
    Variables
    Warehouse Application Profiles
```

Figure 5-2 Data warehouse project folders

Data flows and subflows

Data flows model the SQL-based data movement and transformation activities that are executed by the DB2 database engine. A data flow consists of activities represented by graphical operators that extract data from flat files or relational tables, transform the data, and load it into a relational table in a data warehouse, data mart, or staging area. Data can also be exported to flat files. A sample data flow is shown in Figure 5-3.

Design Studio provides a graphical data flow editor with an intuitive way to visualize and design data flows. Graphical operators model the various steps of a data flow activity. By arranging these source, transform, and target operators in a canvas work area, connecting them, and defining their properties, models can be created that meet the business requirements. After creating data flows, SQL code is generated which perform the specific DB2 SQL operations needed to complete the operations defined by the data flow model.

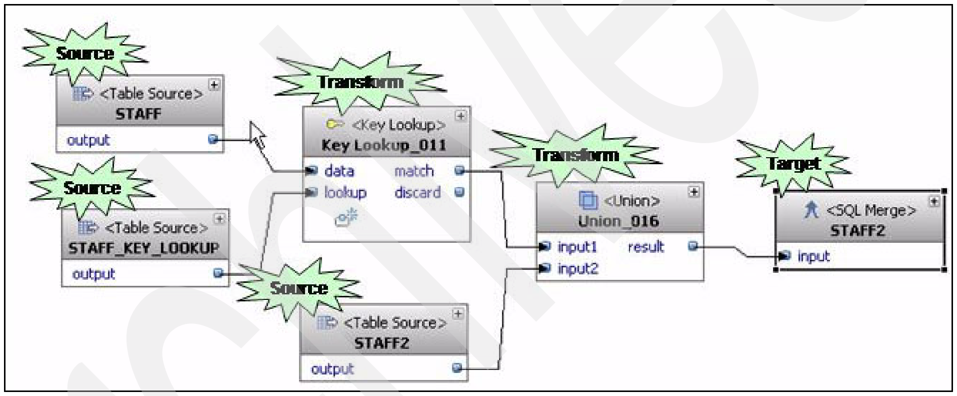


Figure 5-3 A simple data flow

Subflows are similar to data flows but are used like macros or subroutines. Subflows are included in a data flow by using a subflow operator. Subflows have special input and output connections that pass the data into and out of the subflow. Subflows can be used to define common data flow components that might be used in multiple data flows, or just to break a complex data flow into multiple logical sets of operations (similar to using structured techniques in a programming language). Figure 5-4 shows several subflows and their corresponding subflow operators.

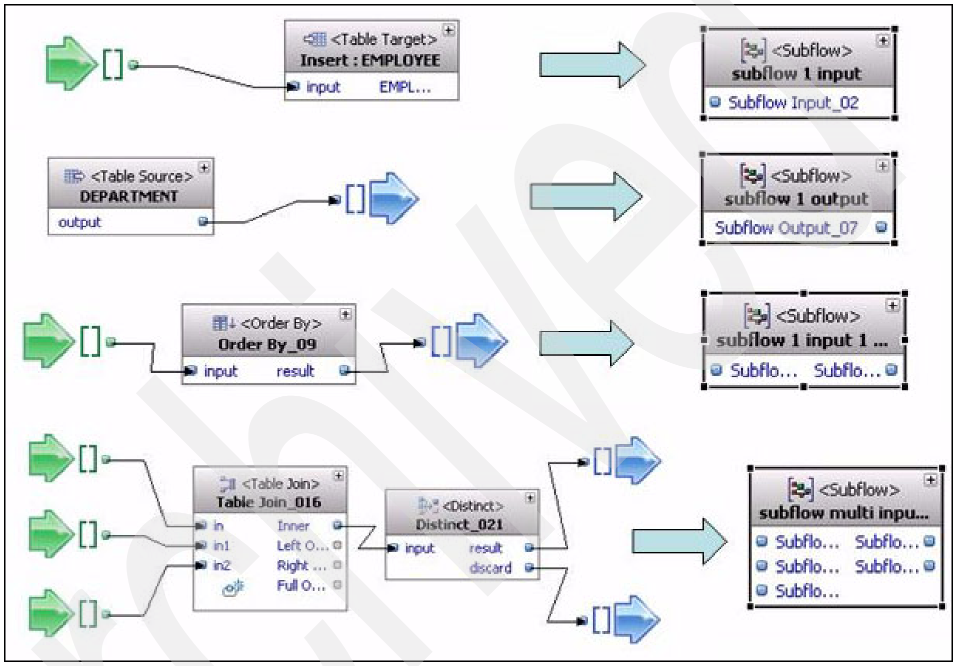


Figure 5-4 Subflows

Control flow and subprocesses

A control flow sequences and manages the flow of activities. Activities are independent units-of-work, and could be, as examples, a data flow, a database utility, an operating system script, or a stored procedure. A control flow is the unit of execution in the runtime environment.

Design Studio provides a graphical editor with an intuitive capability to visualize and design control flows. Graphical operators model various SQW and data processing activities. By arranging these operators in a canvas work area, connecting them, and defining their properties you can create work flow models that define the sequence of execution of the activities.

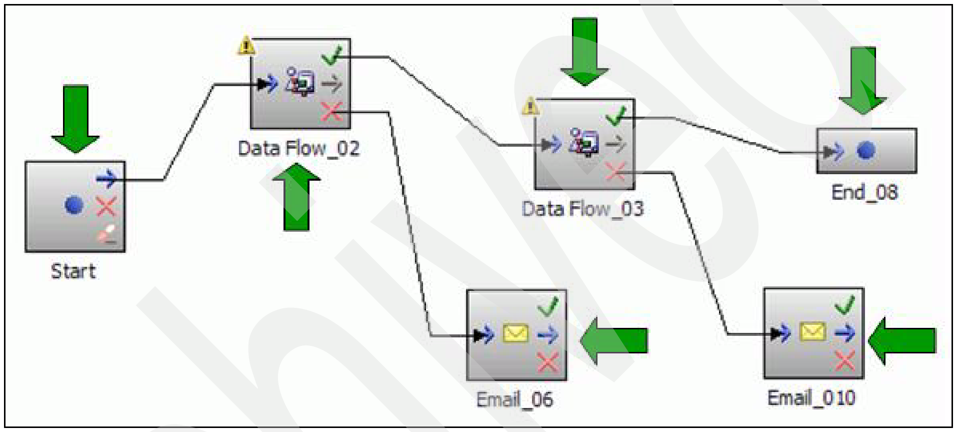


Figure 5-5 A simple control flow

A subprocess is similar to a control flow but is used like a macro or subroutine. Subprocesses can be used to define common control flow components that might be used in multiple control flows or just to break a complex control flow into multiple logical sets of operations similar to using structured techniques in a programming language.

Testing and debugging

Data flow and control flows can be tested and debugged without leaving the Design Studio development environment and without deploying the flows to the runtime environment. Live connections to the source and target databases are required. When testing from Design Studio, your local client machine acts similarly to the SQW runtime service in that it becomes the control point that manages the execution of the flows. There is no scheduling capability in the development environment.

Data flows can be tested and debugged individually as they are developed. Each control flow can also be tested and debugged individually as well. There is a specific debugger for data flows and for control flows.

Deployment

Deployment is the process of promoting a set of control flows from development to a runtime environment. Deployment is accomplished in two independent steps.

1. Define a warehouse application and select the set of control flows that will be deployed as a unit. This generates the code and packages up all of the necessary files and create a deployment package as a zip file. This is accomplished using Design Studio, typically by the developer.
2. Use this zip file to install the SQW application into the runtime environment using the Web-based administration console. This is done by the administrator of the runtime environment (test, QA or production). As part of the process, the administrator defines connections to the source and target databases, installs the SQW application, maps the application to the source and targets connections and, optionally, sets values for variables.

Once the SQW application is deployed, it can be executed manually or through a schedule, and monitored.

5.1.3 Runtime environment

In an IT shop there would be a runtime environment that represent a test, quality assurance or production environment for each runtime system. For each runtime environment, you need perform the following steps:

1. Install SQW applications into the environment.
2. Define execution schedules for control flows.
3. Execute control flows automatically.
4. Monitor the execution and completion of the control flows.

There are functions in the Web-based administration console that provide the user interface for administrators to manage the installation, define the schedules, and monitor the execution.

In addition to the user interface, there is a component to manage the SQW runtime. It is installed as part of the WebSphere Application Server-based InfoSphere administration console. When it is time to execute a control flow, whether invoked manually or through a schedule, it is the SQW runtime server that will manage the execution of each activity, in the proper sequence and on the appropriate database or system.

5.2 Developing data flows

Data flows were introduced in "Data flows and subflows" on page 79. A data flow is essentially the set of operations that performs the following tasks:

- Bring data into the data warehouse
- Move and transform the data in the data warehouse
- Update target tables

Data flows and subflows are developed using the Design Studio graphical data flow editor to create a flow model representing the data movement and transformations of the data. From this model, Design Studio generates optimized SQL to perform the actions specified in the flow model. The SQL is executed at the transformation DB2 database selected as the execution database.

In the next section we demonstrate how to develop a data flow and test the execution in Design Studio.

5.2.1 Data flow basics

When developing a data flow, you are actually developing a flow model. As a data model is a representation of a database, a data flow is a representation of what is to happen to the data as it flows into and in the execution database from source to target. Data flow operators define source data, transformations, and target tables. Operators are placed on the data flow editor canvas and connected to other operators using connectors. Operator properties define the details of the operator behavior.

Operators, connectors and ports

In general, there are three categories of operators:

- Sourcing data operators
- Transforming data operators
- Data targets operators

Figure 5-6 shows the operators in a simple data flow. There are a total of three operators that represent sources of data from relational tables. These are Table Source operators for the STAFF table, STAFF_KEY_LOOKUP table and the STAFF2 table. These table sources are connected to transform operators that perform actions on the data, a Key Lookup and a Union. The Union transform operator is connected to a target operator that does an SQL Merge of the incoming data into the STAFF2 table.

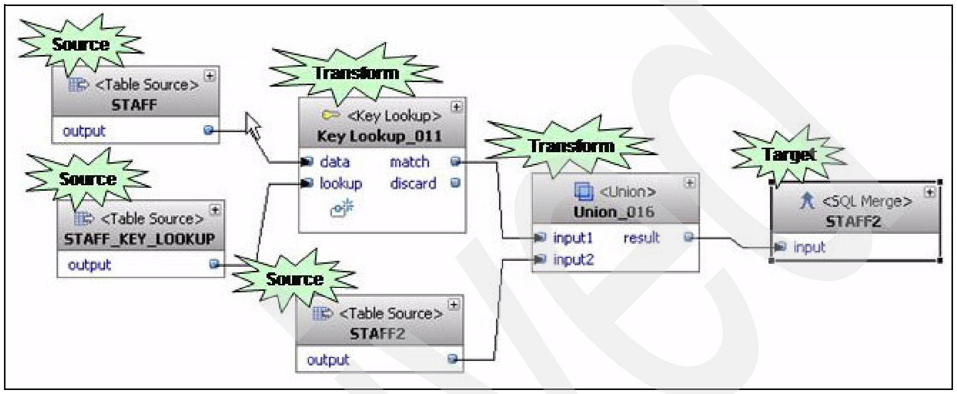


Figure 5-6 Operators in a simple data flow

Operators are linked together using connectors, as shown in Figure 5-7. Connectors represent how the data flows from one operator to another. In this example, the output from the STAFF table source operator flows into the key lookup operator, as does the output from the STAFF_KEY_LOOKUP table source operator. The output from the match port of the key lookup operator flows into the union operator, as does the output from the STAFF2 table source operator. The result of the union operator is merged into the STAFF2 table using the SQL Merge operator.

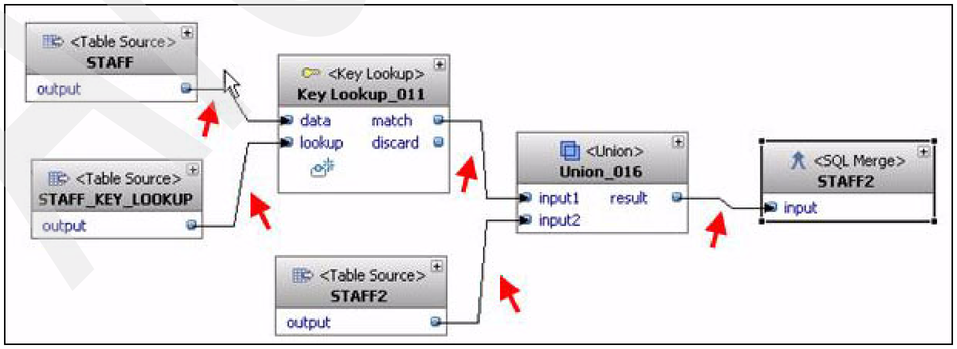


Figure 5-7 Connectors in a simple data flow

Connectors connect to the input and output ports of operators. Ports represent the virtual table definition of the data coming through that port, defining the virtual columns and the data type and size of each virtual column. The virtual table definition of output ports of source operators typically are initialized based on the column metadata of the underlying source or target operator, the source relational table for example.

The virtual table definition of output ports of other operators depend on the properties of the particular operator. The virtual port definition of input ports are typically initialized when first connected to an output port by propagating the virtual table definition from the connected output port. When initialized, the input port retains the virtual table definition even if the connector is deleted. During re-connection, you can specify if the output virtual table should be propagated to the input port and have the virtual table columns mapped by name or mapped by position. In addition, if a modification to an operator's properties changes the definition of the output port's virtual table, only the changes can be propagated.

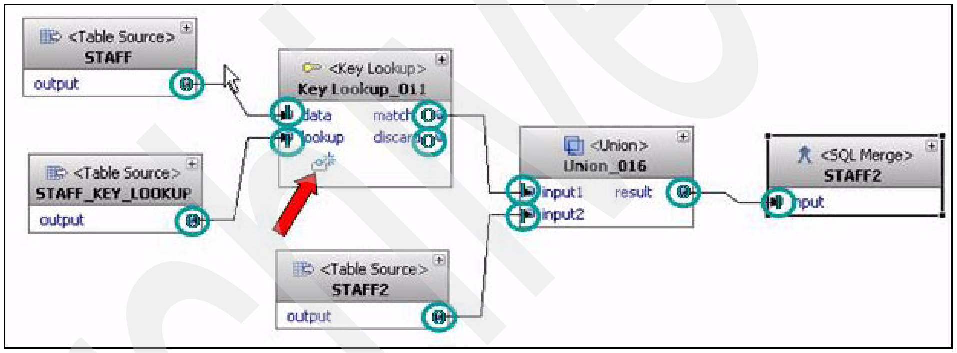


Figure 5-8 Ports in a simple data flow

Source, target, and execution databases

Before going further it is important to understand the definition of the term *execution database* and its relationship to source and target databases. The execution database is the database that does the transformation work when the SQL code in a data warehouse application runs. This database must be a DB2 database appropriate to the platform of your InfoSphere Warehouse product.

The execution database alias is a primary property of a data flow and all SQL code for that data flow is executed at the execution database. Different data flows can have different execution databases. The data flow SQL code is submitted to the execution database defined in the data flow properties. (For example, if you need to move data from a warehouse database to a data mart in another DB2 database or subsystem.) Part of the processing can happen at the data warehouse database and part of the processing can happen at the data mart database.

The notion of a remote or local database source or target is relative to the execution database where the SQL code is submitted. If the source or target database is the same as the execution database, then they are local databases. Only DB2 databases can be local databases. When the target or source database is different from the execution database, they are remote databases. A remote database is accessed by the Java Database Connectivity interface standard (JDBC) and might be a non-DB2 relational database.

Figure 5-9 shows four possible configurations of source schema, execution database, and target schema that can be supported with SQW.

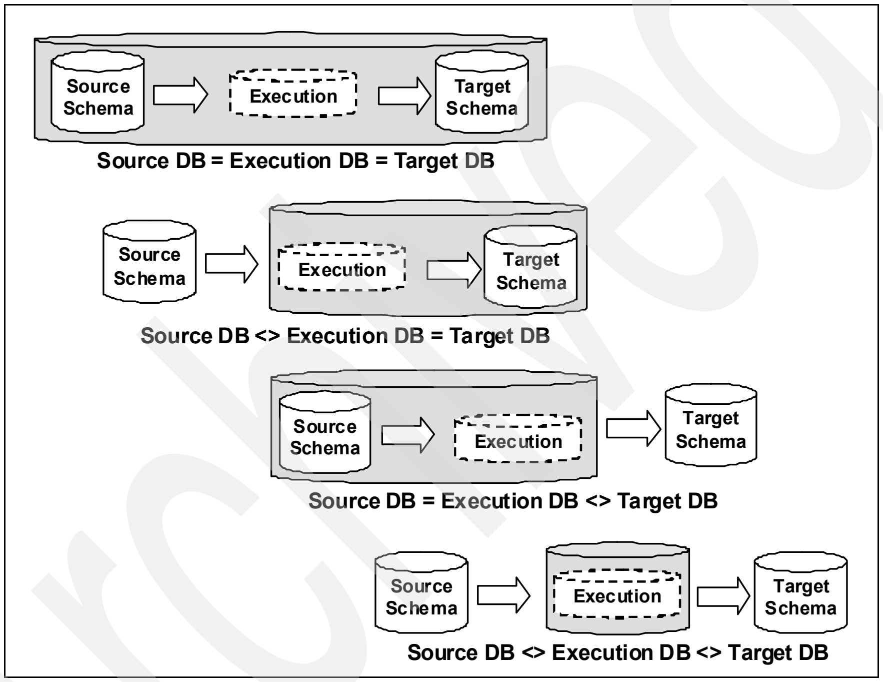

Source Schema
Execution
Target Schema
Source DB = Execution DB = Target DB
Source Schema
Execution
Target Schema
Source DB <> Execution DB = Target DB
Source Schema
Execution
Target Schema
Source DB = Execution DB <> Target DB
Source Schema
Execution
Target Schema
Source DB <> Execution DB <> Target DB

Figure 5-9 Configurations of source, execution, and target databases

In Figure 5-9 The database symbol with the dashed lines represents the DB2 execution database. The database symbols with the solid lines represent source or target schemas. The shaded symbol represents the local database.

The performance characteristics of these various configurations can be significantly different, as remote databases incur network overhead:

- Source DB = Execution DB = Target DB scenario

  This scenario occurs when the source schema and the target schema are local to the execution database. In this case all of the data is in one database and DB2 can process the table-to-table data movement and transformations effectively without the data flowing out of the database.

- Source DB <> Execution DB = Target DB scenario

  This scenario occurs when the source data is not local to the execution database. This could be a database on a remote server, another database on the same server, a non-DB2 database, or a flat file. In this case, the necessary source data must be staged to the execution database before it can be processed.

- Source DB = Execution DB <> Target DB scenario

  This scenario occurs when the target is not local to the execution database. This could be a database on a remote system, another database on the same server, or a non-DB2 database. In this case the data has to flow out of the execution database to the external target using remote inserts.

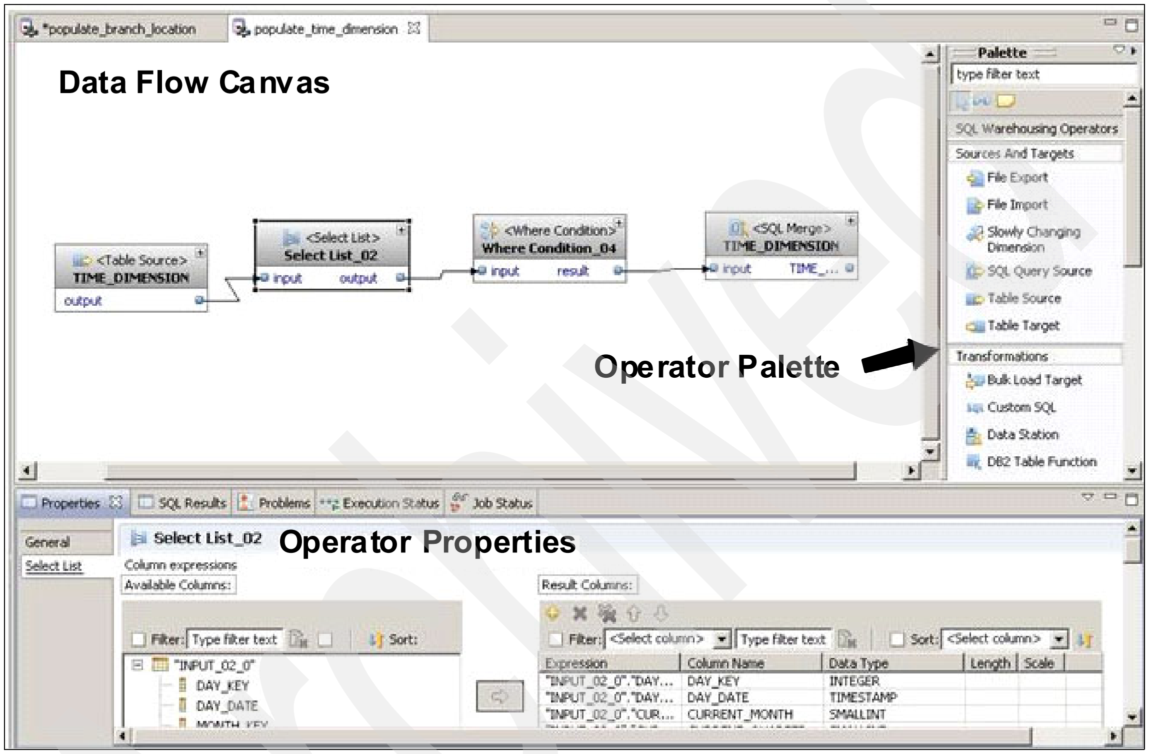
- Source DB <> Execution DB <> Target DB scenario

  This scenario occurs when both the source data and the eventual target are not local to the DB2 database. The source data has to be staged to the execution database for processing and sent to the remote target for updating.

Though all of these scenarios are supported by SQW, the most effective performance is gained by collocating the target and execution database as shown in the first two scenarios.

Data flow editor

In the InfoSphere Warehouse Design Studio, data flows are developed using a number of common Design Studio functions (such as the Data Project Explorer) views (such as Properties, Data Output, and Problems) and, optionally, the Database Explorer. However, the actual creation and editing of data flows occurs in a specific graphical editor called the data flow editor, depicted in Figure 5-10.



Expression	Column Name	Data Type	Length	Scale
"INPUT_02_0"."DAY...	DAY_KEY	INTEGER		
"INPUT_02_0"."DAY...	DAY_DATE	TIMESTAMP		
"INPUT_02_0"."CUR...	CURRENT_MONTH	SMALLINT		

Figure 5-10 Data flow editor and properties

As with most graphical editors, the data flow editor follows the drag, drop, connect, and set properties paradigm of development. When you create a new data flow or open an existing data flow, the data flow editor opens in a new tab in the editor area of Design Studio. The data flow editor consists of a canvas (onto which graphical elements are drawn) and an operator palette (that contains all of the graphical elements that are relevant to building a data flow). In addition, you use the Properties View tab to define the characteristics of the objects on the canvas. As you validate and test run the data flows, you also use the Problems view tab and the Data Output tab.

Attention: Properties can be set two ways:

- Properties View tab

  This is shown in Figure 5-10.

- Properties wizard

  The Properties wizard shows the same property pages as in the Properties View except in a wizard-like fashion. The properties wizard is accessed from the context menu of each operator. Also, a few operators open the wizard automatically when dropped onto the canvas.

  Certain developers work only with the Properties Views and others work exclusively with the properties wizard. It is said that there are always at least two ways to do things in Design Studio.

The following list details a few things to consider when developing data flows:

- Be familiar with the common functions in Design Studio. This includes the following areas:
  - Working with perspectives
  - Working with views
  - Using the Data Project Explore
  - Using the Database Explorer
  - Dragging elements from the Data Project Explorer and palette to the canvas
- Orient the data flow from left to right. This makes a neater diagram because the output ports are on the right side of an operator and the input ports are on the left side.
- Work with just a few operators at a time.
- Operators have properties that must be defined.
- Operator input and output ports have properties that might have to be manually defined or modified. The properties of these ports are the virtual table schemas (column definitions) of the data that flows between the operators. These schemas might have to be managed as the flow progresses.
- A data flow itself has properties with the execution database. Where this data flow is to execute is the most important.

Variables in a data flow

In Design Studio data flows, as in most development tools, variables play a important role and allow a routine to be generalized. For example, suppose you have an input file that will be a source in your flow. However, the name of the file might change from one execution to another. You can use a variable for the file name. The name of the file can then be provided at each execution time.

A related use is when you are developing and testing in a Windows environment but will deploy in a UNIX environment. Path names are different between Windows and UNIX. Again, a variable can be used for the path name and set at deployment time perhaps.

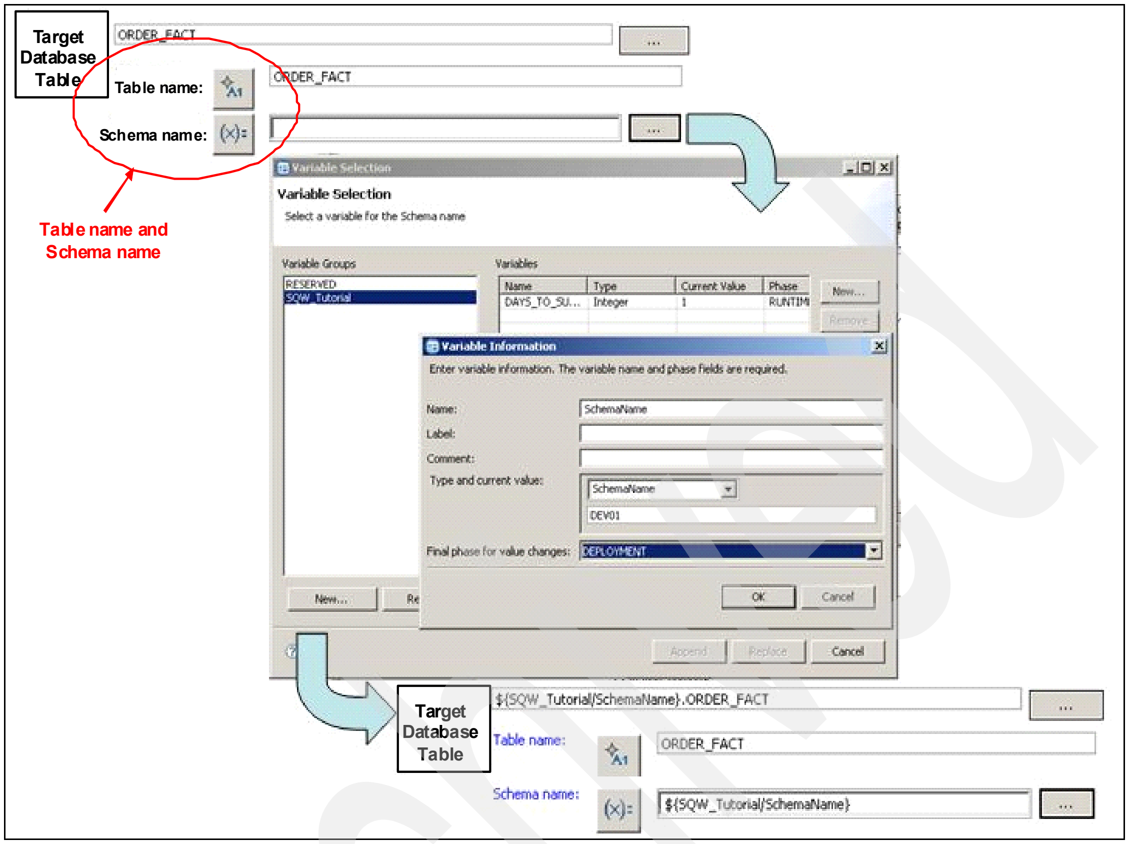
It is also common that the schema names of database tables are different between development, test, and production databases. Here, a variable can be used for the table schema name and set at deployment time.

In data flows, variables can be used in many property fields. They can also be used in relational expressions. Figure 5-11 on page 91 shows the use of a variable in the property field for the schema name of a target table. If a variable can be used in a field, there is a button in front of the properties field. An example is shown in Figure 5-11 on page 91 where the Table name and Schema name fields, and associated buttons, have been highlighted. By clicking one of the buttons, you can select whether the value in that field is a constant or a variable. The Table name field is a constant and the Schema name field is a variable. Clicking the button allows you to change between a variable or constant.

If a field is defined to contain a variable, an ellipses button shows at the end of the field. Clicking this button opens the Variable Selection dialog box in which variables are defined and selected for use. Figure 5-11 on page 91 shows the SchemaName variable being defined with a variable type of SchemaName and a default value of DEV01.

The “Final phase for value changes” field defines where in the development life cycle this value can be modified. This SchemaName variable is expected to be set at DEPLOYMENT time when the flow is installed into a runtime environment. When deployed to the test environment, the value can be set to **TEST01**, for example, and to **PROD01** for the production environment. After the value is selected, the value in the property field contains the variable name. Later in this section, we discuss variables used in a relational expression.

Tip: Variables can be appended with constants or other variables to form more complex variable structures. For example, a property field for a file name could be X:${myvars/path}/${myvars/filename}. If the variable myvars/path is /prod/app1 and the variable myvars/filename is input01.csv, the resolved field value at execution time will be X:/prod/app1/input01.csv.



Target Database Table: ORDER_FACT

Table name: ORDER_FACT

Schema name:

Variable Selection

Select a variable for the Schema name

Variable Groups: RESERVED, SQW_Tutorial

Name	Type	Current Value	Phase
DAYS_TO_SU...	Integer	1	RUNTIME

Enter variable information. The variable name and phase fields are required.

Name: SchemaName

Label:

Comment:

Type and current value: SchemaName, DEV01

Final phase for value changes: DEPLOYMENT

Target Database Table: ${SQW_Tutorial/SchemaName}.ORDER_FACT

Table name: ORDER_FACT

Schema name: ${SQW_Tutorial/SchemaName}

Figure 5-11 Using a variable in a data flow property field

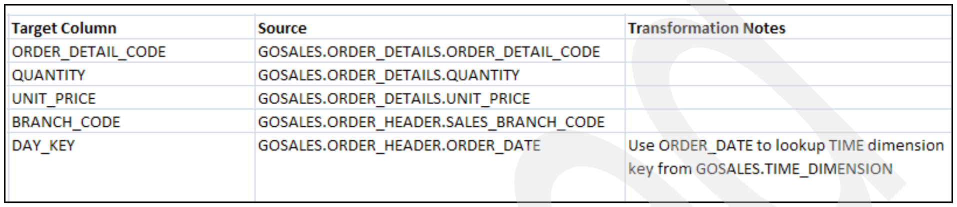
Tip: The Variable Selection dialog box can also be reached from the Data Project Explorer by right-clicking the Variables folder, selecting **Manage Variables** from the context menu and selecting the appropriate data warehouse project.

5.2.2 Developing a simple data flow

Now that you have a basic understanding of data flows, turn your attention to developing and testing a simple data flow. This section is not intended to be a step-by-step tutorial but to give you a feel for the development techniques. Step-by-step tutorials can be found in the InfoSphere Warehouse Information Center for your product platform.

Before beginning to develop a data flow, you must understand what it is to do, and the data sources and targets that will be used. Our simple example is a data flow that updates the data warehouse table, MARTS.ORDER_FACT from the source database order tables, GOSALES.ORDER_HEADER and GOSALES.ORDER_DETAILS. The data warehouse database is GSDB.

Figure 5-12 defines the mapping between the source tables and the target tables.



Target Column	Source	Transformation Notes
ORDER_DETAIL_CODE	GOSALES.ORDER_DETAILS.ORDER_DETAIL_CODE	
QUANTITY	GOSALES.ORDER_DETAILS.QUANTITY	
UNIT_PRICE	GOSALES.ORDER_DETAILS.UNIT_PRICE	
BRANCH_CODE	GOSALES.ORDER_HEADER.SALES_BRANCH_CODE	
DAY_KEY	GOSALES.ORDER_HEADER.ORDER_DATE	Use ORDER_DATE to lookup TIME dimension key from GOSALES.TIME_DIMENSION

Figure 5-12 Source to target mapping requirements

A requirement is that we pull orders from the source based on a specified date range. This range is provided in a file at runtime.

Now that the requirements are met, we can begin to develop a simple data flow.

Logical strategy

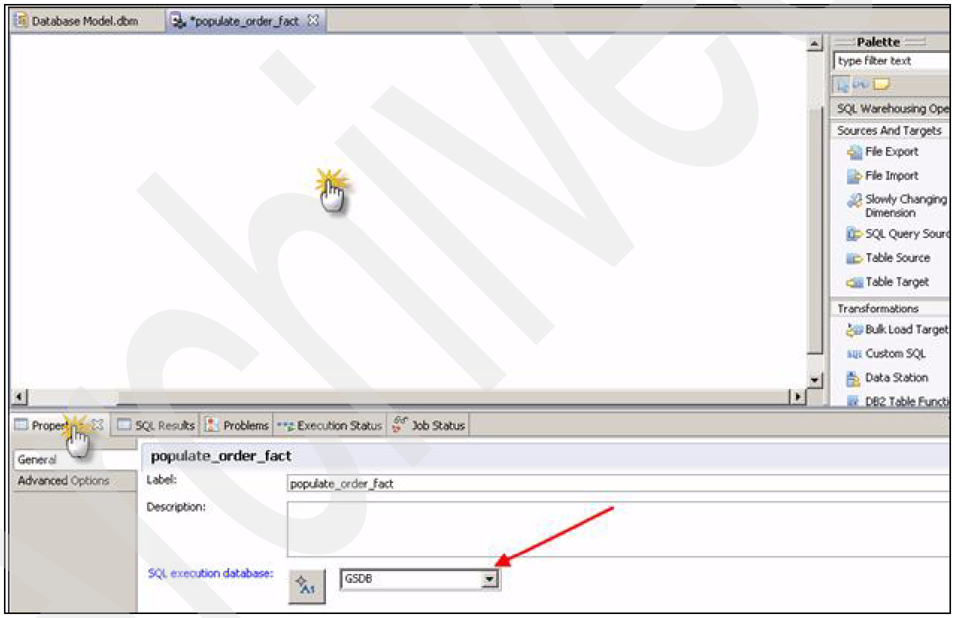
Looking at the requirements we can come up with a high-level strategy to accomplish the work. The strategy is as follows:

1. Obtain the source data from the two tables.
2. Because the two tables update one table, we have to join the two tables using an inner join.
3. Subselect the columns taking on the five that we need.
4. Filter the orders to select only the ones that are in the specified time period.
5. Obtain the surrogate key of the time dimension based on the date of the order.
6. Insert the new orders into the target table.

Creating a new data flow

To create a new data flow, open your data warehouse project (or create a new project), right-click the Data Flows folder and click **New** → **Data Flow**. This opens a new data flow editor similar to Figure 5-13 with an empty canvas. One task that should be done at this time is to define the database at which this data flow executes. This is called the *execution database* and is typically the target database. For a discussion of the execution database, see "Source, target, and execution databases" on page 85.

The execution database is a property of the data flow itself. To see the properties of the data flow, click any white space on the data flow canvas, as indicated by the hand icon shown in Figure 5-13. To actually access the data flow properties use the Properties View tab, which is highlighted at the bottom of the white space with another hand icon. You can also set the SQL execution database, as indicated with the arrow in Figure 5-13.



populate_order_fact

Label: populate_order_fact

Description:

SQL execution database: GSDB

Figure 5-13 Setting the execution database

Tip: A data flow can be created as offline or online. Unless there is an overriding reason, use offline data flows. An online data flow requires a live connection to the source and target databases and creates implicit data models that must be kept in synch. Metadata synchronization takes a lot of overhead and can slow down Design Studio during the metadata synchronization operation.

The data flow is now ready for us to start adding operators.

Accessing the source data

Source operators access various types of source databases and files, extract the data and bring it into the data warehouse. The primary source type is the relational table. Any relational database from which a data model can be created can be used as a source table. See Chapter 4, "Developing the physical data model" on page 53 for more information about the types of databases supported.

Files might also be imported into the execution database. In general, the types of files supported are the ones supported by the load and import utilities on the DB2 database platform. For example, on the LUW platform, various types ASCII files are supported (such as delimited, fixed format and PC/IXF files), on System z, EBCDIC, ASCII and UNICODE file types are supported in flat file, delimited, UNLOAD and SQL/DS formats.

For our example data flow, there are two relational tables for source data, GOSALES.ORDER_HEADER, and GOSALES.ORDER_DETAILS. These are highlighted in Figure 5-14 on page 95. To access and extract that source data from those relational tables, we need operators. The table source operator is used to extract data from relational tables. The following are the steps for extracting the source data:

1. From the data flow editor palette, drop a Table Source operator onto the canvas.

   The table source operator is one of the few operators that automatically opens the properties wizard. You can work with the wizard or close it and work with the Properties View, as they contain the same properties pages but presented in different ways.

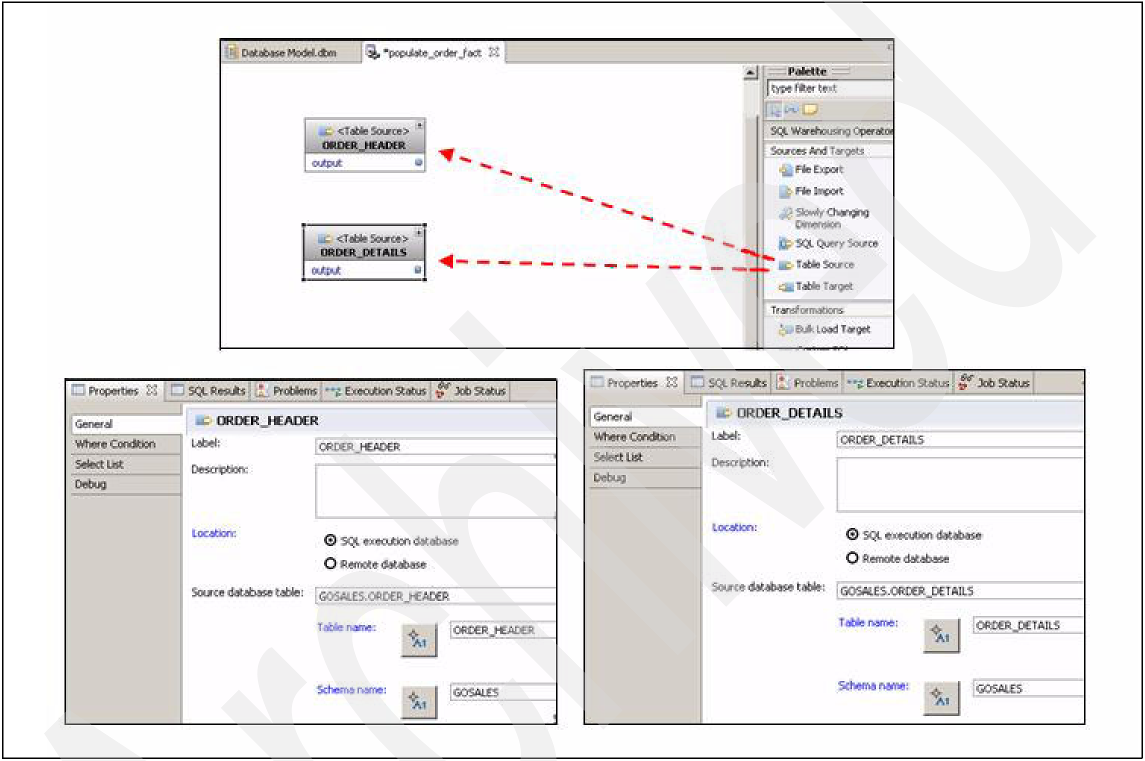
   The table associated with the operator must be set is found on the General tab in the Source database table field.

2. Click the ellipse button that is to the right of the field to open the table picker and select the desired table from the appropriate data model. This populates the Table name, Schema name, and Source database name fields. It also uses the metadata from the data model to create the virtual table of the output port.

   Tip: You can drag a table name from the data project explorer data model to the data flow editor canvas. You are prompted to select the type of operator to create: Source Table, Target Table, or Data Station. The appropriate operator is created on the canvas with the table name property fields populated.

3. Also on the General property tab is the property to specify whether the table is local or remote in relation to the execution database for the data flow. Selecting remote presents another property to specify the connection for the remote database. In this example, the tables are in the GSDB database therefore local to the execution database.

   Figure 5-14 shows data flow editor with the two source tables defined.



ORDER_HEADER
Label: ORDER_HEADER
Description:
Location: [✓ SQL execution database] Remote database
Source database table: GOSALES.ORDER_HEADER
Table name: ORDER_HEADER
Schema name: GOSALES

ORDER_DETAILS
Label: ORDER_DETAILS
Description:
Location: [✓ SQL execution database] Remote database
Source database table: GOSALES.ORDER_DETAILS
Table name: ORDER_DETAILS
Schema name: GOSALES

Figure 5-14 Using the table source operator

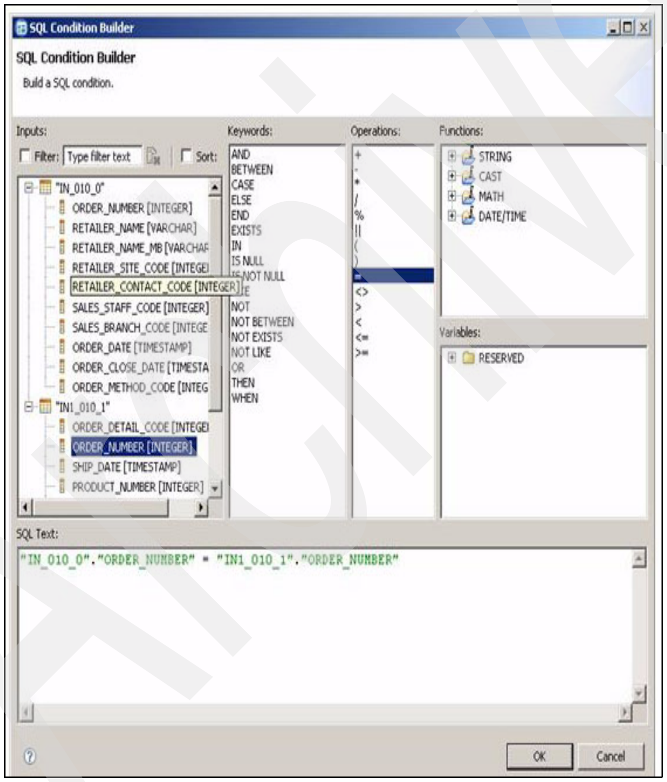
The other properties for the table source operator, “Where Condition” and “Select List”, are most appropriate for remote tables to push processing to the remote database manager. This reduces the amount of data to be sent over the network. These two properties of the table source operator are not necessary when accessing local tables because the Where Condition and Select List operators themselves can be used in that situation. When using these two local operators, the metadata in the flow is exposed and combined with the table source operator when code is generated.

The Where Condition property is used to send a Where Condition to the remote database to filter the returned rows. This needs to be written in the SQL dialect appropriate to the source database system. Be aware that there is no validation on this property.

The Select List property is used to select only the needed columns and to create derived columns again in an effort to push processing down to the source database system. This is to reduce traffic over the network. As with the Where Condition property, the SQL needs to be written in the SQL dialect for the source database system, and no validation is performed.

Joining the data

The requirements in Figure 5-12 on page 92 map the two source tables to the one target table. This requires that we join the tables together and select only the needed columns. To do that, we use the Table Join Operator. The Table Join Operator can perform inner, left outer, right outer, and full outer joins. This is determined by using the appropriate output port of the Table Join Operator. An example of using the Table Join Operator is now presented for the scenario depicted in Figure 5-15.



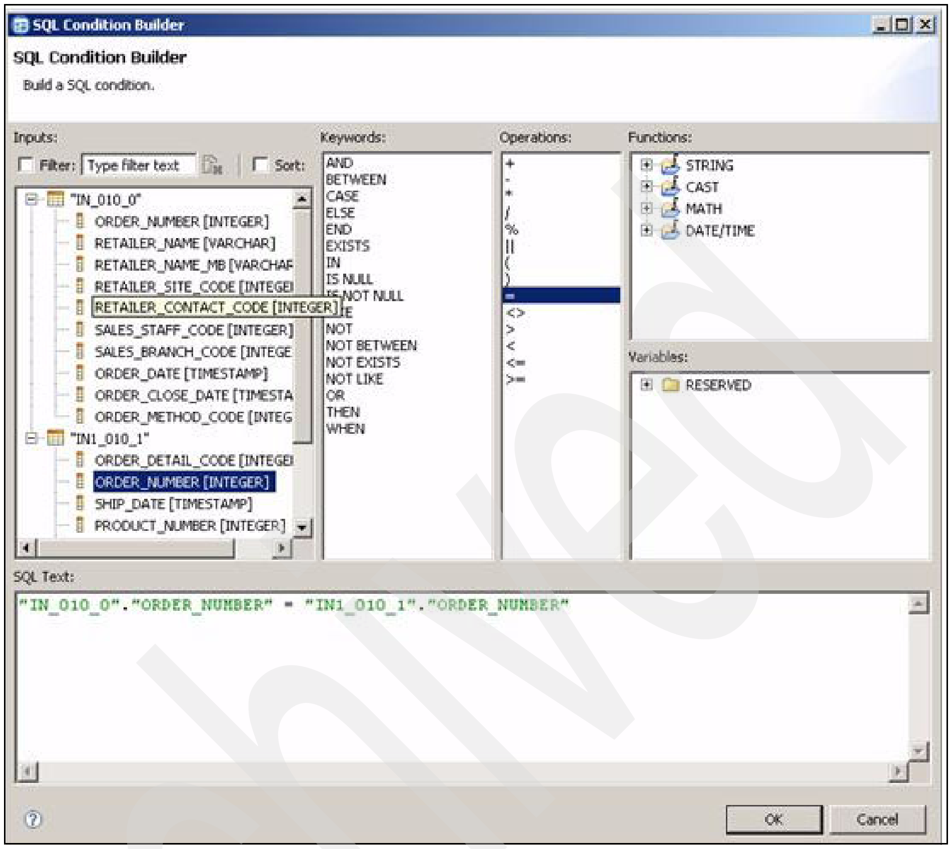
```
"IN_010_0"."ORDER_NUMBER" = "IN1_010_1"."ORDER_NUMBER"
```

Figure 5-15 Using the table join operator

The following steps are required for the join using the Table Join Operator:

1. Select the Table Join operator from the palette and drop it on the canvas to the right of the table source operators.
2. Connect the output port of the ORDER_HEADER table source operator to the first input port of the Table Join operator with the label **in**.

   This propagates the virtual table definition from the output port to the input port. This can be seen by expanding the two operators. Do this by clicking the expansion button (+) in the upper right corner of each operator to see the details of the mapping. Clicking the expansion button again (-) returns the operator to its normal view. You can also view the properties for the virtual table by selecting the port label and opening the Properties View.
3. Connect the output port of the ORDER_DETAILS table source operator to the input port of the Table Join operator labeled `in1`. The join condition is defined using the Condition property page, depicted in Figure 5-16 on page 98.
4. Clicking the ellipses button opens the Expression Builder, which is used to build the condition expression specifying which columns and how they are going to be compared.



```
"IN_010_0"."ORDER_NUMBER" = "IN1_010_1"."ORDER_NUMBER"
```

Figure 5-16 Defining the join condition using the Condition Builder

According to the requirements specified in Figure 5-12 on page 92, we only need a subset of the source columns. We need three columns from ORDER_DETAILS and the two from the ORDER_HEADER.

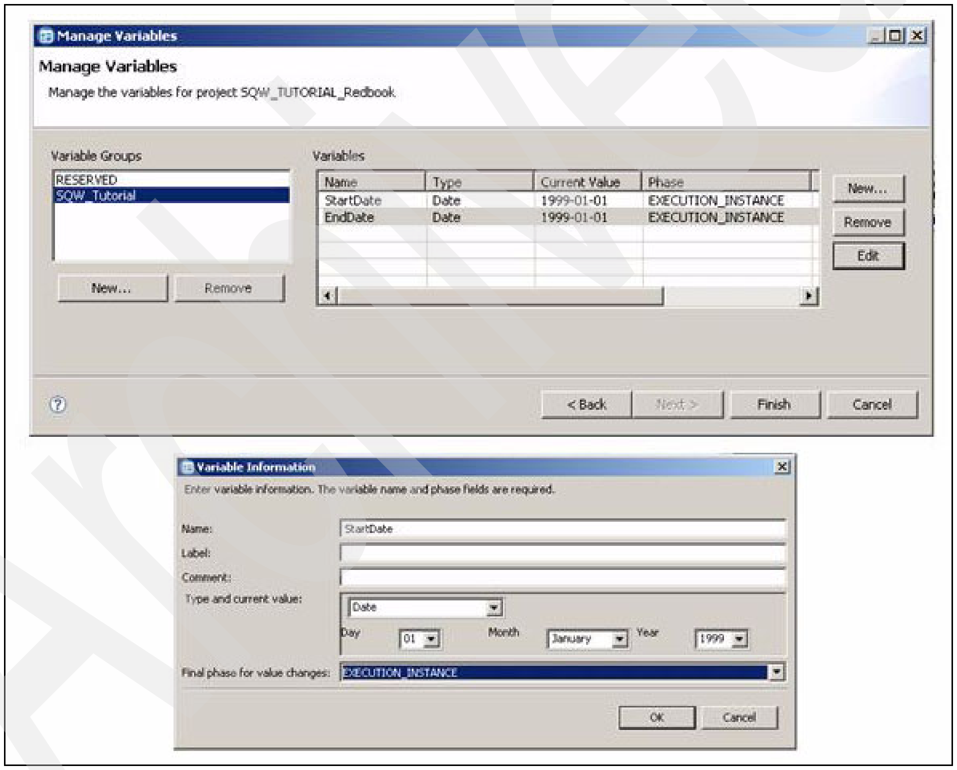
5. Use the Select List property page to select the output columns from the available input columns and to add derived columns. By default, all input columns are selected. Select only the columns needed to pass through to the next operator. These columns define the structure of the output virtual table.

> **Attention:** The Select List property performs the same function as the Select List operator, but is embedded into various operators as a property page.

Filtering the data

Now that the incoming data has been joined, we need to select the orders that fall in a specified date range. The Where Condition Operator is used to filter data. Recall that the date range is provided in a file at runtime. We therefore need variables.

1. Define data range variables so that the variables are available for use in the expression builder.
2. Open the Manage Variables dialog box from the Data Project Explorer. Define a group, SQW_Tutorial, to define our variables (Figure 5-17). Also define two variables, StartDate and EndDate of type Date.
3. Set the Phase to EXECUTION_INSTANCE so that variables can be set in flow execution. The variables are provided in a file, which can be created later.


Manage Variables

Manage the variables for project SQW_TUTORIAL_Redbook

Variable Groups: RESERVED, SQW_Tutorial

Variables

Name	Type	Current Value	Phase
StartDate	Date	1999-01-01	EXECUTION_INSTANCE
EndDate	Date	1999-01-01	EXECUTION_INSTANCE

Variable Information

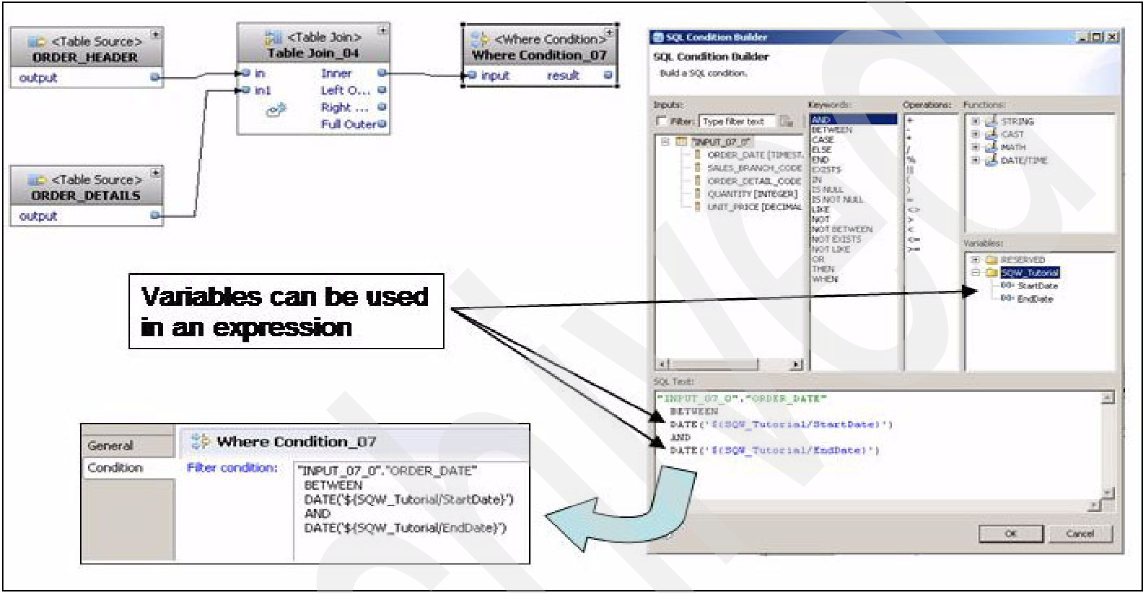
Enter variable information. The variable name and phase fields are required.

Name: StartDate
Label:
Comment:
Type and current value: Date
Day 01 Month January Year 1999
Final phase for value changes: EXECUTION_INSTANCE

Figure 5-17 Defining order date range variables

4. Define the filter by adding a Where Condition operator to the canvas on the right of the table join operator, as seen in Figure 5-18.

   The filter condition is specified on the Condition Property page using the Expression Builder. The date range is specified in the variables defined in step 2 on page 99, and is available in the Expression Builder. The dates are used in a BETWEEN predicate to select only the orders that have an ORDER_DATE in the data range (inclusive).



```
"INPUT_07_0"."ORDER_DATE"
  BETWEEN
  DATE('${SQW_Tutorial/StartDate}')
  AND
  DATE('${SQW_Tutorial/EndDate}')
```

Figure 5-18 Using the Where Condition operator

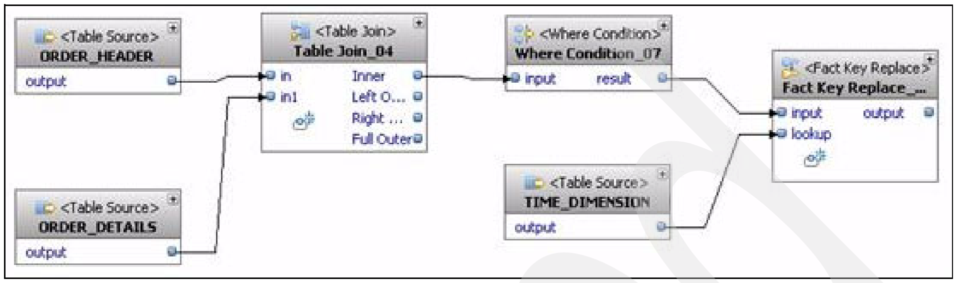
Replacing natural key with surrogate key

The target table is a fact table that uses surrogate keys for the time dimension. The order records that we have at this point contain a date for time which is the natural key. Therefore we must convert the natural key of date to the surrogate key for the time dimension.

The requirements state to use the GOSALES.TIME_DIMENSION table to lookup up the date and replace the natural key, ORDER_DATE with the key from the TIME_DIMENSION table.

The Fact Key Replace operator is used to lookup up one or more natural keys from the input port using lookup tables and replaces the natural keys with the surrogate keys found in the lookup tables. We use the TIME_DIMENSION table as our lookup table, so we need to add a table source operator for GOSALES.TIME_DIMENSION and a Fact Key Replace operator, as shown in

Figure 5-19. The result port of the Where Condition is connected to the input port of the Fact Key Replace operator and the output port from the TIME_DIMENSION Table Source operator is connected to the lookup port. Additional lookup ports can be added.

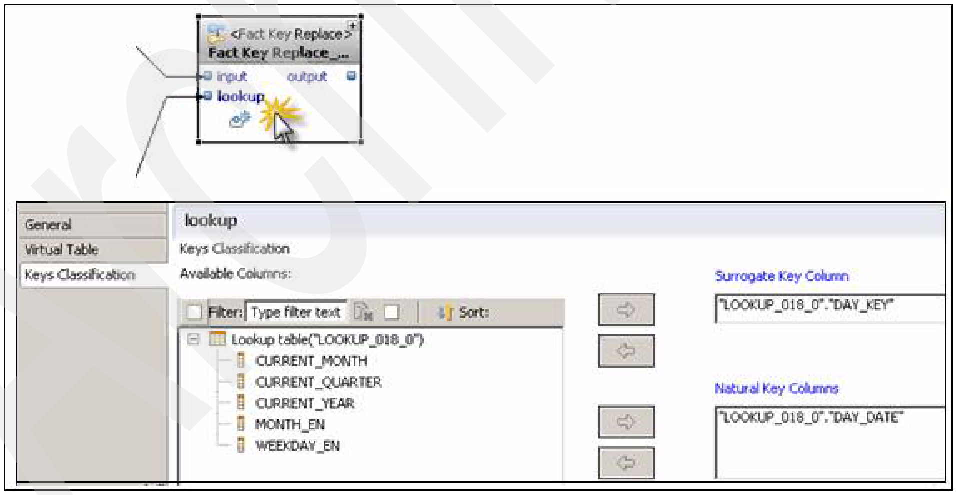


<Table Source> ORDER_HEADER
output
<Table Source> ORDER_DETAILS
output
<Table Join> Table Join_04
in
in1
Inner
Left O...
Right ...
Full Outer
<Where Condition> Where Condition_07
input
result
<Table Source> TIME_DIMENSION
output
<Fact Key Replace> Fact Key Replace_...
input
lookup
output

Figure 5-19 Add Fact Key Replace operator

Figure 5-20 show how to classify the surrogate and natural keys for the lookup table. Perform the following steps:

1. Click the lookup port in the Fact Key Replace operator.
2. In the Keys Classification of the virtual table Properties View, select the column to be the surrogate key, DAY_KEY, and the column to be the natural key, DAY_DATE. Note that the natural key might contain multiple columns.



lookup

Keys Classification

Available Columns:

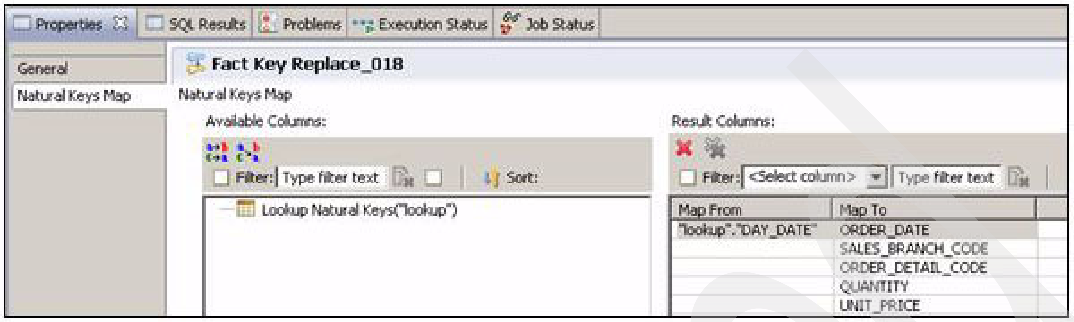
Lookup table("LOOKUP_018_0")
- CURRENT_MONTH
- CURRENT_QUARTER
- CURRENT_YEAR
- MONTH_EN
- WEEKDAY_EN

Surrogate Key Column: "LOOKUP_018_0"."DAY_KEY"

Natural Key Columns: "LOOKUP_018_0"."DAY_DATE"

Figure 5-20 Classifying surrogate and natural keys

3. Map the natural keys of the lookup tables to the appropriate column of the input table. The natural key of the time dimension lookup table, DAY_DATE, maps to the ORDER_DATE columns of the input, as shown in Figure 5-21.



Map From	Map To
"lookup"."DAY_DATE"	ORDER_DATE
	SALES_BRANCH_CODE
	ORDER_DETAIL_CODE
	QUANTITY
	UNIT_PRICE

Figure 5-21 Map natural keys to input columns

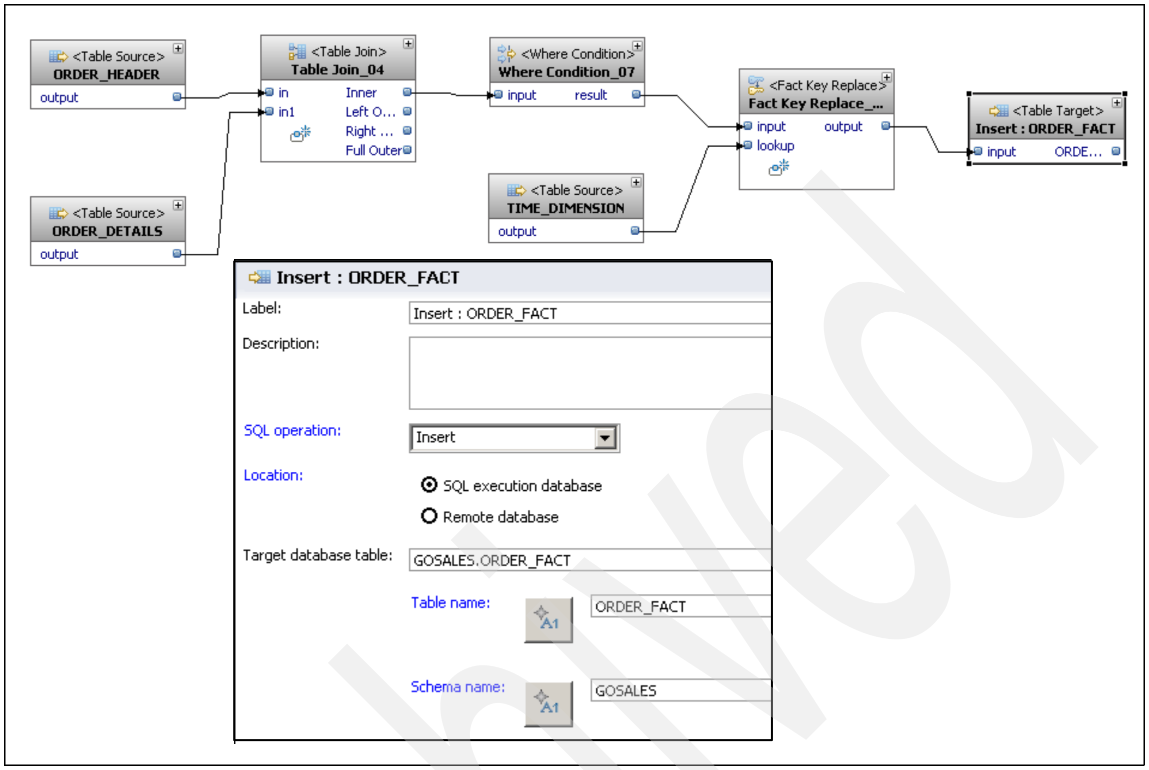
Updating the target table

The final step of the data flow is to insert the resulting rows into the target table. There are multiple operators that use different techniques to update target tables from a simple insert to using DB2 bulk load utilities. In this example, we use a simple insert operation using the Table Target operator.

1. Add a Table Target operator and pick GOSALES.ORDER_FACT as the target database table, as shown in Figure 5-22 on page 103.
2. Set the SQL operation property to **Insert**.

   A Table Target operator can insert incoming data, delete data based on the incoming data or update rows based on incoming data. These are mutually exclusive operations and do not constitute an UPSERT type of operation.

Attention: We have intentionally introduced an error at this point, so we can illustrate debugging techniques later in the chapter.



Insert : ORDER_FACT

Label: Insert : ORDER_FACT

Description:

SQL operation: Insert

Location: [✓ SQL execution database] Remote database

Target database table: GOSALES.ORDER_FACT

Table name: ORDER_FACT

Schema name: GOSALES

Figure 5-22 Using the Table Target operator

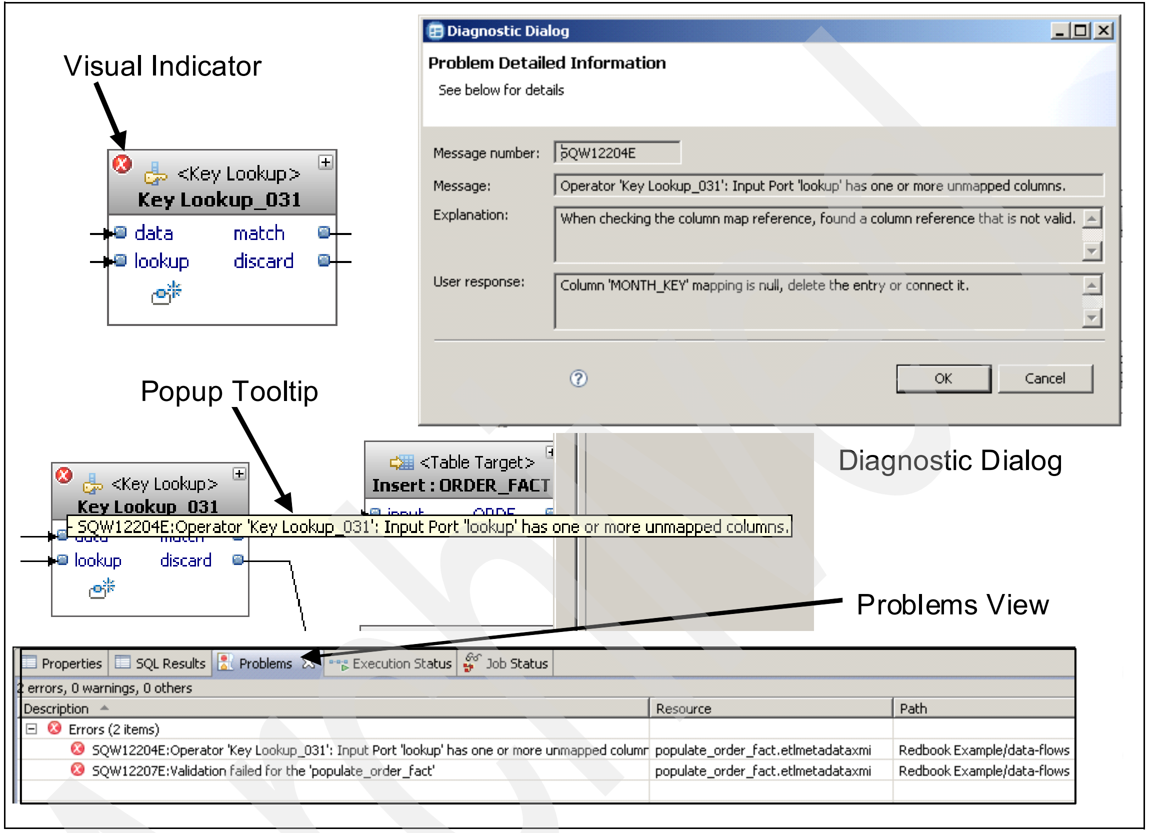
This completes the development of the data flow to bring new orders into the data warehouse.

5.2.3 Validating and finding problems in a data flow

The process of validation examines the data flow to identify specific kinds of problems. Validation can be invoked by selecting **Data Flow** → **Validate** from the menu bar. By default, validation is also done when a data flow is saved. Validation is also invoked as part of executing a data flow in Design Studio.

If any problems are found, there are various indicators set in the data flow, as shown in Figure 5-23 on page 104. There might be visual indicators in the upper left corner of operators that have a problem. Holding the cursor over the indicator opens a tooltip displaying text about the problem. Clicking the indicator opens up a Diagnostic Dialog box with additional details.

All problems that exist in the workspace are listed in the Problems View tab. The Resource column identifies which Design Studio element contains the error. Clicking the Resource column header sorts the problems by resource name to group together all problems for a resource.



Diagnostic Dialog

Problem Detailed Information

See below for details

Message number: SQW12204E

Message: Operator 'Key Lookup_031': Input Port 'lookup' has one or more unmapped columns.

Explanation: When checking the column map reference, found a column reference that is not valid.

User response: Column 'MONTH_KEY' mapping is null, delete the entry or connect it.

SQW12204E:Operator 'Key Lookup_031': Input Port 'lookup' has one or more unmapped columns.

2 errors, 0 warnings, 0 others

Description	Resource	Path
Errors (2 items)		
SQW12204E:Operator 'Key Lookup_031': Input Port 'lookup' has one or more unmapped columr	populate_order_fact.etlmetadataxmi	Redbook Example/data-flows
SQW12207E:Validation failed for the 'populate_order_fact'	populate_order_fact.etlmetadataxmi	Redbook Example/data-flows

Figure 5-23 Finding problems in a data flow

Problems might be errors or might be warnings. Errors prevent subsequent execution of the data flow. Warnings do not prevent execution. You have to make a judgement call on whether the warnings are a problem. Even if a data flow validates successfully, it does not guarantee that it will produce the desired results. There might be errors that validation cannot detect and there might be errors in the logic. For this reason, testing the data flow is required.

5.2.4 Testing and debugging

The next step is to see if the data flow does what it is designed to do. The data flow can be tested without leaving Design Studio. A data flow can be manually executed from start to finish, or breakpoints can be added to the flow and the flow executed in debug mode. How you go about testing and debugging is a matter of personal preference.

You can develop your data flow in an incremental manner, adding a few operators and testing up to that point. This can be accomplished by adding a Data Station or Export operator at the end of the flow and executing or debugging the flow successfully before adding to the data flow.

Certain developers execute the flow first and only go into debug mode if there are problems. Other developers establish breakpoints in the flow and execute the flow in debug mode first, fixing problems as they occur with a full final execution run. Both approaches validate the flow and generate the code before attempting execution.

Executing a data flow

We now take a look at how to execute a data flow manually without debugging. Execution is started by selecting **Execute** from the data flow menu. This opens a Flow Execution dialog box where execution parameters are defined. Figure 5-24 on page 106 shows the panels in the flow execution dialog box.

- General tab

  The General tab is where the current execution database is specified. This can be different from the defined execution database. There must be a current live connection established in the Data Source Explorer. You can also select a saved run profile from this page. For testing, it is helpful to save a run profile that has the run parameters already established for you.

- Diagnostics tab

  The Diagnostics tab contains diagnostic information. To get the maximum amount of information for finding errors, select **Both** for the Trace Level property field. The other fields define information about the log files.

- Resources tab

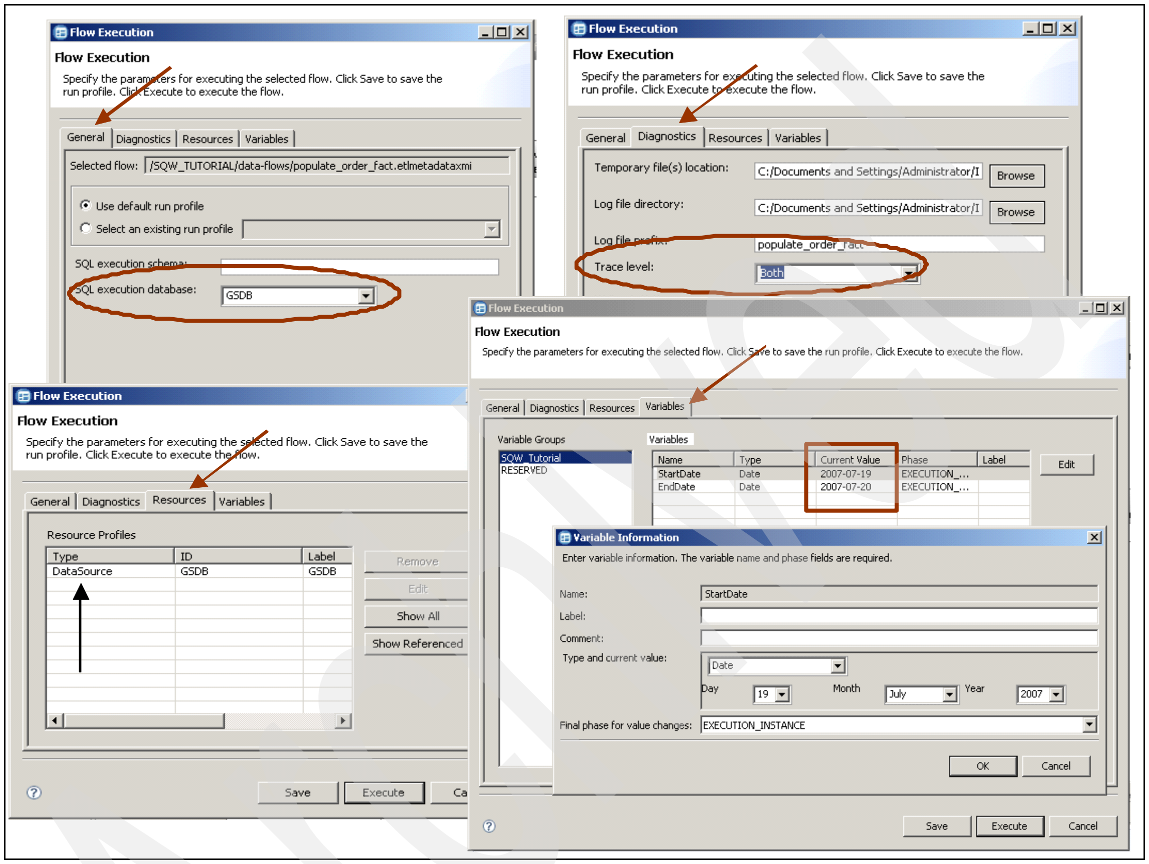
  The Resources tab shows resources used in the data flow, such as source and target databases. These can be remapped to other resource names.

- Variables tab

  The Variables tab lists all of the variables used in the data flow. Starting values can first be set. Set the StartDate and EndDate variables to provide a date range for testing.

Once the run parameters are set, they can be saved in a run profile that could be used in subsequent executions.

Click **Execute** to begin execution.


Flow Execution

Specify the parameters for executing the selected flow. Click Save to save the run profile. Click Execute to execute the flow.

Selected flow: /SQW_TUTORIAL/data-flows/populate_order_fact.etlmetadataxmi

(•) Use default run profile
( ) Select an existing run profile

SQL execution schema:
SQL execution database: GSDB

Temporary file(s) location: C:/Documents and Settings/Administrator/I
Log file directory: C:/Documents and Settings/Administrator/I
Log file prefix: populate_order_fact
Trace level: Both

Resource Profiles

Type	ID	Label
DataSource	GSDB	GSDB

Variable Groups: SQW_Tutorial, RESERVED

Variables

Name	Type	Current Value	Phase	Label
StartDate	Date	2007-07-19	EXECUTION_...	
EndDate	Date	2007-07-20	EXECUTION_...	

Variable Information

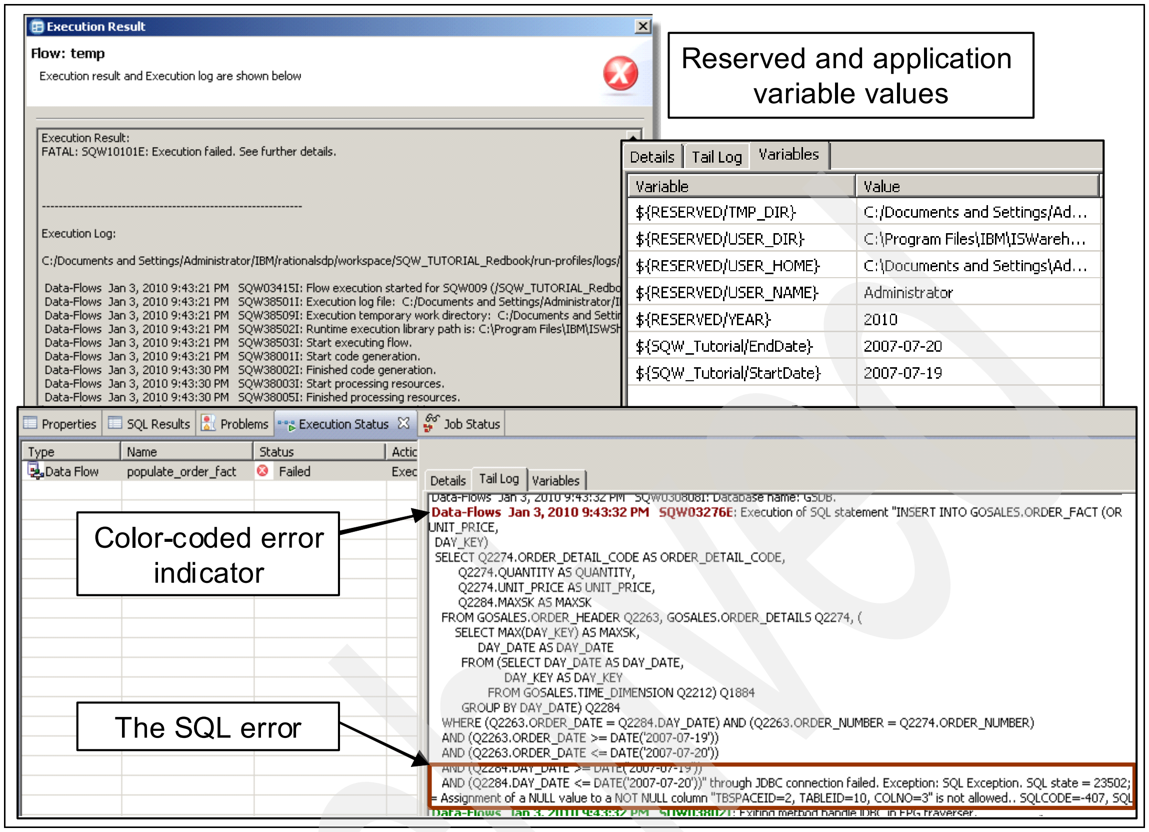
Enter variable information. The variable name and phase fields are required.

Name: StartDate
Label:
Comment:
Type and current value: Date
Day 19 Month July Year 2007
Final phase for value changes: EXECUTION_INSTANCE

Figure 5-24 Flow execution dialog boxes

Progress can be monitored using the Execution Status View tab, where you can view the log and values of variables during execution. This tab is shown in Figure 5-25 on page 107. When execution is complete, a completion message is displayed indicating success or failure and the log is displayed, as shown in Figure 5-25 on page 107. The log in the Execution Status View is color-coded, making it easier to find where errors occurred. Of course, this is only important if the execution is not successful.

In this example, our flow failed. Using the color-coding in the log, we can locate the error that indicated that we are trying to insert a NULL value into a NOT NULL column. Perhaps there is bad data.



```
Execution Result:
FATAL: SQW10101E: Execution failed. See further details.

------------------------------------------------------------

Execution Log:

C:/Documents and Settings/Administrator/IBM/rationalsdp/workspace/SQW_TUTORIAL_Redbook/run-profiles/logs/

 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW03415I: Flow execution started for SQW009 (/SQW_TUTORIAL_Redbo
 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW38501I: Execution log file:  C:/Documents and Settings/Administrator/I
 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW38509I: Execution temporary work directory:  C:/Documents and Setti
 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW38502I: Runtime execution library path is: C:\Program Files\IBM\ISWSH
 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW38503I: Start executing flow.
 Data-Flows  Jan 3, 2010 9:43:21 PM   SQW38001I: Start code generation.
 Data-Flows  Jan 3, 2010 9:43:30 PM   SQW38002I: Finished code generation.
 Data-Flows  Jan 3, 2010 9:43:30 PM   SQW38003I: Start processing resources.
 Data-Flows  Jan 3, 2010 9:43:30 PM   SQW38005I: Finished processing resources.
```

Variable	Value
${RESERVED/TMP_DIR}	C:/Documents and Settings/Ad...
${RESERVED/USER_DIR}	C:\Program Files\IBM\ISWareh...
${RESERVED/USER_HOME}	C:\Documents and Settings\Ad...
${RESERVED/USER_NAME}	Administrator
${RESERVED/YEAR}	2010
${SQW_Tutorial/EndDate}	2007-07-20
${SQW_Tutorial/StartDate}	2007-07-19

Type	Name	Status	Actic
Data Flow	populate_order_fact	Failed	Exec

```
Data-Flows  Jan 3, 2010 9:43:32 PM   SQW03080I: Database name: GSDB.
Data-Flows  Jan 3, 2010 9:43:32 PM   SQW03276E: Execution of SQL statement "INSERT INTO GOSALES.ORDER_FACT (OR
UNIT_PRICE,
 DAY_KEY)
 SELECT Q2274.ORDER_DETAIL_CODE AS ORDER_DETAIL_CODE,
      Q2274.QUANTITY AS QUANTITY,
      Q2274.UNIT_PRICE AS UNIT_PRICE,
      Q2284.MAXSK AS MAXSK
  FROM GOSALES.ORDER_HEADER Q2263, GOSALES.ORDER_DETAILS Q2274, (
     SELECT MAX(DAY_KEY) AS MAXSK,
           DAY_DATE AS DAY_DATE
       FROM (SELECT DAY_DATE AS DAY_DATE,
                  DAY_KEY AS DAY_KEY
             FROM GOSALES.TIME_DIMENSION Q2212) Q1884
       GROUP BY DAY_DATE) Q2284
  WHERE (Q2263.ORDER_DATE = Q2284.DAY_DATE) AND (Q2263.ORDER_NUMBER = Q2274.ORDER_NUMBER)
  AND (Q2263.ORDER_DATE >= DATE('2007-07-19'))
  AND (Q2263.ORDER_DATE <= DATE('2007-07-20'))
  AND (Q2284.DAY_DATE >= DATE('2007-07-19'))
  AND (Q2284.DAY_DATE <= DATE('2007-07-20'))" through JDBC connection failed. Exception: SQL Exception. SQL state = 23502;
= Assignment of a NULL value to a NOT NULL column "TBSPACEID=2, TABLEID=10, COLNO=3" is not allowed.. SQLCODE=-407, SQL
Data-Flows  Jan 3, 2010 9:43:32 PM   SQW03802I: Exiting method handle JDBC in EPG traverser.
```

Figure 5-25 Execution results

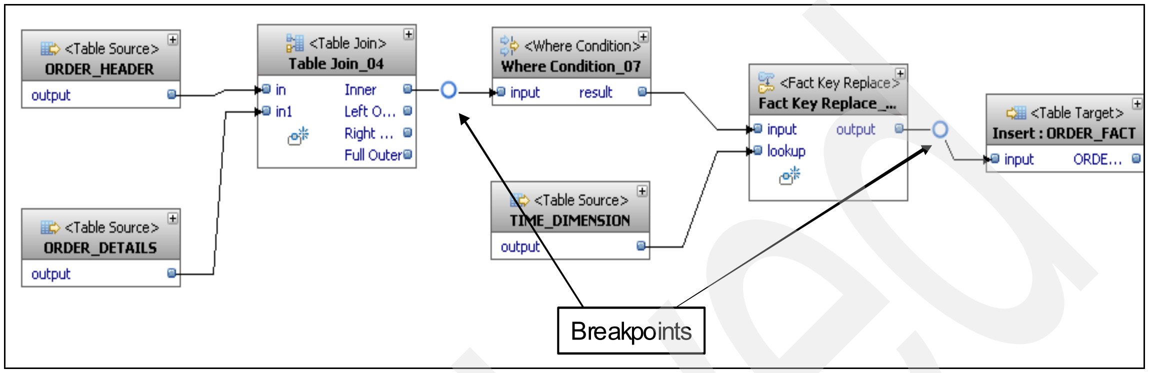
Debugging a data flow

Due to the nature of optimized generated code it can be difficult isolating which operator caused a particular issue. The debugger can be used to help isolate where in a data flow an error is occurring.

Breakpoints can be added to the connectors between operators. In a debug session execution stops at the defined breakpoint and shows the number of rows flowing through the breakpoint. The rows in the breakpoint can be sampled. You can even execute SQL against the breakpoint rows. This is because tables are created for each breakpoint, and data flowing through the connector is stored in that table.

Because we suspect that we are somehow getting NULLS in our data, we want to isolate which part of the data flow might be resulting in NULLS. If NULLS are in the source data, we could put breakpoints after the table source operators. For purposes of this example, assume that we have verified that the source data does not contain NULLS.

Most of our operators are simple. However we do have a join operator and a fact key replace operator, which is essentially a join that might be introducing NULLS. We put breakpoints on the connectors coming out of these two operators by right-clicking the connectors and selecting Toggle Breakpoint from the context menu. The breakpoints are shown in Figure 5-26.

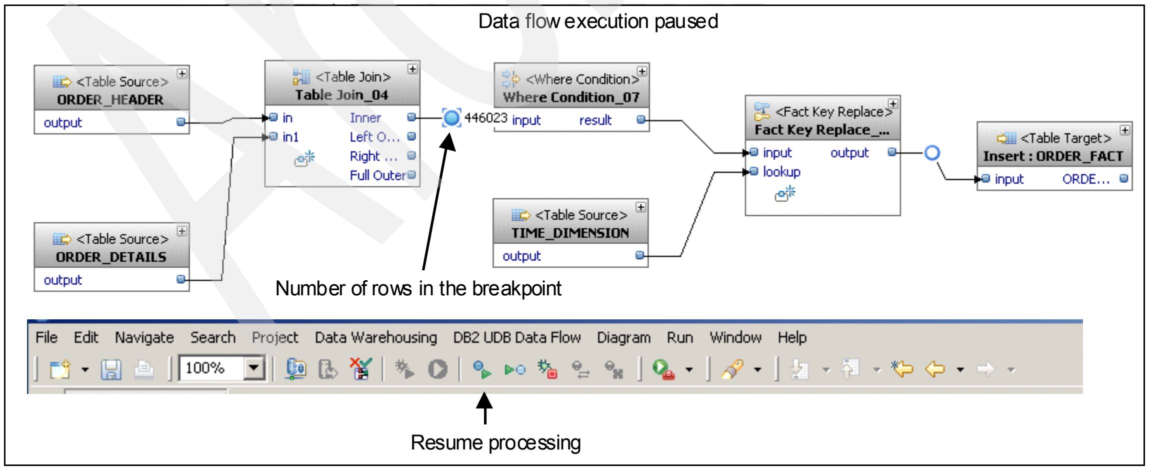


<Table Source> ORDER_HEADER
output
<Table Source> ORDER_DETAILS
output
<Table Join> Table Join_04
in
in1
Inner
Left O…
Right …
Full Outer
<Where Condition> Where Condition_07
input
result
<Table Source> TIME_DIMENSION
output
<Fact Key Replace> Fact Key Replace_...
input
lookup
output
<Table Target> Insert : ORDER_FACT
input
ORDE…
Breakpoints

Figure 5-26 Setting data flow breakpoints

Execute the data flow in debug mode by selecting **Debug data flow** from the data flow menu. The Flow Execution dialog box opens as when executing the data flow, except that the **Execution** button is replaced with a **Debug** button. Set the run parameters as before or use a saved run profile and click **Debug**.

The flow executes until it reaches the breakpoint. The number of rows in the breakpoint table is displayed (Figure 5-27). You can see that 446,023 rows flow through this connector. We could stop and investigate this table, but we shall continue processing by clicking the **Resume processing** icon in the menu bar.

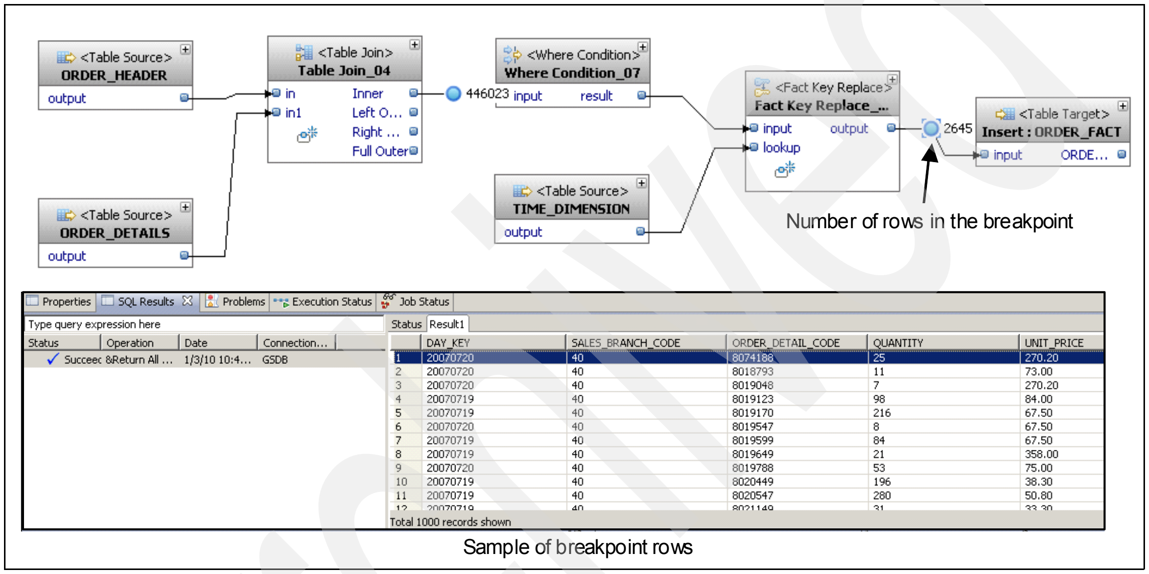


Data flow execution paused
<Table Source> ORDER_HEADER
output
<Table Source> ORDER_DETAILS
output
<Table Join> Table Join_04
in
in1
Inner
Left O…
Right …
Full Outer
446023
<Where Condition> Where Condition_07
input
result
<Table Source> TIME_DIMENSION
output
<Fact Key Replace> Fact Key Replace_...
input
lookup
output
<Table Target> Insert : ORDER_FACT
input
ORDE…
Number of rows in the breakpoint
File Edit Navigate Search Project Data Warehousing DB2 UDB Data Flow Diagram Run Window Help
100%
Resume processing

Figure 5-27 Data flow breakpoint suspended

Processing continues, but suspends at the next breakpoint. It shows that 2,645 rows flow through this final connector to be inserted into the target table. A sample of the rows in the breakpoint can be obtained by right-clicking the breakpoint and selecting **Sample breakpoint rows** from the context menu. The results are shown in the SQL results view.

The sample returned 1000 rows, as shown in Figure 5-28. Scrolling through the results does not show any NULLS in the non-null columns. Further investigation is needed. We do not need to finish the flow, because we know it will fail, so cancel the execution by clicking the **Cancel debug** icon in the menu bar.



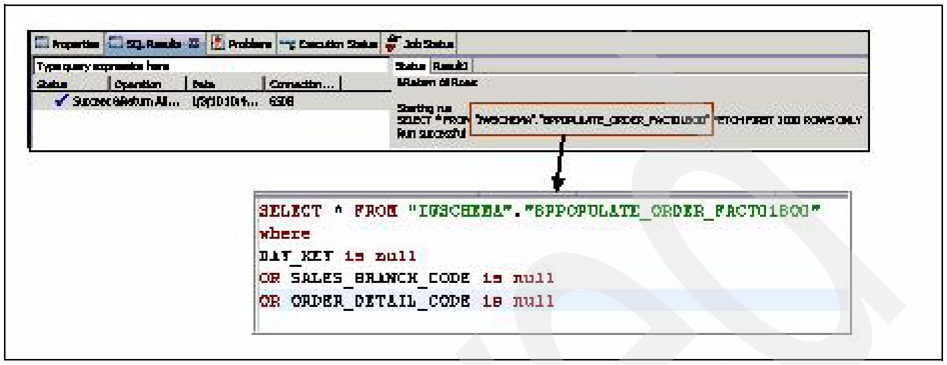
Status	Operation	Date	Connection...
✓ Succeec	&Return All ...	1/3/10 10:4...	GSDB

	DAY_KEY	SALES_BRANCH_CODE	ORDER_DETAIL_CODE	QUANTITY	UNIT_PRICE
1	20070720	40	8074188	25	270.20
2	20070720	40	8018793	11	73.00
3	20070720	40	8019048	7	270.20
4	20070719	40	8019123	98	84.00
5	20070719	40	8019170	216	67.50
6	20070720	40	8019547	8	67.50
7	20070719	40	8019599	84	67.50
8	20070719	40	8019649	21	358.00
9	20070720	40	8019788	53	75.00
10	20070719	40	8020449	196	38.30
11	20070719	40	8020547	280	50.80
12	20070719	40	8021149	31	33.30

Total 1000 records shown

Figure 5-28 Second data flow breakpoint

The tables created for the breakpoints still exist and can be queried with SQL. However, we need to know the name that can be found in the SQL execution results view. We can use that name in a SQL script to see if there are any NULLS in the non-null columns in our result, shown in Figure 5-29.



Status	Operation	Date	Connection...
✓ Succeeded	Return All...	1/3/10 10:4...	6308

```
Return All Rows

Starting run
SELECT * FROM "IWSCHEMA"."BPPOPULATE_ORDER_FACT01B00" FETCH FIRST 1000 ROWS ONLY
Run successful
```

```
SELECT * FROM "IWSCHEMA"."BPPOPULATE_ORDER_FACT01B00"
where
DAY_KEY is null
OR SALES_BRANCH_CODE is null
OR ORDER_DETAIL_CODE is null
```

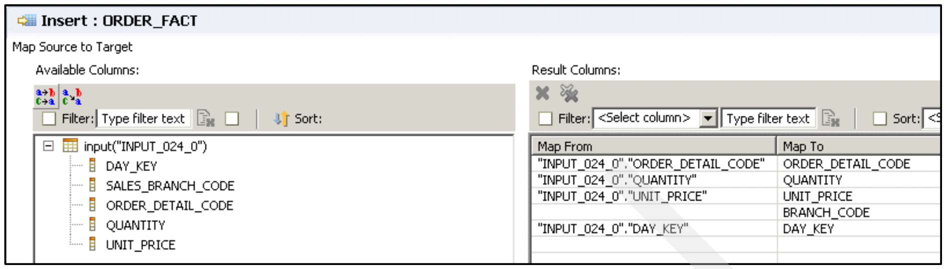
Figure 5-29 Running a script against a breakpoint table

There are no rows returned so we know that there are no NULLS in the data, which seems to be correct at this point.

Take a closer look at the table target operator. Examining the property pages of the target table operator, notice in the Map page that there is an empty column mapping, as seen in Figure 5-30 on page 111. When a connector is linked to the input port of a table target operator, Design Studio automatically maps the columns of the input virtual table to the columns in the target table by name. If all names match, all columns are mapped. In our case, the virtual table column SALES_BRANCH_CODE does not have a target table column of the same name. Therefore, it was not mapped.

This is not strictly an error in that all columns do not have to be mapped in an insert operation. This column should be mapped to BRANCH_CODE in the target table. When data is inserted and there is no mapping, a NULL is assumed if no default value is specified and the column is not defined as NOT NULL.

When we added the target table operator, we should have checked the mapping property page to ensure that input columns were mapped to the target table columns appropriately. We did not do this in this example intentionally, so we could have an error to guide the discussion.

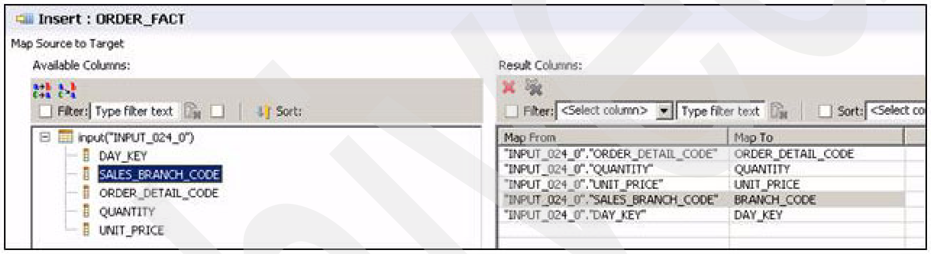


input("INPUT_024_0")
- DAY_KEY
- SALES_BRANCH_CODE
- ORDER_DETAIL_CODE
- QUANTITY
- UNIT_PRICE

Map From	Map To
"INPUT_024_0"."ORDER_DETAIL_CODE"	ORDER_DETAIL_CODE
"INPUT_024_0"."QUANTITY"	QUANTITY
"INPUT_024_0"."UNIT_PRICE"	UNIT_PRICE
	BRANCH_CODE
"INPUT_024_0"."DAY_KEY"	DAY_KEY

Figure 5-30 Mapping missing in target table operator

To correct this, we drag the SALES_BRANCH_CODE column name from the left to the BRANCH_CODE row on the right, as shown in Figure 5-31.



input("INPUT_024_0")
- DAY_KEY
- SALES_BRANCH_CODE
- ORDER_DETAIL_CODE
- QUANTITY
- UNIT_PRICE

Map From	Map To
"INPUT_024_0"."ORDER_DETAIL_CODE"	ORDER_DETAIL_CODE
"INPUT_024_0"."QUANTITY"	QUANTITY
"INPUT_024_0"."UNIT_PRICE"	UNIT_PRICE
"INPUT_024_0"."SALES_BRANCH_CODE"	BRANCH_CODE
"INPUT_024_0"."DAY_KEY"	DAY_KEY

Figure 5-31 Corrected target table mapping

We validate and re-test the flow, which executes successfully now. Check the values in the output table by sampling the data from the target table operator context menu.

5.2.5 Code generation

The data flow is a logical flow model that represents the movement and transformation activities needed to accomplish the intended tasks. From this flow model comes DB2-specific SQL code. That code is generated as part of the test execution process and as part of the deployment preparation process.

The generated SQL is embedded into an internal script-like control language that the runtime server uses to control the execution of the SQL. This is called the execution plan graph (EPG). You can explicitly generate and view the EPG.

However, sometimes the data flow developer wants to view the generated SQL without the control language. This could be done for various reasons, such as testing the code as a DB2 script.

The data flow menu has two options related to generating code. One is **Generate Execution Plan,** which generates the execution plan graph and opens it in a Design Studio text editor. The other is **Generate SQL Code,** which generates the execution plan graph and opens it in a Design Studio text editor, It also extracts the SQL code and opens it in another Design Studio text editor.

Example 5-1 shows the data flow, generated as an EPG. Although the SQW runtime engine can use this to execute the data flow, it is difficult for humans to read. This is a simple flow and the resulting EPG is fairly simple, but an EPG can get complex for other flows. Even simple flows can generate complex EPGs.

Example 5-1 Generated EPG

```
DEPLOYMENT : Graph ( ) : type GRAPH : node
{
   PREP ( ) : type BLOCK : node /graph9
   {
   }
}
CATCH_BLOCK ( ) : type BLOCK : node /graph7
{
}
FINALLY_BLOCK ( ) : type BLOCK : node /graph8
{
}

RUNTIME : Graph ( ) : type GRAPH : node
{
   PREP ( ) : type BLOCK : node /graph9
   {
   }
   EPGTXN ( ) : type TXN : node /graph12
      (db connection =  [GSDB])
   {
      ( ):CODE_UNIT, node /graph12/node9;


      CODE_UNIT:JAVA
      (Database Name = GSDB

      Schema Name = IWSCHEMA${RESERVED/RUN_ID}

      Action = CREATE

      )
      {
```

```
      Java Runtime Unit Class:
com.ibm.datatools.etl.dataflow.baselib.runtimeunits.JDBCSchema
      }
   }
   CATCH_BLOCK ( ) : type BLOCK : node /graph12/graph7
   {
   }
   FINALLY_BLOCK ( ) : type BLOCK : node /graph12/graph8
   {
   }
   EPGTXN ( ) : type TXN : node /graph15
      (db connection =  [GSDB])
   {
      ( ):CODE_UNIT, node /graph15/node9;


      CODE_UNIT:JDBC
      (Database Connection 0 = GSDB

      )
      {
      SET CURRENT SCHEMA "IWSCHEMA${RESERVED/RUN_ID}"


      }
   }
   CATCH_BLOCK ( ) : type BLOCK : node /graph15/graph7
   {
   }
   FINALLY_BLOCK ( ) : type BLOCK : node /graph15/graph8
   {
   }
   EPGTXN ( ) : type TXN : node /graph18
      (db connection =  [GSDB])
   {
      ( ):CODE_UNIT, node /graph18/node9;


      CODE_UNIT:JDBC
      (Database Connection 0 = GSDB

      OperatorIDs =
com.ibm.datatools.etl.common.impl.StringListImpl@777d777d (content:
[01, 02, 024, 026, 017, 018, 07]))
      {
```

```
      INSERT INTO GOSALES.ORDER_FACT (ORDER_DETAIL_CODE, QUANTITY,
UNIT_PRICE,
        BRANCH_CODE, DAY_KEY)
        SELECT Q2246.ORDER_DETAIL_CODE AS ORDER_DETAIL_CODE,
               Q2246.QUANTITY AS QUANTITY,
               Q2246.UNIT_PRICE AS UNIT_PRICE,
               Q2235.SALES_BRANCH_CODE AS SALES_BRANCH_CODE,
               Q1960.MAXSK AS DAY_KEY
          FROM GOSALES.ORDER_HEADER Q2235, GOSALES.ORDER_DETAILS Q2246,
(
              SELECT MAX(DAY_KEY) AS MAXSK,
                     DAY_DATE AS DAY_DATE
                FROM (SELECT DAY_DATE AS DAY_DATE,
                             DAY_KEY AS DAY_KEY
                        FROM GOSALES.TIME_DIMENSION Q2224) Q1884
                GROUP BY DAY_DATE) Q1960
          WHERE (Q2235.ORDER_DATE = Q1960.DAY_DATE) AND
(Q2235.ORDER_NUMBER = Q2246.ORDER_NUMBER)
          AND (Q2235.ORDER_DATE >= DATE('${SQW_Tutorial/StartDate}'))
          AND (Q2235.ORDER_DATE <= DATE('${SQW_Tutorial/EndDate}'))
          AND (Q1960.DAY_DATE >= DATE('${SQW_Tutorial/StartDate}'))
          AND (Q1960.DAY_DATE <= DATE('${SQW_Tutorial/EndDate}'))


      }
   }
   CATCH_BLOCK ( ) : type BLOCK : node /graph18/graph7
   {
   }
   FINALLY_BLOCK ( ) : type BLOCK : node /graph18/graph8
   {
   }
}
CATCH_BLOCK ( ) : type BLOCK : node /graph7
{
}
FINALLY_BLOCK ( ) : type BLOCK : node /graph8
{
   EPGTXN ( ) : type TXN : node /graph8/graph7
      (db connection =  [GSDB])
   {
      ( ):CODE_UNIT, node /graph8/graph7/node9;


      CODE_UNIT:JAVA
      (Database Name = GSDB

      Schema Name = IWSCHEMA${RESERVED/RUN_ID}

      Action = DROP

      )
      {
      Java Runtime Unit Class:
com.ibm.datatools.etl.dataflow.baselib.runtimeunits.JDBCSchema
      }
   }
   CATCH_BLOCK ( ) : type BLOCK : node /graph8/graph7/graph7
   {
      EPGJump  : type CONTINUE
   }
   FINALLY_BLOCK ( ) : type BLOCK : node /graph8/graph7/graph8
   {
   }
}

UNDEPLOYMENT : Graph ( ) : type GRAPH : node
{
   PREP ( ) : type BLOCK : node /graph9
   {
   }
}
CATCH_BLOCK ( ) : type BLOCK : node /graph7
{
}
FINALLY_BLOCK ( ) : type BLOCK : node /graph8
{
}
```

Example 5-2 shows the generated SQL only. This data flow results in one SQL statement even though there are seven operators in the flow. The SQL statement does an INSERT INTO SELECT FROM statement that is passed to DB2 for processing. The data does not move outside of the DB2 database engine.

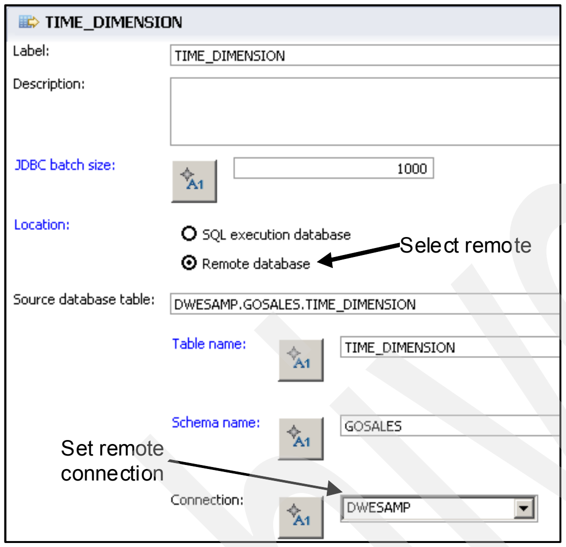
Example 5-2 Generate SQL code

```
-- <ScriptOptions statementTerminator="@" />

SET CURRENT SCHEMA "IWSCHEMA" @

INSERT INTO GOSALES.ORDER_FACT (ORDER_DETAIL_CODE, QUANTITY,
UNIT_PRICE,
  BRANCH_CODE, DAY_KEY)
  SELECT Q2246.ORDER_DETAIL_CODE AS ORDER_DETAIL_CODE,
         Q2246.QUANTITY AS QUANTITY,
         Q2246.UNIT_PRICE AS UNIT_PRICE,
         Q2235.SALES_BRANCH_CODE AS SALES_BRANCH_CODE,
         Q1960.MAXSK AS DAY_KEY
    FROM GOSALES.ORDER_HEADER Q2235, GOSALES.ORDER_DETAILS Q2246, (
        SELECT MAX(DAY_KEY) AS MAXSK,
               DAY_DATE AS DAY_DATE
          FROM (SELECT DAY_DATE AS DAY_DATE,
                       DAY_KEY AS DAY_KEY
                  FROM GOSALES.TIME_DIMENSION Q2224) Q1884
          GROUP BY DAY_DATE) Q1960
    WHERE (Q2235.ORDER_DATE = Q1960.DAY_DATE) AND (Q2235.ORDER_NUMBER =
Q2246.ORDER_NUMBER)
    AND (Q2235.ORDER_DATE >= DATE('1999-01-01'))
    AND (Q2235.ORDER_DATE <= DATE('1999-01-01'))
    AND (Q1960.DAY_DATE >= DATE('1999-01-01'))
    AND (Q1960.DAY_DATE <= DATE('1999-01-01')) @
```

Most data flows are not necessarily this simple and might result in multiple SQL statements being generated and perhaps the use of staging tables. If the code generator determines that a staging table is needed, it generates the appropriate code. Developers do not have to be concerned with creating these staging tables.

There are a number of factors that influence code generation. Factors include remote tables, multiple connectors for an output port, certain operators and so on. To illustrate this, let us modify the TIME_DIMENSION source table in our data flow to be a remote table and see how that affects the generated code.

It is a simple modification to change a local table source into a remote table source. We keep it simple by not including any filtering or select list processing. In the TIME_DIMENSION table source operator properties, select the **Remote database** radio button for the location option. A new property is added to the page to set the remote database connection. This results in the property setting shown in Figure 5-32.



TIME_DIMENSION

Label:	TIME_DIMENSION
Description:	
JDBC batch size:	1000
Location:	SQL execution database [✓ Remote database]
Source database table:	DWESAMP.GOSALES.TIME_DIMENSION
Table name:	TIME_DIMENSION
Schema name:	GOSALES
Connection:	DWESAMP

Figure 5-32 Defining TIME_DIMENSION as a remote table

Example 5-3 on page 118 shows the generated SQL code for the modified data flow. Because the TIME_DIMENSION table is in a remote database, the data needs to be staged to the execution database so it can be processed. However, in the data flow, we did not have to worry about the details of staging the data. We specified that the table is remote in the properties. The first part of the generated code handles the staging of the remote table by generating a global temporary table in the execution database. It then executes a Java program that connects to the remote database through JDBC, selects the data from the remote table and to the execution database to insert the data into the temporary table. The temporary table is used in the generated INSERT INTO SELECT FROM SQL that processes the data.

Example 5-3 Generated SQL code for the modified data flow

```
-- <ScriptOptions statementTerminator="@" />

SET CURRENT SCHEMA "IWSCHEMA" @

DECLARE GLOBAL TEMPORARY TABLE "IWTEMP02311"(
  MAXSK INTEGER,
  DAY_DATE TIMESTAMP)
 ON COMMIT PRESERVE ROWS NOT LOGGED @

---- BEGIN NON-SQL CODE ----

-- JAVA

-- Java Runtime Unit Class:
com.ibm.datatools.etl.dataflow.baselib.runtimeunits.JDBCInsert

-- Source Database Name = DWESAMP
--
--Source SQL Query = SELECT MAXSK AS MAXSK,
--       DAY_DATE AS DAY_DATE
--  FROM (SELECT MAX(DAY_KEY) AS MAXSK,
--               DAY_DATE AS DAY_DATE
--          FROM (SELECT DAY_DATE AS DAY_DATE,
--                       DAY_KEY AS DAY_KEY
--                  FROM GOSALES.TIME_DIMENSION Q2224) Q1884
--          GROUP BY DAY_DATE) Q2312
--  WHERE (DAY_DATE >= DATE('${SQW_Tutorial/StartDate}'))
--    AND (DAY_DATE <= DATE('${SQW_Tutorial/EndDate}'))
--
--
--Target Database Name = GSDB
--
--Target Insert Statement = INSERT INTO
SESSION."IWTEMP${RESERVED/RUN_ID}02311"(MAXSK, DAY_DATE) VALUES (?, ?)
--
--NumTargetColumns = 2
--
--CommitInterval = 1000
--
--BatchSize = 1000
--
--Duplicate Connection On = TARGET
--
```

```
--Number of Database Connection(s) = 2
--
--Database Connection 0 = DWESAMP
--
--Database Connection 1 = GSDB
--
--OperatorIDs =
com.ibm.datatools.etl.common.impl.StringListImpl@550b550b (content:
[017, 018])

---- END NON-SQL CODE ----

INSERT INTO GOSALES.ORDER_FACT (ORDER_DETAIL_CODE, QUANTITY,
UNIT_PRICE,
  BRANCH_CODE, DAY_KEY)
  SELECT Q2246.ORDER_DETAIL_CODE AS ORDER_DETAIL_CODE,
         Q2246.QUANTITY AS QUANTITY,
         Q2246.UNIT_PRICE AS UNIT_PRICE,
         Q2235.SALES_BRANCH_CODE AS SALES_BRANCH_CODE,
         Q1960.MAXSK AS DAY_KEY
    FROM GOSALES.ORDER_HEADER Q2235, GOSALES.ORDER_DETAILS Q2246,
         SESSION."IWTEMP02311" Q1960
    WHERE (Q2235.ORDER_DATE = Q1960.DAY_DATE) AND (Q2235.ORDER_NUMBER =
Q2246.ORDER_NUMBER)
    AND (Q2235.ORDER_DATE >= DATE('1999-01-01'))
    AND (Q2235.ORDER_DATE <= DATE('1999-01-01')) @

DROP TABLE SESSION."IWTEMP02311" @
```

The code generation process can also generate intermediate staging tables in the data flow if needed. This is done automatically, allowing you to focus on the transformation logic.

5.2.6 data flow operators

In this chapter we have discussed data flows in a generic sense without regard to InfoSphere Warehouse platform, distributed (LUW), or System z. The development methodology is the same across both platforms and many operators are common across the two products. However, a data flow for the LUW product is different from a data flow for the System z product due to the nature of each platform.

See the IBM DB2 Information Center for platform for information about the operators.

Figure 5-33 on page 123 shows the operators for InfoSphere Warehouse V9.7 for the Linux, UNIX and Windows (LUW) platform and Figure 5-34 on page 125 shows the data flow operators for InfoSphere Warehouse on System z V9.5.2.

See the DB2 Information Center for your platform for information about the operators, using the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp

Common data flow operators

There is a lot of commonality between SQW operators of the LUW and System z product platforms. In fact, there are many operators in common. However, while performing the same basic function, those common operators have differences in properties specific to each platform.

Source operators

Table 5-1 details source operators.

Table 5-1 Source operators

Source operator	Description
Table source	The table source operator extract data from a relational table. The table might be local to the execution database or might be in a remote DB2 or non-DB2 database. Most important to remote tables is the ability to push row and column filtering and derived column functions to the remote source database for evaluation. This reduces the amount of data transferred over the network.
SQL query source	The SQL query source operator extracts data from a relational table using a user-defined SQL statement. The table might be local or remote. The operator does not use table metadata from a data model. The result column metadata must be provided in the operator properties. When defining the operator, Design Studio tries to perform a DESCRIBE operation to obtain the result column metadata which, if successful, populates the metadata in the operator. Therefore, a live connection to the source database is required. If the DESCRIBE fails, the user must define the column metadata manually.
File import	A file import operator extracts data from flat files and makes it available to the data flow. The output from the file import operator can be connected directly to relational operators such as join. Code generation generates the code to load the data into a staging table using the DB2 bulk load utility appropriate for the target DB2 platform.

Transformation operators

Table 5-2 details transformation operators.

Table 5-2 Transformation operators

Operator	**Description**
Where condition	This operator provides a basic filtering mechanism similar to a where clause in an SQL statement. The where condition applies the defined condition to the incoming data, passing through only rows that match. Unmatched rows are ignored.
Splitter	The splitter operator is similar in concept to the filter in that you can apply conditions to the incoming data rows. However, the splitter allows multiple outputs with an independent condition for each output to allow splitting the data into multiple streams. The outputs are not exclusive. In other words, a row might satisfy multiple conditions and be present in more than one output.
Distinct	This operator removes duplicate rows from the incoming data based on a defined set of columns. When duplicate rows are found, only the first row that is read during processing is retained. All other duplicate rows are sent to the discard port.
Order by	The order by operator performs a sort function, ordering the data in a sequence defined in the properties based on a set of user-selected columns. This is equivalent to the DB2 order by clause.
Group by	The group by operator performs aggregations on the incoming data based on DB2 aggregate column functions (such as Min, Max, Count, Sum, Average, and so forth).
Table Join	This operator matches rows from two or more related tables according to defined conditions to create a singe output with columns selected from across the tables. Table join supports inner join, left outer join, right outer join and full outer join. Of course, cross joins are also supported by not including a join condition.
Union	Union operators combine two data sets that have column schemas that match, and have compatible data types, into a single data set. The new set of rows is created according to one of three rules: UNION, INTERSECT, or EXCEPT.
Select List	The select list operator allows you to add, remove, or modify the columns in your data set at any point in a data flow, using column expressions based on arithmetic functions, scalar functions, and constant values. In addition, any DB2 user-defined scalar function (UDF) might also be used as custom transformation logic. Other operators (such as group by, join, splitter and key lookup) can also perform the same tasks as the select list operator.
Fact key replace	A fact key replace operator allows you to replace natural keys in a data warehouse fact table with surrogate keys from dimension, or key lookup, tables.
Key lookup	The key lookup operator compares keys from a single input table with keys in one or more lookup tables and discards input table rows that do not have matching rows in the lookup tables.

Operator	Description
Pivot	Pivot operators group data from several related columns into a single column by creating additional rows. You can use a pivot operator when a number of column headings are related closely enough that they make sense as values in a single column. The result table contains fewer columns and more rows than the input table.
Unpivot	You can rearrange a data set under a new set of columns that are created from repeating values in a single source column. Use the unpivot operator properties to choose a column to create the new headings, data to be placed in the new columns, and columns that will become keys.
Sequence	A sequence operator is used to generate data values for a column, based on an existing DB2 sequence object. The sequence object needs to exist as an object in the physical data model.
Data station	The data station operator represents a defined staging point in a data flow. While code generation automatically stages data as needed by the flow, you might want to store intermediate processed data for the purpose of tracking, debugging, or ease of data recovery. A data station can also act as a target operator.
DB2 table function	The DB2 table function operator invokes predefined DB2 table functions. A table function is a logical database object that returns a table (one or more rows) based on a statement expressed in a specific language (such as SQL, C, or Java). The returned table is passed into the data flow through the output operator. DB2 table function operators might also be used as data sources as well. A table function could be written to read, using a C API, for example, a complex or proprietary data source and pass the resulting data into the data flow. DB2 table function operators are powerful and flexible data transformation tools.

Target Operators

Table 5-3 details target operators.

Table 5-3 Target operators

Target operator	Description
Table Target	The Table Target operator populates a relational database table with data that has passed through a data flow. performing one of three SQL operations: Insert, Update or Delete. The table can be either local or remote.
Bulk Load	Each platform has an operator that will use the DB2 bulk load utility to load the data coming in from the data flow. On the LUW platform the operator is called
Export	This is called File Export on LUW and Data Set Export on System z. They both will export data from the data flow into flat files appropriate to the platform.
Custom SQL	The Custom SQL operator allow you to develop one or more custom SQL statements that modify the database. The operator can function as a standalone operator or as a target operator in a data flow.

LUW operators

In addition to the common data flow operators, the LUW platform has additional SQW Operators, XML Operators, Text Analysis Operators, and Data Mining Operators. XML support is new in V9.7. Text Analysis and Mining Operators are covered in other chapters in this book. See Figure 5-33.

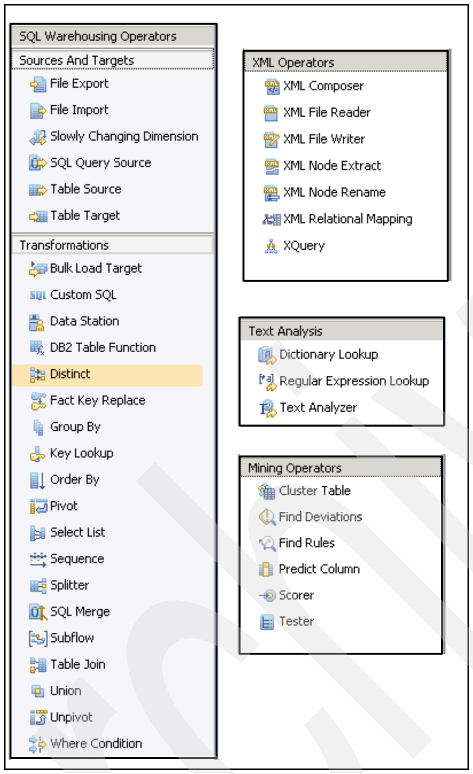


Figure 5-33 Data flow operators for InfoSphere Warehouse V9.7 for LUW

Additional SQW and XML operators

The LUW platform has two additional SQW operators and a set of XML operators.that are not yet in the System z. See Table 5-4.

Table 5-4 Additional SQW and XML operators

Operator	**Description**
SQL Merge	The SQL Merge operator is a target operator that updates the target table using a combination of SQL INSERT and UPDATE operations. This is commonly referred to as an UPSERT.
Slowly Changing Dimension (SCD)	The SCD operator is a target operator that updates the target table using Type 1 or Type 2 dimension maintenance strategies.
XML File Reader	This source operator retrieves data from an XML file or from specific nodes of an input XML file and inserts the data of these retrieved nodes into an XML column of a table row.
XML Node Extract	The XML node extract operator is a transform operator that you can use to extract one or more nodes from an XML column.
XML Node Rename	The XML node rename operator is a transform operator that is used to rename a node in an XML column of a source table, and which then flows into the output target table.
XQuery	The XQuery operator retrieves data from one or more XML columns of a table and places the resulting XML data in an XML column using the DB2 XMLQuery function to run the XQuery.
XML Relational Mapping	The XML relational mapping operator retrieves data from an XML column and maps the resulting XML data to the columns of a relational table decomposing the XML structure into individual data elements that mapped to independent columns.
XML Composer	The XML composer operator converts relational data to XML data and saves the result in one or multiple XML documents.
XML File Writer	The XML file writer operator is a data flow operator that you can use to retrieve data from an XML column, and write the data into one XML file or into separate XML files.

System z operators

The System z data flow operators are depicted in Figure 5-34. At this time there are no additional System z operators to discuss, but we expect more to be added over time.

Data flows might result in DB2 utilities being invoked. All utilities are invoked using the DSNUTILU stored procedure. Therefore, this stored procedure must be installed and configured correctly prior to working with SQW.

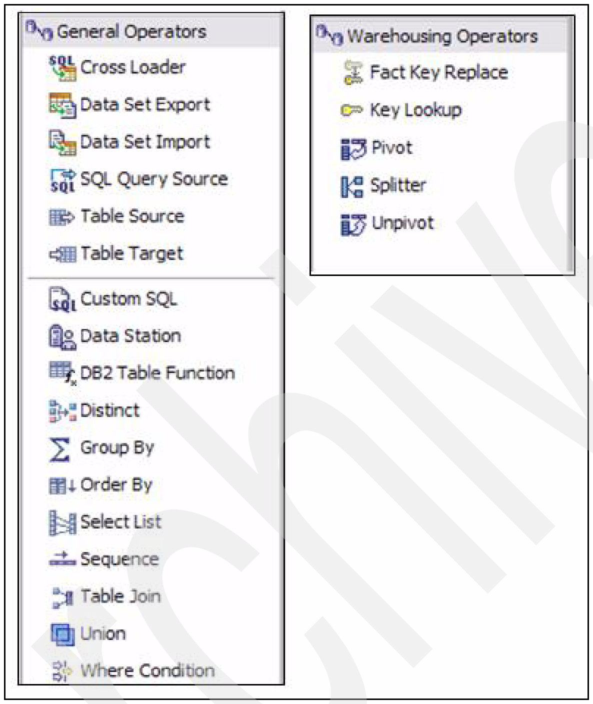


Figure 5-34 data flow operators for InfoSphere Warehouse on System z V9.5.2

5.3 Developing control flows

Control flows were introduced in “Control flow and subprocesses” on page 81. A control is a set of operations that accomplish the following tasks:

- Invoke activities of various technologies
- Sequence activities into a defined flow
- Defines how control is passed between activities based on defined conditions

Control flows, and subprocesses, are developed using the Design Studio graphical control flow editor to create a flow model representing the activities and flow of control. Activities represent work that is to be accomplished at a server or a flow control decision point. Activities are arranged in sequences that define in what order and under what conditions the activities are executed.

From this model, Design Studio generates code that manages the execution of activities in the correct sequence dynamically based on the defined conditions.

In 5.3.1, “Control flow basics” on page 126 we demonstrate how to develop a control flow and test the execution in Design Studio.

5.3.1 Control flow basics

When developing a control flow, you are actually developing a flow model. You define discrete work activities, where those activities execute, in what order, and under what conditions.

Operators, connectors, and ports

In general, control flow operators fall into two categories:

- Task activities

  Task activities are operators that cause a work task to happen.

- Flow control activities

  Flow control activities are operators that manage the flow of control.

Connectors between operator ports define the path to take after the activity completes following the appropriate connector to the next activity.

Figure 5-35 on page 127 shows a simple control flow. This flow contains four task operators, which cause work to happen and two flow control operators. The start operator is always present in a control flow and is the point where execution starts. The end operator, End_08, is an explicit terminal point ending a flow branch. These are flow control operators. There are two operators, data flow_02 and data flow_03, that each invoke a previously developed data flow. There are also two email operators, Email_06 and Email_010, that send a status message through the email network.

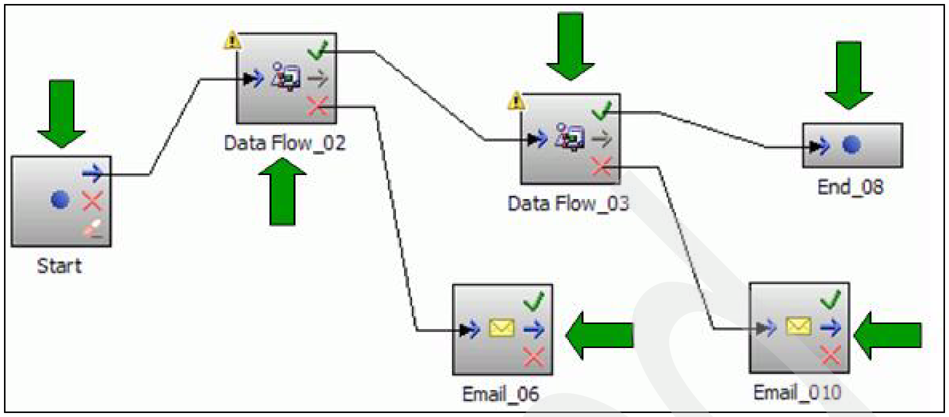


Figure 5-35 A simple control flow: Operators

Operators are sequenced together by drawing connectors between output and input ports of operators. These connectors define what operator to execute after the completion of an activity, as shown in Figure 5-36.

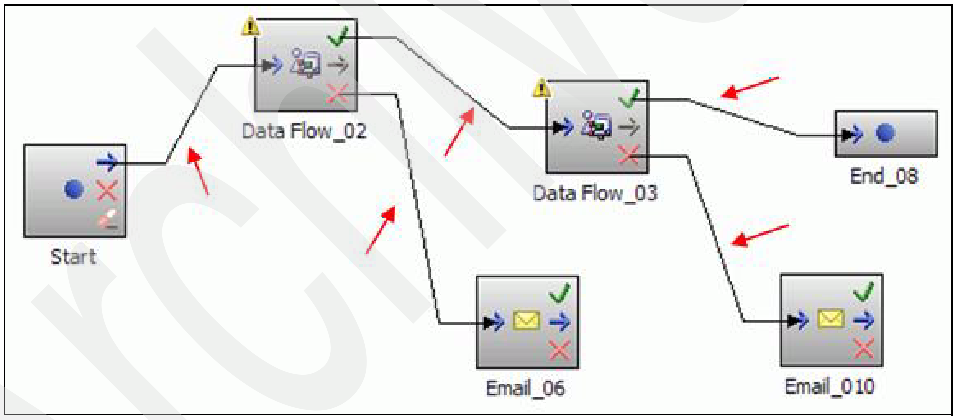


Figure 5-36 A simple control flow: Connectors

Control flow operators also have ports that define the entry and exit points of the operator. The output ports, circled in Figure 5-37 on page 128, define which connector to follow based on the completion status of the current operator. Output ports are connected to the input of the next operator to be executed, if this port is selected. Output ports are selected based on the completion status of the operator. Output ports are optional.

The On-Success port is the top output port with a green check icon and defines the flow branch to follow if the operator completes successfully. The On-Failure port is the bottom output port with a red ‘X’ icon and defines the flow branch to follow if the operator completes with a failure status. One or both of these might be connected. The middle output with the blue arrow icon is the Unconditional port, the branch that is taken regardless of the completion status of the operator. There cannot be a connection to the Unconditional port if there is a connection to either the On-Success or the On-Failure port.

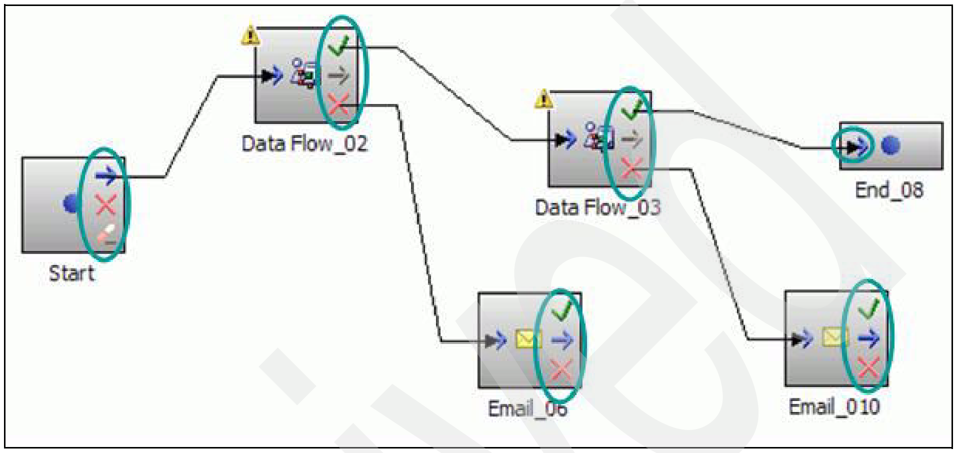
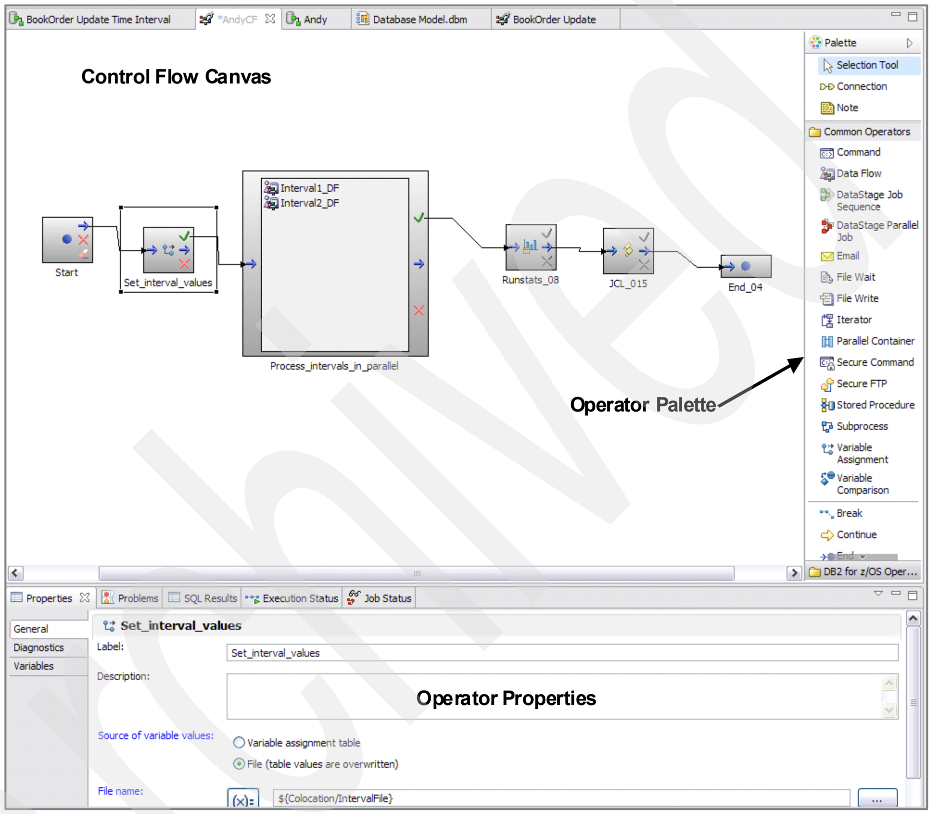


Figure 5-37 A simple control flow: Ports

The output ports of the start operator have different purposes from the output ports of the other operators. These ports define process level branching. The top port of the start operator, the blue arrow, defines the starting point of the flow with the operator connect to this port executing first. This is called the Start port. The middle port, identified by a red X, is the On-Failure port, which defines the start of the process-level error branch of the control flow. If there is an error in any operator in the control flow, the operators connected to the On-failure port are run. Activity-level error branches are run before the process-level error path. The bottom port is the Cleanup port, which defines the sequence of operators that are run after completing all other main or error branches.

Control flow editor

In Design Studio, control flows are developed using a number of common Design Studio functions (such as the Data Project Explorer), views (such as Properties, Data Output, and Problems), and, optionally, the Database Explorer. However, the actual creation and editing of data flows occurs in a specific graphical editor called the control flow editor, shown in Figure 5-38.



Set_interval_values

Label: Set_interval_values

Description:

Source of variable values: ( ) Variable assignment table [✓ (•) File (table values are overwritten)]

File name: ${Colocation/IntervalFile}

Figure 5-38 Control flow editor

As with most graphical editors, the control flow editor follows the drag, drop, connect, and set properties paradigm of development. When you create a new control flow or open an existing control flow, the control flow editor opens in a new tab in the editor area of Design Studio. The control flow editor consists of a canvas onto which graphical elements are drawn and an operator palette that contains all of the graphical elements that are relevant to building a control flow. In addition, you use the common Properties View tab to define the characteristics of the objects on the canvas. As you validate and test run the control flows, you also use the Problems View tab and the Data Output tab.

Attention: Properties can be set two ways. The first way, shown in Figure 5-11, is to use the Properties View tab. The other way is to use the properties wizard associated with every operator. It shows the same property pages as in the Properties View except in a wizard-like fashion. The properties wizard is accessed from the context menu of each operator. Also, a few operators, open the wizard automatically when dropped onto the canvas. A few developers work with the Properties Views and others work with the properties wizard. It is said that there are always at least two ways to do thing in Design Studio.

Here are a few things to consider when developing data flows:

- Be familiar with the common functions in Design Studio, such as:
  - Working with perspectives
  - Working with views
  - Using the Data Project Explore
  - Using the Database Explorer
  - Dragging from the Data Project Explorer and palette to the canvas
- Orient the data flow from left to right. This makes a neater diagram because the output ports are on the right side of an operator and the input ports are on the left side.
- Work with just a few operators at a time.
- Operators have properties that must be defined.
- A control flow itself has properties to set logging and trace levels.

Variables in a control flow

Just as variables are important to data flows they are important to control flows as well. Many properties in control flows can use variables. Working with variables in control flow properties is the same as for data flows. This is discussed in “Variables in a data flow” on page 90.

Unlike variables in a data flow, variables in a control flow can be set or modified by operators in the control flow. They can be passed as input variables to data flows and subprocesses. For LUW, variables can be passed to mining flows as well. Subprocesses can modify variable values and pass back variables to the calling flow.

This capability allows you to perform activities such as reading in a number of variables from a flat file at the beginning of a control, and passing those variables to subsequent data flows.

5.3.2 Developing a simple control flow

This section looks at how to develop and test a simple control flow. This section is not intended to be a step-by-step tutorial but rather to give a feel for the development techniques. Step-by-step tutorials can be found in the InfoSphere Warehouse Information Center for your product platform.

Just as with the data flow, we need to have an understanding of what we need to accomplish in this control flow.

- The overall objective of this control is to update our star schema dimensions and fact table. We have two dimensions (TIME and BRANCH_LOCATION) and one fact table (ORDER_FACT). The following list details the SQW data flows that have been developed and that exist in our Design Studio data warehouse project:
  - populate_branch_location
  - populate_time_dimension
  - Populate_Orders_Fact

    The Populate_Orders_Fact data flow uses two variables to define the start and end dates for extracting from the source tables. The variables are provided before each run in a comma-delimited data file. The variables have to be read from the file and passed to the data flow.

    The two data flows for populating the dimensions, populate_branch_location and populate_time_dimension, have no sequencing dependencies but they must complete successfully before the fact table can be populated using the data flow Populate_Orders_Fact.
- To reduce overall processing time, process work concurrently when possible.
- After the fact table has been populated, runstats must be run to update the DB2 statistics.
- If anything fails, an email is sent to the application administrator.

Logical strategy

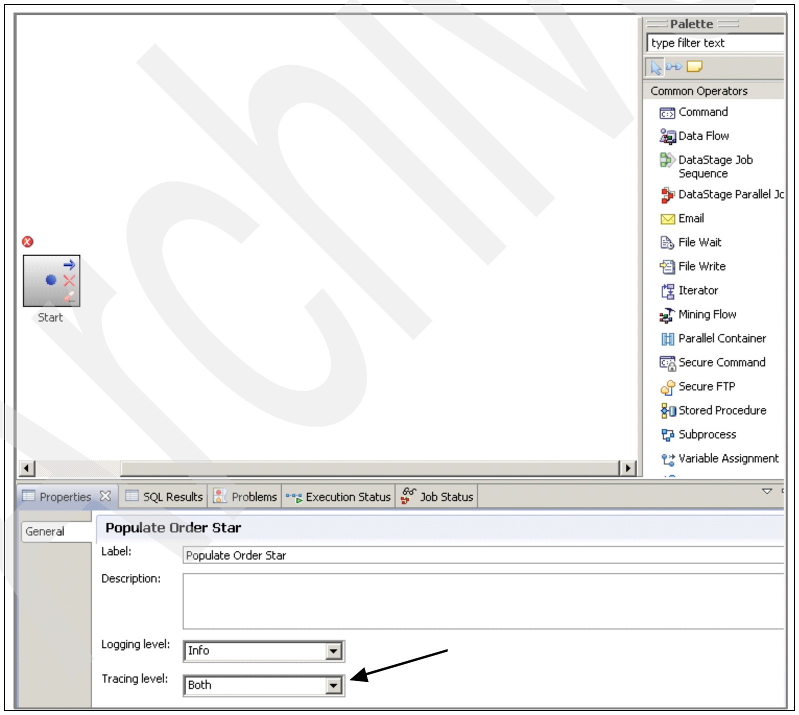
Looking at the requirements, we can come up with an approach to guide the control flow development. We are provided the two variables in a file, so we need to read that file and set those variables. However, this should be done first in the flow. Then, execute the two dimension data flows concurrently. After both finish successfully, we can execute the fact table data flow passing in the two variables. After the fact table is loaded, call the runstats utility. If anything fails anywhere in the flow we send an email.

Creating a new control flow

To create a new control flow, perform the following steps:

1. Open the data warehouse project (or create a new project), right-click the data flows folder and click **New** → **Control Flow**. This opens a new data flow editor similar to Figure 5-11 on page 91 with an empty canvas.
2. Define the database at which this data flow executes. This is called the execution database and is typically the target database. For a discussion of the execution database, see “Source, target, and execution databases” on page 85.

   The execution database is a property of the data flow itself. To see the properties of the data flow, click any white space on the data flow canvas. The data flow properties can be accessed in the Properties View tab.
3. Name the control flow `Populate Order Star`.
4. Set the SQL execution database as shown in Figure 5-39. Setting the tracing level property field to **Both** gives you the maximum amount of information in the run log, which is helpful if the flow fails and needs to be debugged.



Populate Order Star
Label: Populate Order Star
Description:
Logging level: Info
Tracing level: Both

Figure 5-39 New control flow and control flow properties

Setting the variables from file input

Perform the following steps to set the variables from file input:

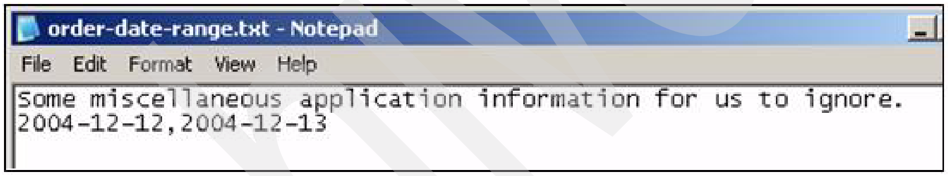
1. Set the two variables that define the date range for extracting orders. The values are provided in a file that can be read at runtime to set the variables. We can use the variable assignment operator for this task.

   The input file name will always be the same name, order-date-range.txt. The file will be in a path associated with the application. However, the exact path name has not been decided and, besides, it might change over time and between systems. Therefore, we can use a variable to hold the application path name. This path name will be set upon deployment to runtime.

   The values needed will be provided in the second record of the file. The first record is used for tracking and versioning information and can be ignored. The record will contain two dates in the following format, where the first date is the start date of the range and the second date is the end date of the range:

   ```
   YYYY-MM-DD,YYYY-MM-DD
   ```

   See the example in Figure 5-40.

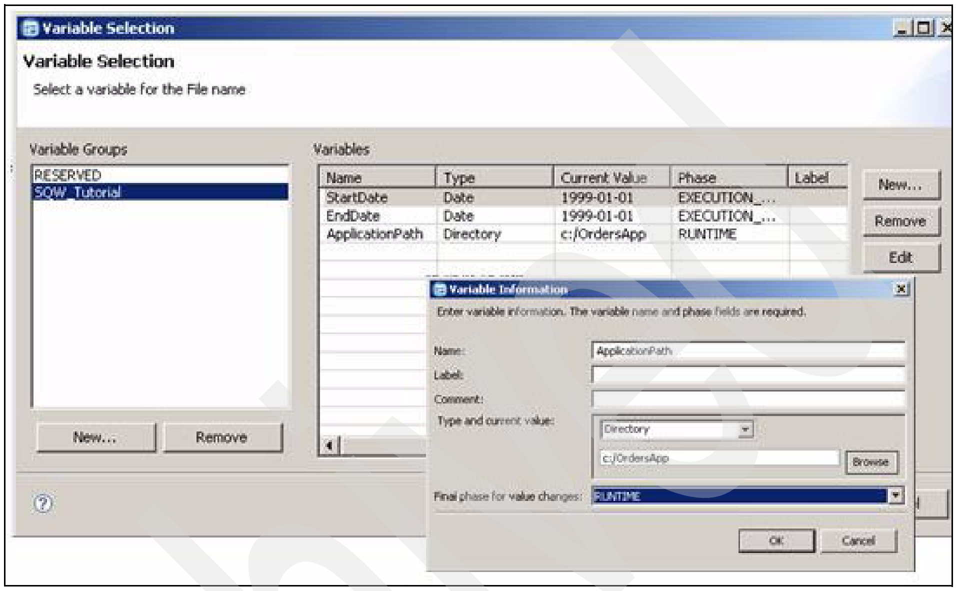


```
Some miscellaneous application information for us to ignore.
2004-12-12,2004-12-13
```

Figure 5-40 Sample order-date-range.txt file

2. Drop a variable assignment operator on the canvas, connect it to the Start port of the start operator and define the properties.

3. Define a new variable, ApplicationPath, to hold the application path and set it to a default value. This variable value would be set at deployment time but might still change from time to time, so the Phase property is set to RUNTIME. Figure 5-41 shows the creation of the ApplicationPath variable.



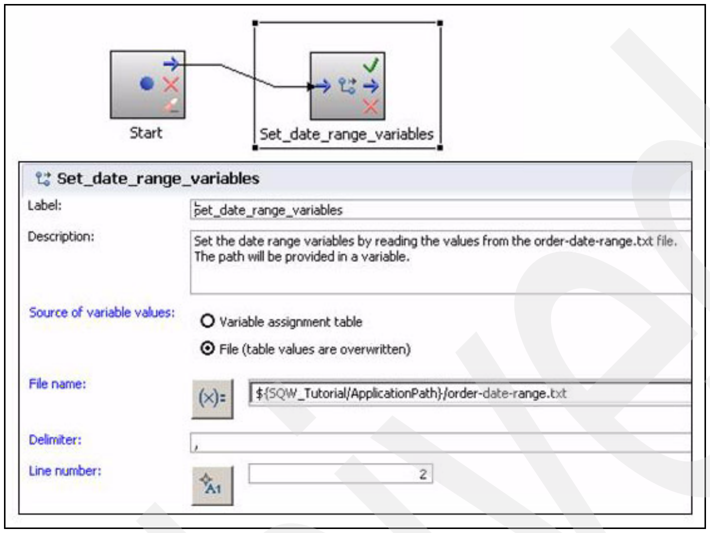
Name	Type	Current Value	Phase	Label
StartDate	Date	1999-01-01	EXECUTION_...	
EndDate	Date	1999-01-01	EXECUTION_...	
ApplicationPath	Directory	c:/OrdersApp	RUNTIME	

Enter variable information. The variable name and phase fields are required.

Name:	ApplicationPath
Label:	
Comment:	
Type and current value:	Directory c:/OrdersApp
Final phase for value changes:	RUNTIME

Figure 5-41 Creating the ApplicationPath variable

The input source file name (Figure 5-42) has been set to a concatenation of the ApplicationPath variable and the file name, the delimiter to a comma and the line number set to 2, which populates the values from the comma-delimited records on record 2 of the input file.



Set_date_range_variables

Label: Set_date_range_variables

Description: Set the date range variables by reading the values from the order-date-range.txt file. The path will be provided in a variable.

Source of variable values: Variable assignment table [✓ File (table values are overwritten)]

File name: ${SQW_Tutorial/ApplicationPath}/order-date-range.txt

Delimiter: ,

Line number: 2

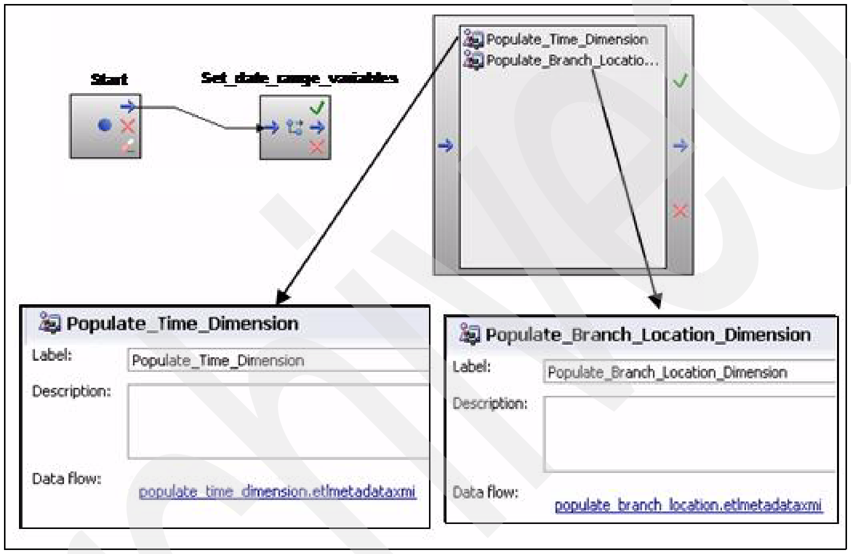
Figure 5-42   Defining the variable assignment operator

Populating the dimension tables

The next task is to populate the two dimension tables. The requirements state that two data flows that populate the dimensions are independent and have no other sequencing dependencies. The requirements also state to use concurrent processing whenever feasible. We should therefore execute the two dimension data flows concurrently. We can use the Parallel Container operator for executing activities concurrently by dropping other operators inside of the parallel container.

Another related requirement states that we can execute the fact data flow only after both the dimension data flows complete successfully. The Parallel Container will take the On-Success path only if all of the contained activities complete successfully. If any fail, then the On-Failure path will be taken.

1. Add a parallel container operator to the canvas. We need something to represent the execution of a data flow which is the data flow operator.
2. To invoke our two data flows concurrently, drop two data flow operators into the parallel container operator and define their properties, as shown in Figure 5-43.



Populate_Time_Dimension

Label: Populate_Time_Dimension

Description:

Data flow: populate_time_dimension.etlmetadataxmi

Populate_Branch_Location_Dimension

Label: Populate_Branch_Location_Dimension

Description:

Data flow: populate_branch_location.etlmetadataxmi

Figure 5-43 Adding data flow operators to the parallel container

Setting the variables can be done anytime before executing the fact data flow. Therefore, the variable assignment operator can, in fact, be executed concurrently with the two data flows. When we drag the Set_data_range_variables task into the parallel container, it results in the flow shown in Figure 5-44, in which there are three activities executing concurrently.

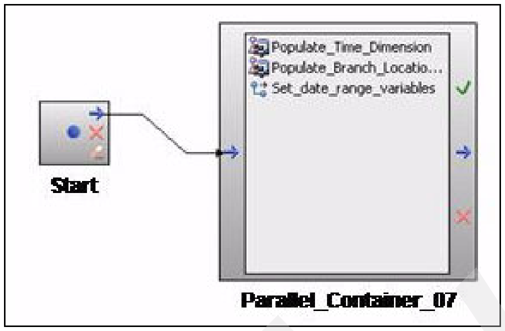


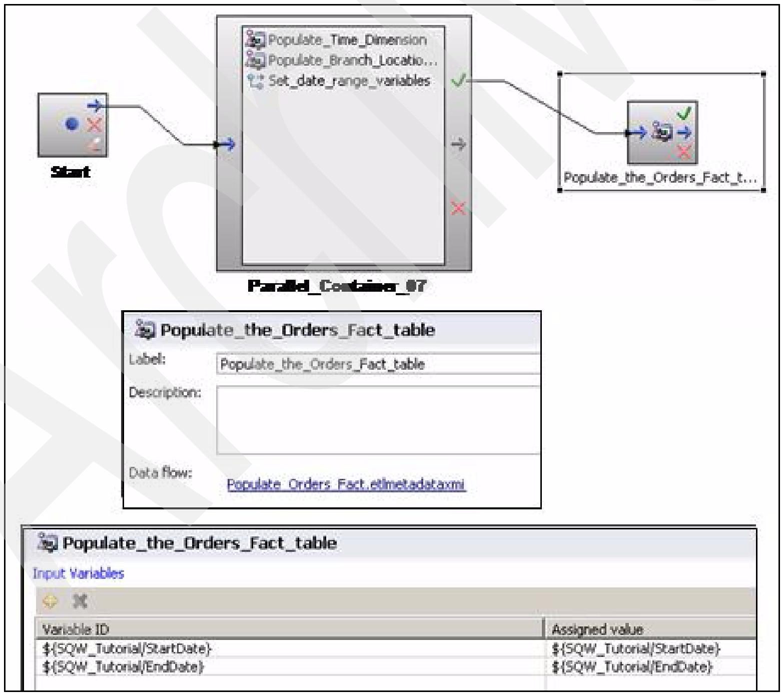
Figure 5-44 The completed parallel container operator

Populating the fact table

The next task is to populate the fact table by executing the Populate_Orders_Fact data flow but only after the successful completion of all of the dependent activities that are executed in parallel.

1. Add a data flow operator to the canvas and connect the On-Success port of the parallel container to the input port of the data flow operator. The On-Success port is only taken if all of the activities in the parallel operator complete successfully. If any of the operators fail, the On-Failure port is taken.
2. Set the properties of the data flow operator so it executes the Populate_Orders_Fact data flow.

   This data flow however needs the values of the two variables that was just read from the input file. These variable are globally available to called data flows and subprocesses. However, what if we wanted to execute two instances of the data flow with different date ranges? We could end up with unpredictable results. But, if we use the input variables property page to map the variables to the data flow instance, local copies of the variable are made for each data flow. The names can be the same. Figure 5-45 shows the completed Populate_the_Orders_Fact_table data flow operator.



Populate_the_Orders_Fact_table

Label: Populate_the_Orders_Fact_table

Description:

Data flow: Populate_Orders_Fact.etlmetadataxmi

Populate_the_Orders_Fact_table

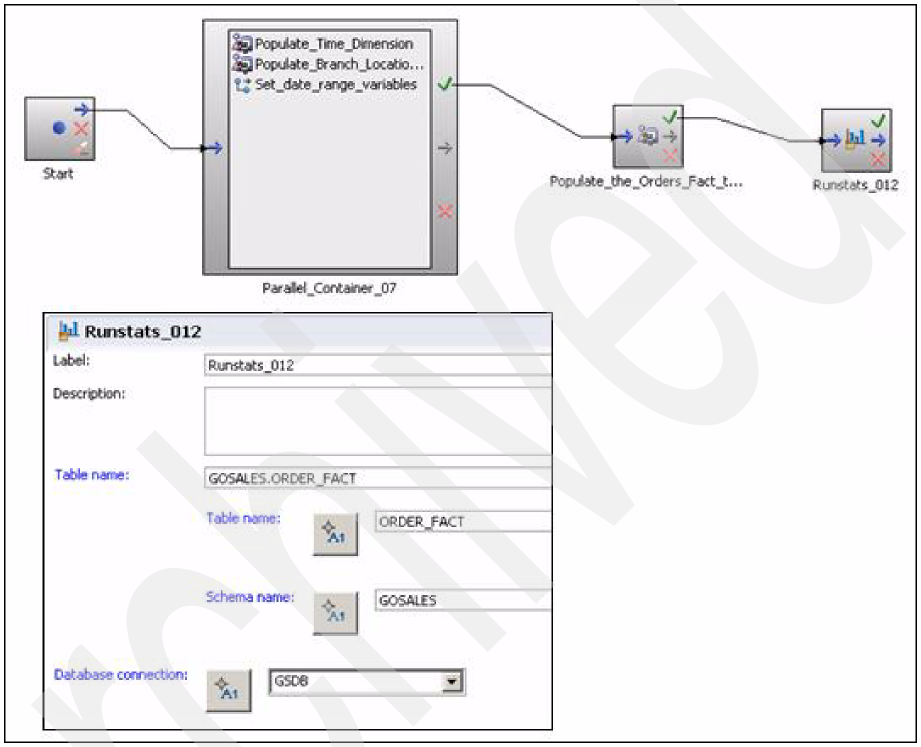
Input Variables

Variable ID	Assigned value
${SQW_Tutorial/StartDate}	${SQW_Tutorial/StartDate}
${SQW_Tutorial/EndDate}	${SQW_Tutorial/EndDate}

Figure 5-45 Data flow operator with mapped variables

Updating DB2 statistics

After a massive update to a table, update the database statistics so that the DB2 optimizer has the latest information for creating access paths. DB2 has a utility to update the database catalog statistics. Design Studio has an update statistics operator specific to the target platform. Figure 5-46 shows the Runstats operator for the LUW Design Studio.



Runstats_012

Label: Runstats_012

Description:

Table name: GOSALES.ORDER_FACT

Table name: ORDER_FACT

Schema name: GOSALES

Database connection: GSDB

Figure 5-46 LUW runstats operator

Terminating the branch

At this point all of the tasks for the main nominal processing branch have been added to the data flow. After the runstats operator, there are no connections. At execution time this results in an implicit completion of the data flow and the data flow completes without a problem. However, it is good practice to end the flows with either the end or fail terminal operator.

We will add the end operator to signify the end of this branch of the data flow, as shown in Figure 5-47.

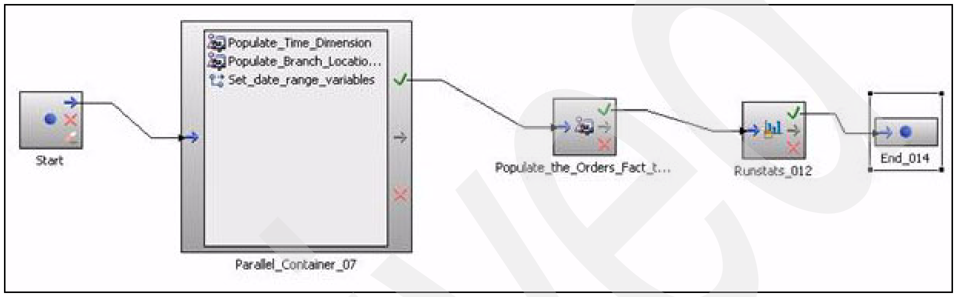


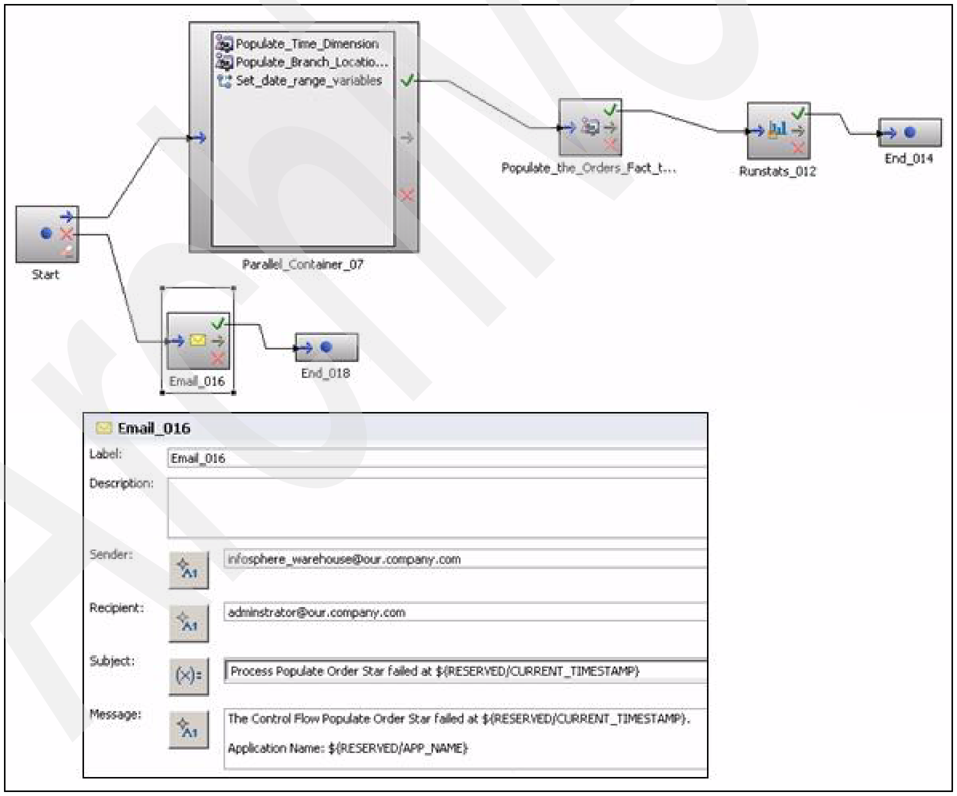
Figure 5-47 Terminating the branch with the end operator

Handling a failure

The final requirement is to send an email on any failure in the flow. This could be handled by connecting an email operator to the On-Failure port of every operator. Another option would be to use the process-level On-Failure port of the start operator. If anything fails in the data flow, this port is taken after a branch has terminated and before the Cleanup branch is taken. This is where we connect the email operator.

Drop an email operator on the canvas and connect the On-Failure port of the Start operator to the input port of the email operator. Add the end operator, as well, to terminate this flow branch. Set sender email, recipient email, subject and the message text, as shown in Figure 5-48.

Important: When executing a control from Design Studio, no emails are actually sent from email operators. The email text is sent to the execution log.



Email_016

Label:	Email_016
Description:	
Sender:	infosphere_warehouse@our.company.com
Recipient:	adminstrator@our.company.com
Subject:	Process Populate Order Star failed at ${RESERVED/CURRENT_TIMESTAMP}
Message:	The Control Flow Populate Order Star failed at ${RESERVED/CURRENT_TIMESTAMP}. Application Name: ${RESERVED/APP_NAME}

Figure 5-48 Adding the email operator

Note the use of reserved variables in the subject and message properties. You might have noticed that the message property is designated as a fixed value field according to the icon to its left but that is has variables inside. This is a way to add more structured formatted text to the message property. This can be done by keying in the variable names in the variable format.

An easier technique is to switch the property to a variable temporarily. Use the variable picker to select the variables that you want to use and append them to the field with spaces between each variable name. Highlight the variable field and copy to the clipboard. Switch the property back to a fixed value field and paste the variables into the field. They are now available in the multi-line property field for formatting into a nicer multi-line message, as shown in Example 5-4.

Example 5-4 Multi-line formatted message property with variables

```
The Control Flow Populate Order Star failed at
${RESERVED/CURRENT_TIMESTAMP}.

Application Name: ${RESERVED/APP_NAME}
Process name:  ${RESERVED/PROC_NAME}
Instance: ${RESERVED/INSTANCE_NAME}
Process start time:  ${RESERVED/START_TIMESTAMP}
```

5.3.3 Testing and debugging

The next task is to see if the control flow does what it is designed to do. The control flow can be tested without leaving Design Studio. That is, it can be manually executed from start to finish or breakpoints can be added to the flow and the flow executed in debug mode. How you go about testing and debugging is primarily based on personal preference.

You can go about developing your data flow in an incremental manner adding a few operators and testing up to that point before adding to the control flow. A few developers execute the flow first and go into debug mode if there are problems. Other developers establish breakpoints in the flow and execute the flow in debug mode first, fixing problems as they occur with a full final execution run.

Both approaches validate the flow and generate the code before attempting execution. Validation is performed on every operator in the control flow. If there are any errors. use the same techniques as in data flows to find the problems.

Debugging the flow

A control flow behaves more like a traditional program, with independent units of execution in a sequential fashion. Debugging is also more traditional as compared to the data flow debugger. You can set breakpoints on individual operators which, during the debug session, pause just before executing that operator. You have the option to execute to the next operator to the next breakpoint, to execute to the end of the control flow, or to cancel.

Tip: As a best practice, set a breakpoint on the start operator. If there are no breakpoints a debug session executes through the flow without pausing. It is easy to jump into a debug session forgetting to first set a breakpoint. Always setting a breakpoint on the start operator pauses at the beginning of the flow during a debug session.

Perform the following steps:

1. Set a breakpoint on the start operator by highlighting the start operator and selecting **Toggle Breakpoint** from the control flow menu. The start operator now has a blue border indicating that it has a breakpoint. This is shown in Figure 5-49.

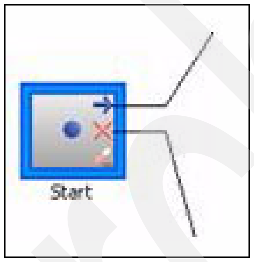
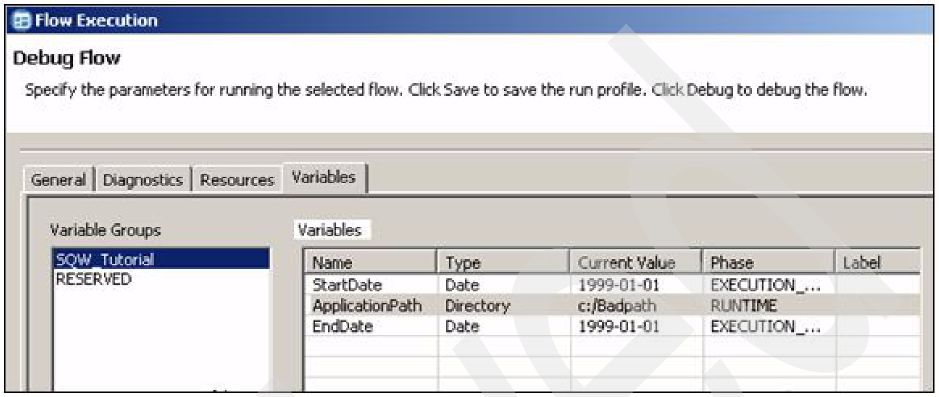


Figure 5-49 Breakpoint set on an operator

2. Start the debug session by selecting **Debug Control Flow** from the control flow menu. The Flow Execution dialog box opens.

3. Verify that the Trace level is set to **Both** on the Diagnostics tab. It should already be set because we defined the trace level in the control flow properties. Introduce an error by setting an invalid value for the ApplicationPath variable, as shown in Figure 5-50.


Flow Execution
Debug Flow
Specify the parameters for running the selected flow. Click Save to save the run profile. Click Debug to debug the flow.
General
Diagnostics
Resources
Variables
Variable Groups
SQW_Tutorial
RESERVED
Variables

Name	Type	Current Value	Phase	Label
StartDate	Date	1999-01-01	EXECUTION_...	
ApplicationPath	Directory	c:/Badpath	RUNTIME	
EndDate	Date	1999-01-01	EXECUTION_...	

Figure 5-50 Setting the variable with an invalid value

4. Click **Debug** to continue into the debug session.

   The flow starts executing in debug mode and pauses at the first breakpoint encountered, which, for our flow, is the start operator. The current operator is highlighted in orange (Figure 5-51) and the flow is paused.

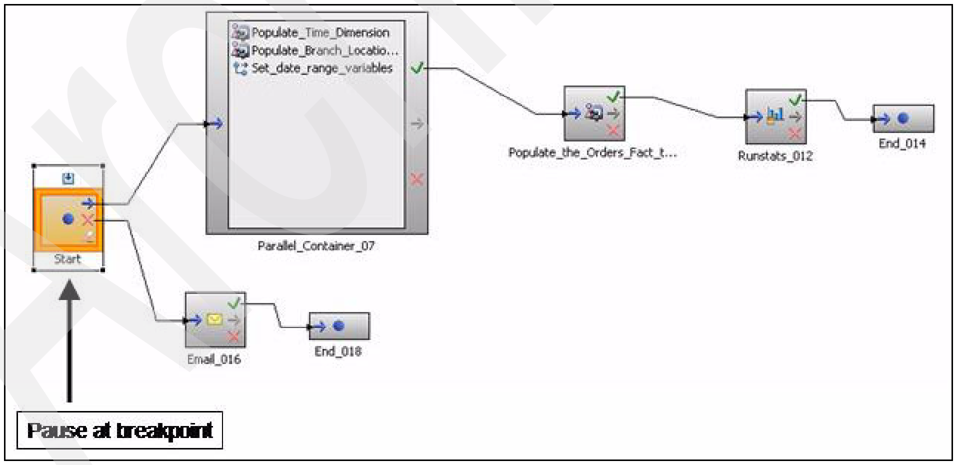


Figure 5-51 Flow paused at the first breakpoint

5. In our debug session we execute each operator one at a time. Click **Next** from the control flow menu to execute the start operator and continue to the parallel container operator, which is now highlighted in orange.

6. Click **Next** to execute the parallel container operator. The data flows and the variable assignment operator execute concurrently and continue to the next operator.

   Look at Figure 5-52 to see what happened. The parallel container turned red. The next operator to execute is the email operator. One of the three activities in the parallel container failed. However, at this point we cannot tell which one failed. Because the parallel container failed, the branch connected to the On-Failure port of the parallel container is taken. There is nothing there, so it is an implicit end to that branch. Because there was a failure in an operator in the control flow, the process-level On-Failure port of the start operator is taken, which leads to the email operator that is highlighted as the next to execute.

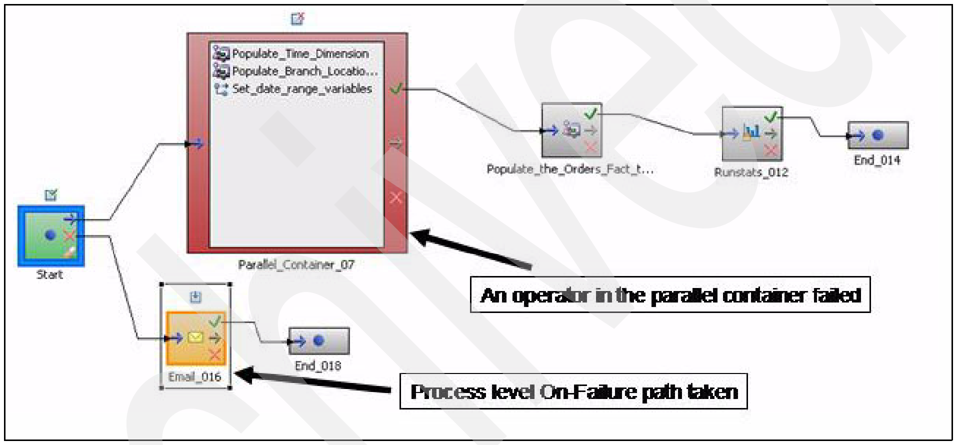

Populate_Time_Dimension
Populate_Branch_Locatio...
Set_date_range_variables
Populate_the_Orders_Fact_t...
Runstats_012
End_014
Start
Parallel_Container_07
An operator in the parallel container failed
Email_016
End_018
Process level On-Failure path taken

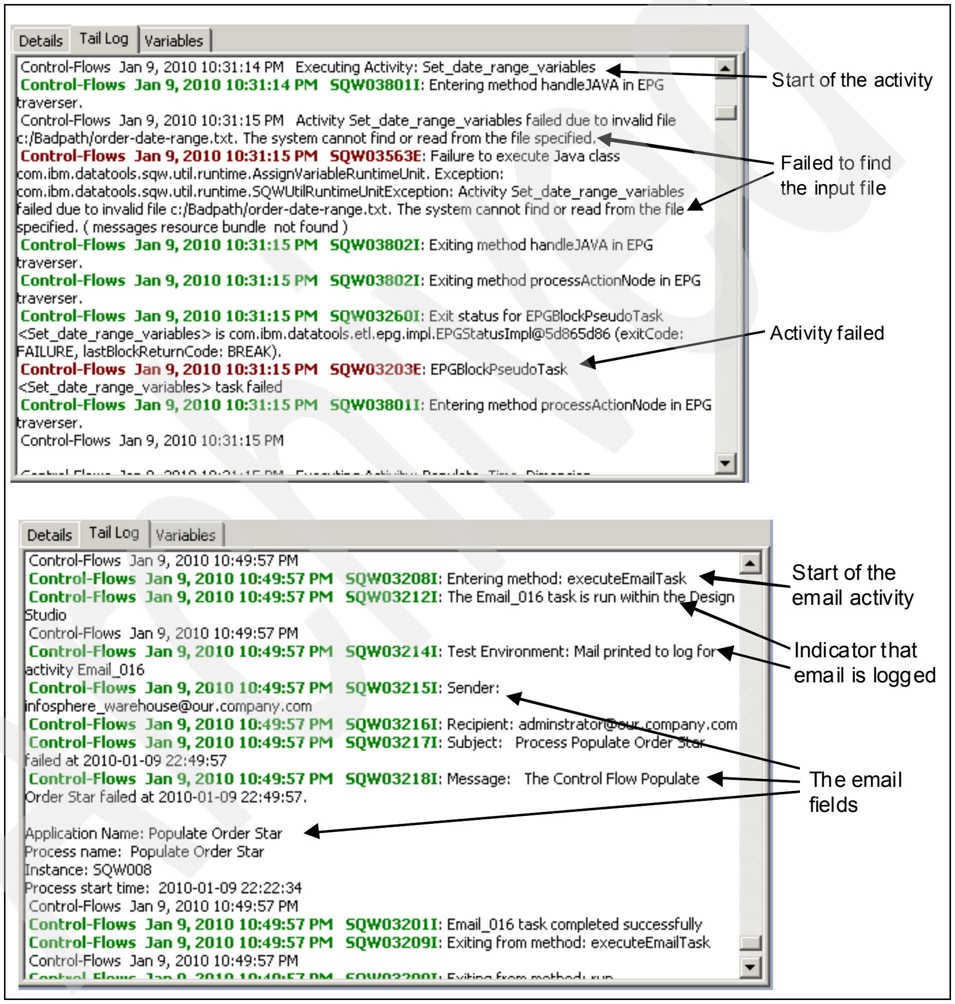
Figure 5-52 Control flow failed

   We understand the error because we intentionally created it by providing an invalid value for the ApplicationPath variable. That variable was used by the variable assignment operator, called Set_date_range_variables. As a result, the input file containing the data range values could not be found. That can be verified by looking at the log.

7. Execute the rest of the flow by selecting **Continue to end** from the control flow menu. The flow finishes executing and presents us with the completion dialog box, indicating that the failure failed. Close that dialog box and examine the log from the Execution Status view tab.

As when testing data flows, we can examine the log from the Execution Status view where the log is color-coded. This makes it easier to locate errors. Examining the log excerpts in Figure 5-53 on page 146, we find the error highlighted, in red, which shows the failure of the Set_date_range_variables. This is the variable assignment operator. It failed because it could not find or read the input file. That is not a surprise as we provided an invalid value for the ApplicationPath variable. Therefore, the operator could not locate the input file.

Also in Figure 5-53 is the excerpt from the log file showing the execution of the email operator. There is an indication that because we are executing from Design Studio, the email is logged instead of sent. The content of the email is logged, and we see that the variables have been replaced with the specific values at the time of failure.


Details
Tail Log
Variables
Start of the activity
Failed to find the input file
Activity failed
Start of the email activity
Indicator that email is logged
The email fields

```
Control-Flows  Jan 9, 2010 10:31:14 PM  Executing Activity: Set_date_range_variables
Control-Flows  Jan 9, 2010 10:31:14 PM  SQW03801I: Entering method handleJAVA in EPG
traverser.
Control-Flows  Jan 9, 2010 10:31:15 PM  Activity Set_date_range_variables failed due to invalid file
c:/Badpath/order-date-range.txt. The system cannot find or read from the file specified.
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03563E: Failure to execute Java class
com.ibm.datatools.sqw.util.runtime.AssignVariableRuntimeUnit. Exception:
com.ibm.datatools.sqw.util.runtime.SQWUtilRuntimeUnitException: Activity Set_date_range_variables
failed due to invalid file c:/Badpath/order-date-range.txt. The system cannot find or read from the file
specified. ( messages resource bundle  not found )
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03802I: Exiting method handleJAVA in EPG
traverser.
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03802I: Exiting method processActionNode in EPG
traverser.
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03260I: Exit status for EPGBlockPseudoTask
<Set_date_range_variables> is com.ibm.datatools.etl.epg.impl.EPGStatusImpl@5d865d86 (exitCode:
FAILURE, lastBlockReturnCode: BREAK).
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03203E: EPGBlockPseudoTask
<Set_date_range_variables> task failed
Control-Flows  Jan 9, 2010 10:31:15 PM  SQW03801I: Entering method processActionNode in EPG
traverser.
Control-Flows  Jan 9, 2010 10:31:15 PM
```

```
Control-Flows  Jan 9, 2010 10:49:57 PM
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03208I: Entering method: executeEmailTask
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03212I: The Email_016 task is run within the Design
Studio
Control-Flows  Jan 9, 2010 10:49:57 PM
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03214I: Test Environment: Mail printed to log for
activity Email_016
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03215I: Sender:
infosphere_warehouse@our.company.com
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03216I: Recipient: adminstrator@our.company.com
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03217I: Subject:   Process Populate Order Star
failed at 2010-01-09 22:49:57
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03218I: Message:   The Control Flow Populate
Order Star failed at 2010-01-09 22:49:57.

Application Name: Populate Order Star
Process name:  Populate Order Star
Instance: SQW008
Process start time:  2010-01-09 22:22:34
Control-Flows  Jan 9, 2010 10:49:57 PM
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03201I: Email_016 task completed successfully
Control-Flows  Jan 9, 2010 10:49:57 PM  SQW03209I: Exiting from method: executeEmailTask
Control-Flows  Jan 9, 2010 10:49:57 PM
```

Figure 5-53   Examining the log of the failed control flow

Tip: When debugging a control flow with a parallel operator, it can be difficult to identify all of the entries associated with the activities that execute concurrently, as the entries might be interleaved in the log. There is a property in the parallel container that forces the activities to execute sequentially, resulting in all the log entries for the activities to be listed sequentially in the log.

Another reason to set the execution to sequential is if you have fewer resources in your development environment compared to the target runtime system. Running many activities concurrently can overwhelm your development machine. Remember to change it back before deploying the flow.

Executing the flow

We could continue to execute in debug after correcting the above error. However, we choose to execute the flow providing the correct value for the ApplicationPath variable. Perform the following steps:

1. Select **Execute** from the control flow menu. This opens the flow execution dialog box.
2. Set the value of the ApplicationPath variable to `c:/OrdersApp`. This is the default value
3. Execute the flow. It successfully completes and populates the target ORDER_FACT table.

5.3.4 Code generation

When code is generated for a control flow, code for all of the underlying data flows and subprocesses is also generated. The code generated for a control flow is an EPG similar to the EPG we saw with a data flow in Example 5-1 on page 112. However, there are code units that invoke tasks other than the SQL generated from a data flow.

In our control flow, we used the variable assignment operator to read values from a file and assign them to variables, which would not generate SQL. Example 5-5 on page 148 contains an excerpt from the Control Flow EPG that invokes a Java program to perform the function of the variable assignment operator.

Example 5-5   Excerpt of Control Flow EPG for variable assignment operator

```
RUNTIME : Set_date_range_variables ( ) : type GRAPH : node
{
   PREP ( ) : type BLOCK : node /graph9
   {
   }
   ( ):CODE_UNIT, node /node12;

   CODE_UNIT:JAVA
   (OperatorTag = /op:07/op:07_2

   OperatorLabel = Set_date_range_variables

   Activity: = Set_date_range_variables

   assignValueOption = fileOption

   fileName = ${SQW_Tutorial/ApplicationPath}/order-date-range.txt

   delimiter = ,

   lineNum = 2

   numVarAssign = 2

   varID0 = ${SQW_Tutorial/StartDate}

   varVal0 =

   varID1 = ${SQW_Tutorial/EndDate}

   varVal1 =

   )
   {
   Java Runtime Unit Class:
com.ibm.datatools.sqw.util.runtime.AssignVariableRuntimeUnit
   }
}
CATCH_BLOCK ( ) : type BLOCK : node /graph7
{
}
FINALLY_BLOCK ( ) : type BLOCK : node /graph8
{
}
```

A control flow EPG contains all of the code for all of the data flows and other activities in one large EPG script. This, in effect, is the execution plan for the control flow used by the runtime engine.

5.3.5 Control flow operators

To this point in this book we have discussed control flows in a generic sense without regard to whether the InfoSphere Warehouse platform is distributed (LUW) or System z. Development methodology is the same across both products and many operators are also similar across the two products. However, a control flow for the LUW product is different from a control flow for the System z product due to the nature of each platform. See the DB2 Information Center for your platform for information about the operators.

Figure 5-54 shows the control flow operators for InfoSphere Warehouse V9.7 for the Linux, UNIX, and Windows (LUW) platforms and InfoSphere Warehouse on System z V9.5.2.

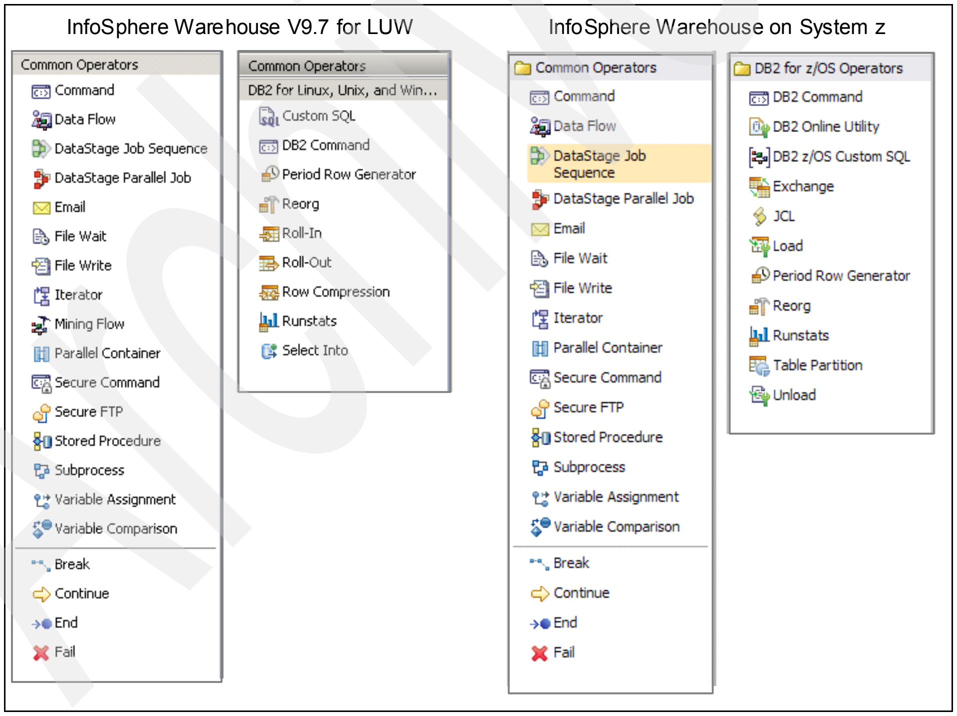


Figure 5-54 InfoSphere Warehouse control flow operators

Common control flow operators

There is a lot of commonality between the SQW operators of the two product platforms. In fact, there are many operators in common. However, while performing the same basic function, those common operators likely have differences in properties specific to each platform. The following sections detail task and flow activity descriptions, and flow control operators for the LUW and System z platforms:

Task activities

Task activities are shown in Table 5-5.

Table 5-5 Task activity descriptions

Activity	Description
Command	The Command operator invokes tasks through the command line interface. Non-interactive executables and ftp commands are supported. The command will execute local to the control point from which it is submitted, either Design Studio or the administration console.
data flow	The data flow operator invokes a data flow.
DataStage Job Sequence	The DataStage Job Sequence operator invokes a DataStage job of the type sequence. which will execute at the defined DataStage server. A system resource must be defined for the DataStage server.
DataStage Parallel Job	The DataStage Parallel Job operator will invoke a DataStage parallel job, which will execute at the defined DataStage server. A system resource must be defined for the DataStage server.
Email	The Email operator will compose and send an email. When executing in Design Studio the email is written to the log rather than being sent. In a deployed application, the email will be sent using the SMTP server defined at the administration console.
File Wait	A File Wait operator stops control flow processing for a period of time while the system checks for the existence or non-existence of a specified file at the control point, either Design Studio or administration console. You can specify the amount of time to allocate to this operator. In addition to the obvious use of waiting for a data file to arrive, the file wait operator can be used to coordinate processing by checking for the existence or non-existence of files used as flags.
File Write	The File Write operator is used to write information to the log or to a user file at the control point, Design Studio or administration console. This is useful for debugging a flow or for creating an application-specific log file.

Activity	Description
Parallel Container	The Parallel Container operator is used to hold other control flow operators that can be started concurrently from an SQW perspective. Whether they actually execute in parallel depends upon the configuration of the runtime resources. The flow designer needs to understand that all activities will be started and could overwhelm the various servers if the appropriate resources are not available. There is a property that forces the activities to be executed sequentially which is useful for testing in a development or small test environment where the systems are configured with less resources that then target environment.
Secure Command	The Secure Command operator uses a secure shell (SSH) connection to run a script or a command on a remote computer. You can execute DB2 scripts, shell scripts and executables. This operator can be useful in a development environment where Design Studio is running under a Windows environment but scripts or executables are on a Linux or UNIX environment with no Windows equivalent.
Secure FTP	Use the Secure FTP operator to transfer files between a local and a remote computer using the SSH File Transfer Protocol (SFTP) to transfer the files. The Secure FTP operator supports a larger number SFTP commands and can use wildcards to transfer multiple files.
Stored Procedure	Use the Stored Procedure operator to execute an existing DB2 stored procedure that is executed at the defined database connection. Stored Procedures are an easy way to add customized code to your SQW application. However, the return of row sets is not supported. You can type or select values in a variable assignment table, or read the variable values from an external delimited file.
Variable Assignment	The Variable Assignment operator assigns values to variables in a control flow. A variable can accept either a fixed value or a variable.
Custom SQL/DB2 z / OS Custom SQL	The Customer SQL operator will execute one or more user provided SQL statements in a control flow.
DB2 Command	The DB2 Command operator will invoked DB2 SQL scripts or DB2 Shell commands. The script or command will execute at the control point, Design Studio or administration console.
Reorg	The Reorg operator is used to invoked the DB2 Reorg utility to reorganize tables or indexes.
Runstats	Use the Runstats operator to update DB2 database statistics using the Runstats utility.
Period Row Generator	Use a Period Row Generator operator in a control flow to generate rows and insert them into a period or time dimension table. Operator properties define the granularity of the table and functions that generate values for specific columns in the target table.

Flow activities

Flow activities are shown in Table 5-6.

Table 5-6 Flow activity descriptions

Activity	Description
Variable Comparison	The Variable Comparison operator compares the input variable with a value that is either fixed or variable. The operator evaluates a comparison condition to determine whether the output is true or false. The operator has true and false output ports which determines the flow branch to take based on the result of the comparison condition.
Iterator	The Iterator operator defines a processing loop. The loop might iterate over each delimited data item in a file, each file in a directory or a fixed number of times. The iteration value is available in a variable. An End Iterator is automatically placed in the canvas. Operators to process in the loop will be connected between the Interator and the End Interator operators.
Break	The Break operator is used inside an iterator operator to explicitly end the loop.
Continue	The Continue operator is used inside an iterator operator to start the next iteration in the loop even when an activity in the iteration loop fails.
End	An End operator represents the end of a sequence of operators or of a control flow. The use of end operators is optional but, as a best practice, should be used.
Fail	You can use the Fail operator on the output port of an activity to explicitly cause the control flow to fail when a certain condition is met.

DB2 for LUW control flow operators

The operators in Table 5-7 are specific to InfoSphere Warehouse for LUW V9.7.

Table 5-7 DB2 for LUW control flow operator descriptions

Operator	Description
Roll-In	The roll-in operator attaches a new data partition to a partitioned DB2 table.
Roll-Out	The roll-out operator detaches a partition from an existing partitioned DB2 table to a separate table.
Row Compression	The row compression operator enables or disables compression on the data that is stored in a DB2 table and reduces the disk storage space that the table requires.
Select Into	The select into operator uses an SQL SELECT statement to retrieve column values from a single row in either a local or a remote database table. You can assign the column values to variables that you use later in the control flow or subprocess.

DB2 for z/OS operators

The operators in Table 5-8 are specific to InfoSphere Warehouse for z/OS V9.2.1.

Table 5-8 DB2 for z/OS operator descriptions

Operator	Description
DB2 Online Utility	The DB2 online utility operator runs any DB2 for z/OS online utility that can be called by the stored procedure DSNUTILU. You can use the utility operator to provide control statements to execute DB2 utilities inside a control flow.
Exchange	Use an exchange operator in a control flow to exchange the content of a base table and its associated clone table.
JCL	Use the job control language (JCL) operator to run JCL that processes a batch job on a z/OS system. The JCL script can either be on a local computer or on a remote z/OS system. You can also specify custom JCL directly in the Properties View of the JCL operator.
Load	You can define a load operator in a control flow to use the DB2 LOAD utility for loading a relational database table in the DB2 for z/OS database. The operator calls the DB2 LOAD utility to load the records from input files, tables, or data sets into the target table. Data from remote files might be transferred over FTP or Batch Pipes.
Unload	You can define an unload operator in a control flow to use the DB2 UNLOAD utility for unloading data from one or more source objects (table, table space, image copy data set) to one or more Basic Sequential Access Method (BSAM) sequential data sets in external formats.
Table Partition	The table partition operator can rotate or create additional partitions in partitioned tables and change partition boundaries depending on the amount of data that is stored in each partition.

5.4 Moving from development to runtime

Once the data flows and control flows have been developed and tested in the development environment of Design Studio, they have to be installed into the runtime environment of the administration console for production use. You can also have a separate environment for system test or quality assurance testing.

In this deployment process there is deployment preparation and deployment into the runtime environment. In Design Studio this consists of defining a warehouse application that is a collection of control flows. Code for the included flows is generated and packaged into a deployment zip file. This file is used by the administrator of the runtime environment. The administrator uses the web-based administration console to define any database and system resources required, to install the Warehouse Application, and to configure any needed schedules.

To illustrate the process, we shall describe the deployment of the flows developed in this chapter to the runtime environment.

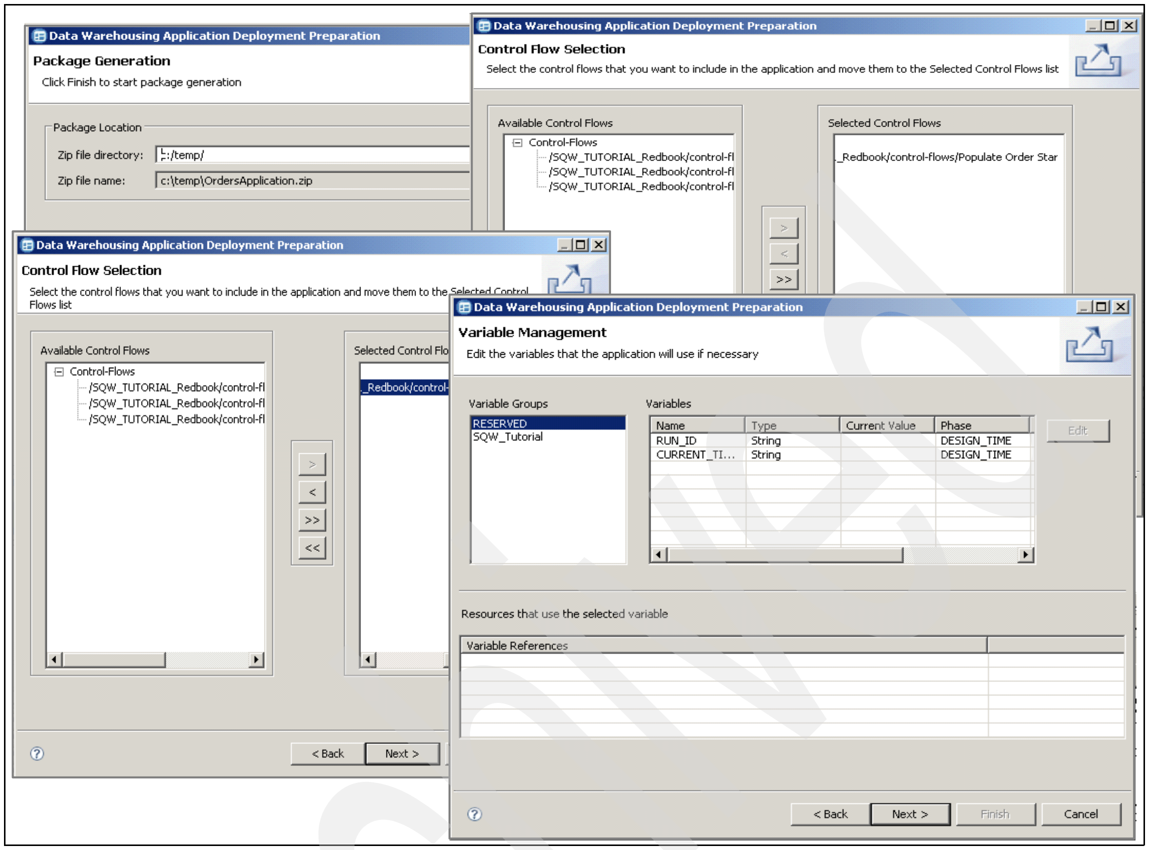
Creating the deployment package

The first step in the deployment process is to define the warehouse application and generate the deployment package.

In the Data Project Explorer there is a folder called Warehouse Application Profiles. This is where objects that define warehouse applications are created.

1. Select New → Data Warehouse Application from the folder context menu to start the Data Warehousing Application Deployment Preparation Wizard.
2. Select the project and name it `OrdersApplication`.
3. We only have one control flow, so select the **Populate Orders Star** control flow. At this point, we can edit the database and system resource definitions, set the values of variables, and include any miscellaneous files. However, our application does not require any modifications to these definitions.

   Continuing through the wizard, the application profile is saved and the code is generated.
4. Specify a directory where to save the zip file. When the wizard completes, the zip file deployment package is saved. Figure 5-55 on page 155 shows sample select pages from the wizard. It is not necessary to read the select pages, they are displayed for your familiarity.


Data Warehousing Application Deployment Preparation

Package Generation

Click Finish to start package generation

Package Location

Zip file directory:	c:/temp/
Zip file name:	c:\temp\OrdersApplication.zip

Data Warehousing Application Deployment Preparation

Control Flow Selection

Select the control flows that you want to include in the application and move them to the Selected Control Flows list

Available Control Flows

- Control-Flows
  - /SQW_TUTORIAL_Redbook/control-fl
  - /SQW_TUTORIAL_Redbook/control-fl
  - /SQW_TUTORIAL_Redbook/control-fl

Selected Control Flows

_Redbook/control-flows/Populate Order Star

Data Warehousing Application Deployment Preparation

Control Flow Selection

Select the control flows that you want to include in the application and move them to the Selected Control Flows list

Available Control Flows

- Control-Flows
  - /SQW_TUTORIAL_Redbook/control-fl
  - /SQW_TUTORIAL_Redbook/control-fl
  - /SQW_TUTORIAL_Redbook/control-fl

Selected Control Flo

_Redbook/control-

Data Warehousing Application Deployment Preparation

Variable Management

Edit the variables that the application will use if necessary

Variable Groups

- RESERVED
- SQW_Tutorial

Variables

Name	Type	Current Value	Phase
RUN_ID	String		DESIGN_TIME
CURRENT_TI...	String		DESIGN_TIME

Resources that use the selected variable

Variable References	

Figure 5-55 Creating the deployment package

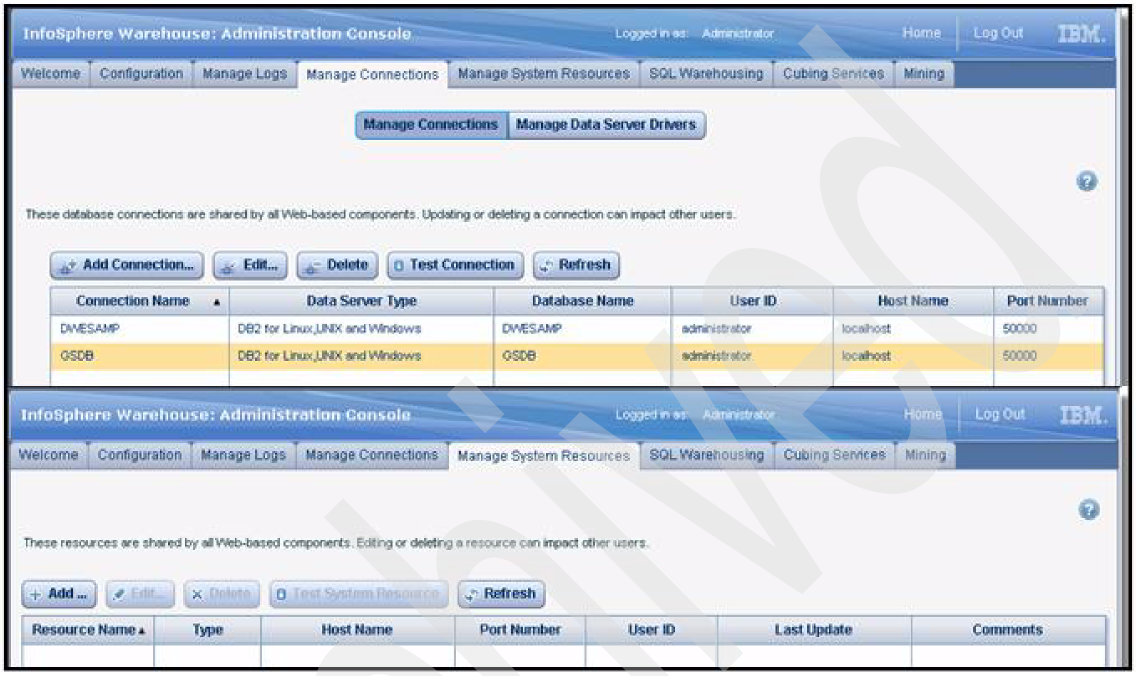
Installing the deployment package into runtime

The deployment package is given to the administrator of the runtime environment for installation using the administration console. The administrator uses the administration console to define any database and system resources that are needed by the application. The administration console is also used to install the warehouse application into the runtime environment, and to schedule, execute and monitor the application activity.

Defining database and system resources

Any database used in the warehouse application (whether a source, execution, or target database) must have a connection defined so that the administration console can make the database connections. Connections also have to be made for any system resources, such as FTP servers or remote command systems.

Figure 5-56 shows the administration console pages for defining database connections and system resources. We have defined the connection to our target database, GSDB. The connection definition hold the specifics of how to connect and log in to the database.



These database connections are shared by all Web-based components. Updating or deleting a connection can impact other users.

Connection Name ▲	Data Server Type	Database Name	User ID	Host Name	Port Number
DWESAMP	DB2 for Linux,UNIX and Windows	DWESAMP	administrator	localhost	50000
GSDB	DB2 for Linux,UNIX and Windows	GSDB	administrator	localhost	50000

These resources are shared by all Web-based components. Editing or deleting a resource can impact other users.

Resource Name ▲	Type	Host Name	Port Number	User ID	Last Update	Comments

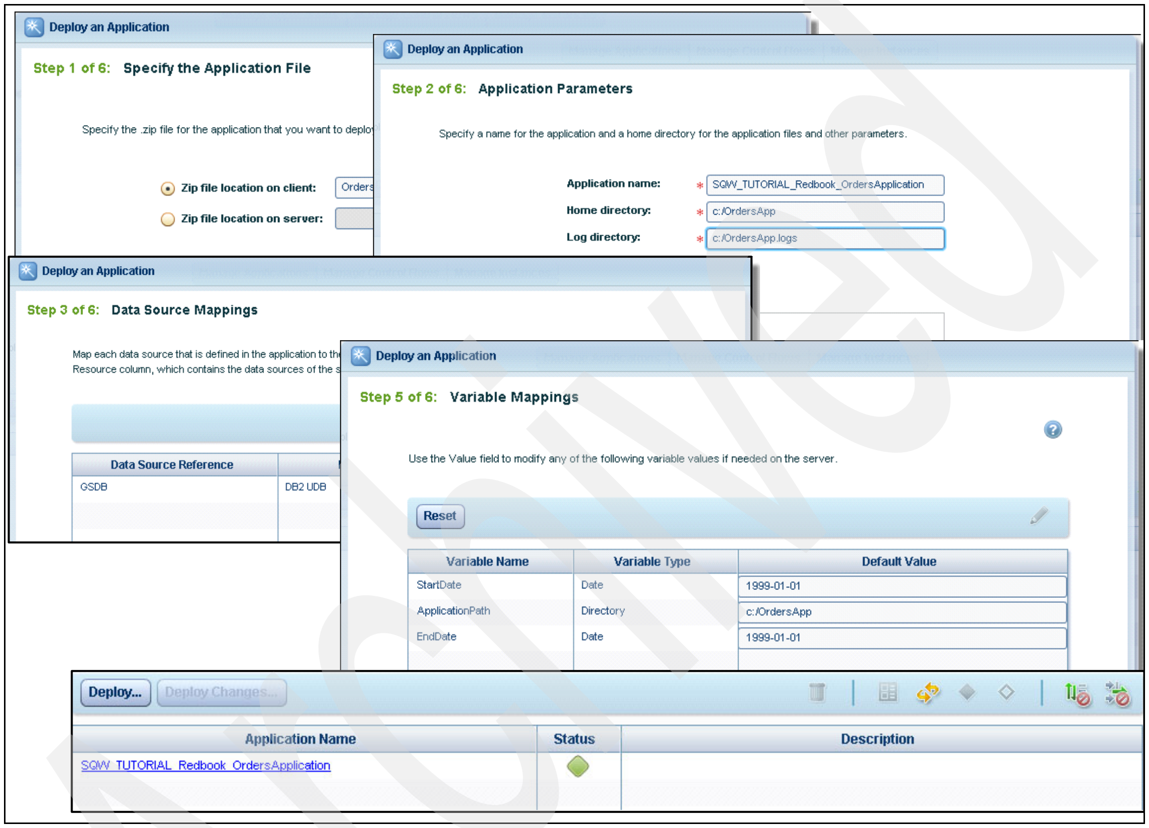
Figure 5-56 Defining database and system resources

Installing the deployment package

The deployment zip file created by the pre-deployment process in Design Studio is used as input to install the application. The SQL Warehousing tab on the administration console contains the functionality to manage warehouse applications, manage the control flows of the warehouse applications, and manage the specific execution instances of control flows.

Install the application from the Manage Applications page of the SQL Warehousing tab through the following steps:

1. Click the **Deploy** button on the Manage Applications page of the SQL Warehousing tab. This starts the Deploy an Application wizard. Figure 5-57 shows the windows generated by the deploy process.



Deploy an Application

Step 1 of 6: Specify the Application File

Specify the .zip file for the application that you want to deplo

[✓ Zip file location on client:] Orders

Zip file location on server:

Deploy an Application

Step 2 of 6: Application Parameters

Specify a name for the application and a home directory for the application files and other parameters.

Application name: *	SQW_TUTORIAL_Redbook_OrdersApplication
Home directory: *	c:/OrdersApp
Log directory: *	c:/OrdersApp.logs

Deploy an Application

Step 3 of 6: Data Source Mappings

Map each data source that is defined in the application to th
Resource column, which contains the data sources of the s

Data Source Reference	
GSDB	DB2 UDB

Deploy an Application

Step 5 of 6: Variable Mappings

Use the Value field to modify any of the following variable values if needed on the server.

Reset

Variable Name	Variable Type	Default Value
StartDate	Date	1999-01-01
ApplicationPath	Directory	c:/OrdersApp
EndDate	Date	1999-01-01

Deploy... Deploy Changes...

Application Name	Status	Description
SQW_TUTORIAL_Redbook_OrdersApplication	◆	

Figure 5-57 Deploying the warehouse application

2. Specify the application deployment file location, which could be on the server with the Administration Server or on the machine that is running the browser.

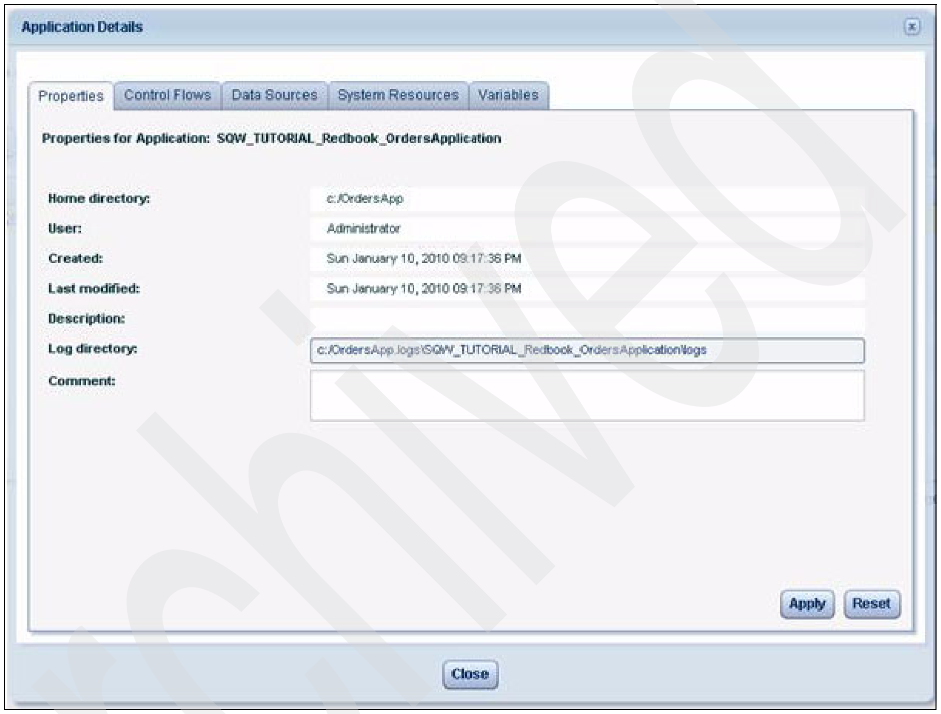
3. Define the application home and log directory.

   Each database in the application must be mapped to a database as defined in the administration console. The names do not have to be the same.

4. Set any variables that are still eligible to be modified at this time. Set the ApplicationPath variable to the directory that contains the file for the StartDate and EndDate values. However, it can be modified later. The StartDate and EndDate variables are set in the control flow, so there is no need to modify

them now. When the wizard completes without errors, the application is installed and enabled.

For more details about the application, click the application name to open the Application Details (Figure 5-58) window. Details include the control flows, the database and system resources, and the variables. The default values of variables might be modified depending on the defined change phase.



Properties for Application: SQW_TUTORIAL_Redbook_OrdersApplication

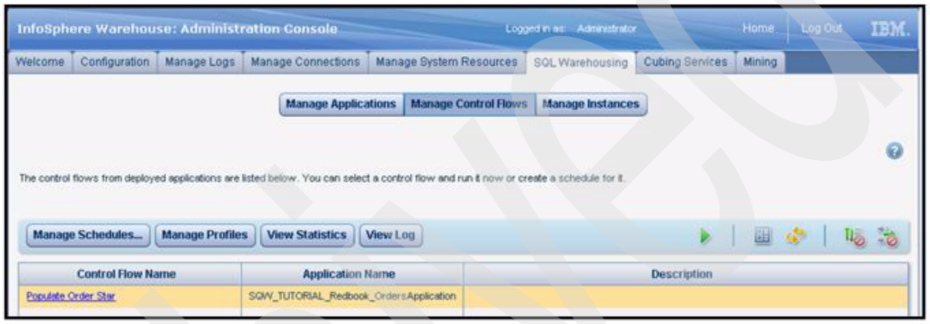
Home directory:	c:/OrdersApp
User:	Administrator
Created:	Sun January 10, 2010 09:17:36 PM
Last modified:	Sun January 10, 2010 09:17:36 PM
Description:	
Log directory:	c:/OrdersApp.logs\SQW_TUTORIAL_Redbook_OrdersApplication\logs
Comment:	

Figure 5-58 Application Details window in the administration console

5.5 Executing and monitoring control flows

In the runtime environment, the unit of execution is the control flow. Control flows can be managed from the control flows page of the SQL Warehousing tab, as shown in Figure 5-59. From this page, you can perform the following tasks:

- Manually execute a control flow
- Create an execution schedule
- Manage execution profiles
- View execution statistics
- View the application log



The control flows from deployed applications are listed below. You can select a control flow and run it now or create a schedule for it.

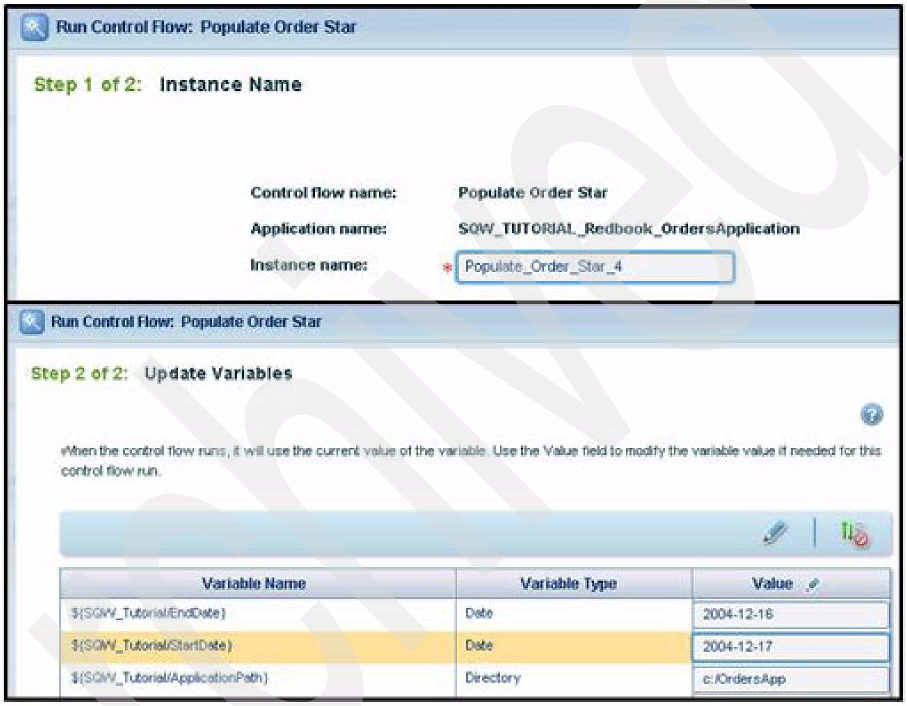
Control Flow Name	Application Name	Description
Populate Order Star	SQW_TUTORIAL_Redbook_OrdersApplication	

Figure 5-59 List of control flows for all deployed applications

Manually executing a control flow

To manually run a control flow, perform the following steps:

1. Highlight the control flow entry and click **Run**. This opens the Run Control Flow window shown in Figure 5-60.
2. Provide a unique Instance name to identify this running of the control flow.
3. Set variables as needed. When the dialog is complete, the control flow is submitted for execution.



Run Control Flow: Populate Order Star

Step 1 of 2: Instance Name

Control flow name:	Populate Order Star
Application name:	SQW_TUTORIAL_Redbook_OrdersApplication
Instance name:	* Populate_Order_Star_4

Run Control Flow: Populate Order Star

Step 2 of 2: Update Variables

When the control flow runs, it will use the current value of the variable. Use the Value field to modify the variable value if needed for this control flow run.

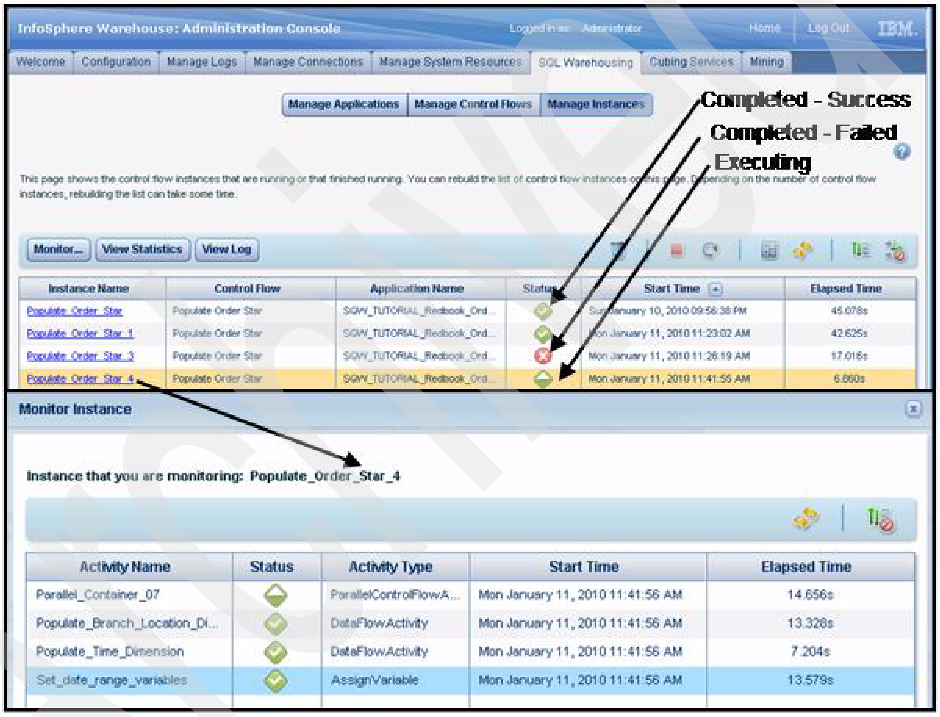
Variable Name	Variable Type	Value
${SQW_Tutorial/EndDate}	Date	2004-12-16
${SQW_Tutorial/StartDate}	Date	2004-12-17
${SQW_Tutorial/ApplicationPath}	Directory	c:/OrdersApp

Figure 5-60 Run Control Flow window

Monitoring the control flow execution

View the status of control flow execution instances from the Manage Instances page of the SQL Warehousing tab. This page lists all execution instances and basic information about the instance such as an icon depicting the execution status, start time and elapsed time. This provides a quick view of what is happening in your runtime environment.

Figure 5-61 shows two instances that have completed successfully, one that has failed, and one that is currently executing. Clicking the link for an instance displays properties of the instance, including variables that were set. Highlighting an instance and clicking **Monitor** shows the status of the individual activities.



This page shows the control flow instances that are running or that finished running. You can rebuild the list of control flow instances on this page. Depending on the number of control flow instances, rebuilding the list can take some time.

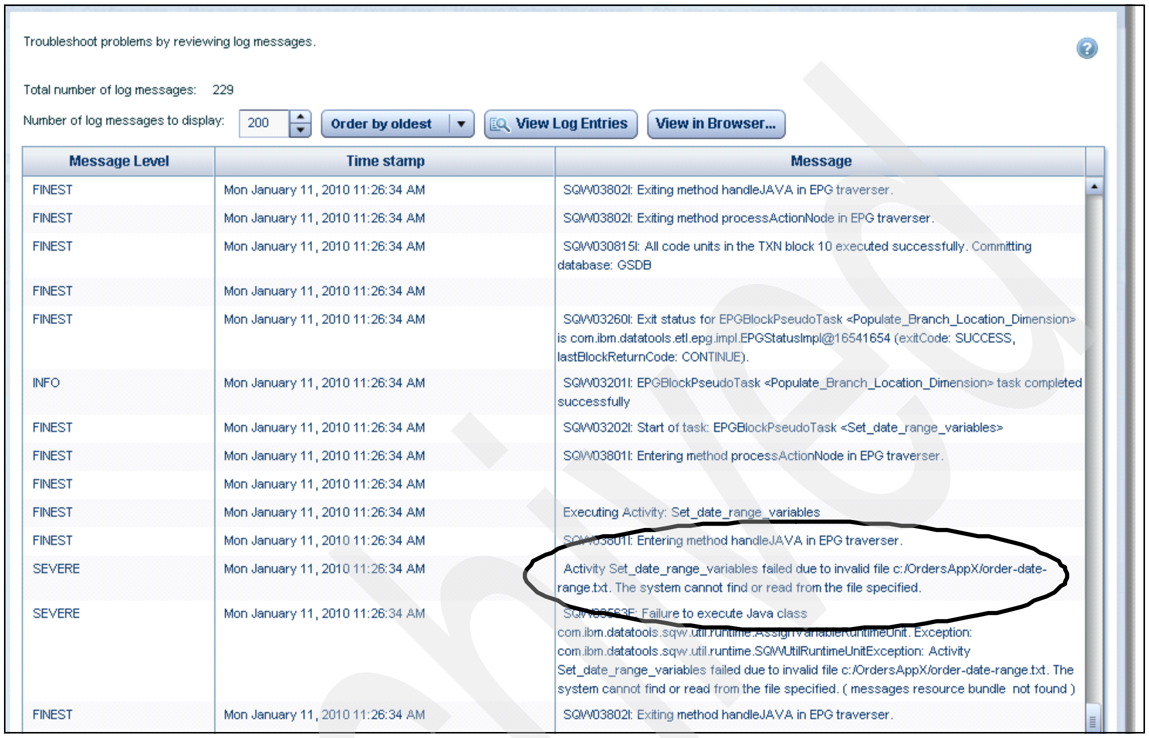
Instance Name	Control Flow	Application Name	Status	Start Time	Elapsed Time
Populate_Order_Star	Populate Order Star	SQW_TUTORIAL_Redbook_Ord...	✓	Sun January 10, 2010 09:56:38 PM	45.078s
Populate_Order_Star_1	Populate Order Star	SQW_TUTORIAL_Redbook_Ord...	✓	Mon January 11, 2010 11:23:02 AM	42.625s
Populate_Order_Star_3	Populate Order Star	SQW_TUTORIAL_Redbook_Ord...	✗	Mon January 11, 2010 11:26:19 AM	17.016s
Populate_Order_Star_4	Populate Order Star	SQW_TUTORIAL_Redbook_Ord...	◓	Mon January 11, 2010 11:41:55 AM	6.860s

Instance that you are monitoring: Populate_Order_Star_4

Activity Name	Status	Activity Type	Start Time	Elapsed Time
Parallel_Container_07	◓	ParallelControlFlowA...	Mon January 11, 2010 11:41:56 AM	14.656s
Populate_Branch_Location_Di...	✓	DataFlowActivity	Mon January 11, 2010 11:41:56 AM	13.328s
Populate_Time_Dimension	✓	DataFlowActivity	Mon January 11, 2010 11:41:56 AM	7.204s
Set_date_range_variables	✓	AssignVariable	Mon January 11, 2010 11:41:56 AM	13.579s

Figure 5-61 Monitoring execution

You can also view the application log for any instance. Figure 5-61 on page 161 shows that the Populate_Order_Star_3 instance failed. Examining the log (Figure 5-62) we can find that the input file could not be found.


Troubleshoot problems by reviewing log messages.

Total number of log messages: 229

Number of log messages to display: 200

Message Level	Time stamp	Message
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03802I: Exiting method handleJAVA in EPG traverser.
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03802I: Exiting method processActionNode in EPG traverser.
FINEST	Mon January 11, 2010 11:26:34 AM	SQW030815I: All code units in the TXN block 10 executed successfully. Committing database: GSDB
FINEST	Mon January 11, 2010 11:26:34 AM	
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03260I: Exit status for EPGBlockPseudoTask <Populate_Branch_Location_Dimension> is com.ibm.datatools.etl.epg.impl.EPGStatusImpl@16541654 (exitCode: SUCCESS, lastBlockReturnCode: CONTINUE).
INFO	Mon January 11, 2010 11:26:34 AM	SQW03201I: EPGBlockPseudoTask <Populate_Branch_Location_Dimension> task completed successfully
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03202I: Start of task: EPGBlockPseudoTask <Set_date_range_variables>
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03801I: Entering method processActionNode in EPG traverser.
FINEST	Mon January 11, 2010 11:26:34 AM	
FINEST	Mon January 11, 2010 11:26:34 AM	Executing Activity: Set_date_range_variables
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03801I: Entering method handleJAVA in EPG traverser.
SEVERE	Mon January 11, 2010 11:26:34 AM	Activity Set_date_range_variables failed due to invalid file c:/OrdersAppX/order-date-range.txt. The system cannot find or read from the file specified.
SEVERE	Mon January 11, 2010 11:26:34 AM	SQW03563E: Failure to execute Java class com.ibm.datatools.sqw.util.runtime.AssignVariableRuntimeUnit. Exception: com.ibm.datatools.sqw.util.runtime.SQWUtilRuntimeUnitException: Activity Set_date_range_variables failed due to invalid file c:/OrdersAppX/order-date-range.txt. The system cannot find or read from the file specified. (messages resource bundle not found)
FINEST	Mon January 11, 2010 11:26:34 AM	SQW03802I: Exiting method handleJAVA in EPG traverser.

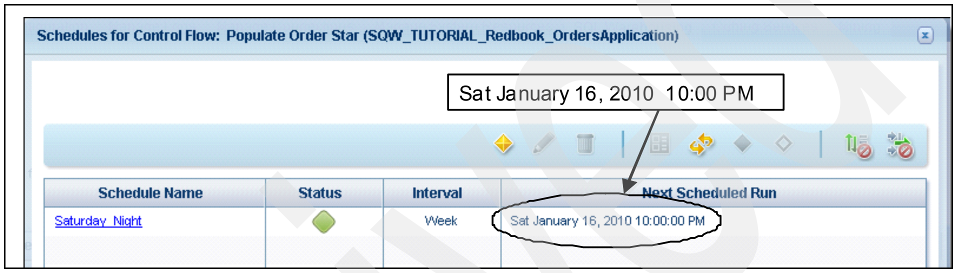
Figure 5-62 Instance execution log

Scheduling control flows

It is good to execute control flows on demand. However, it might be more convenient for control flows to be executed automatically on a scheduled basis. For those, we can create a schedule for executing them.

Working with schedules is done from the Manage Control Flows page, as illustrated in Figure 5-59 on page 159. Perform the following steps:

1. Click **Manage Schedules** to see the schedules for the selected control flow. Figure 5-63 shows that our control flow has an active schedule to execute weekly on Saturday at 10:00 PM.


Schedules for Control Flow: Populate Order Star (SQW_TUTORIAL_Redbook_OrdersApplication)
Sat January 16, 2010 10:00 PM

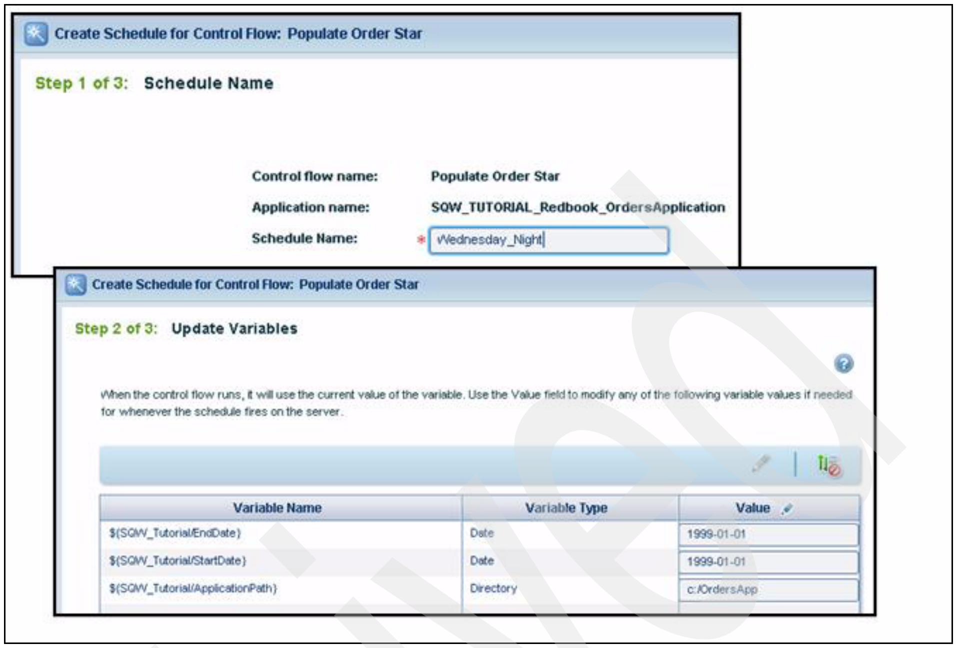
Schedule Name	Status	Interval	Next Scheduled Run
Saturday_Night	🟩	Week	Sat January 16, 2010 10:00:00 PM

Figure 5-63 Schedule list for the Populate Order Star Control Flow

Assume the users of the Orders database want to update the data more frequently adding an update on Wednesday nights as well.

2. Click the gold diamond icon to add a new schedule. Figure 5-64 on page 164 and Figure 5-65 on page 164 shows the dialog steps for adding a new schedule. It calls for specifying a name for the schedule, so we choose Wednesday_Night.
3. Modify variables as needed. The ApplicationPath variable does not change and the other two variables are set dynamically in the control flow from a file, therefore there is no need to modify them.
4. Define the schedule timing. The scheduling choices are extensive. We can schedule by minutes, hourly, daily, weekly and monthly. in each choice there is a lot of flexibility.

   We need to schedule this flow weekly on Wednesday night at 10:00 pm. We end up with two schedules for our control flow with the flow executing twice a week on Wednesday and Saturday nights.



Create Schedule for Control Flow: Populate Order Star

Step 1 of 3: Schedule Name

Control flow name:	Populate Order Star
Application name:	SQW_TUTORIAL_Redbook_OrdersApplication
Schedule Name: *	Wednesday_Night

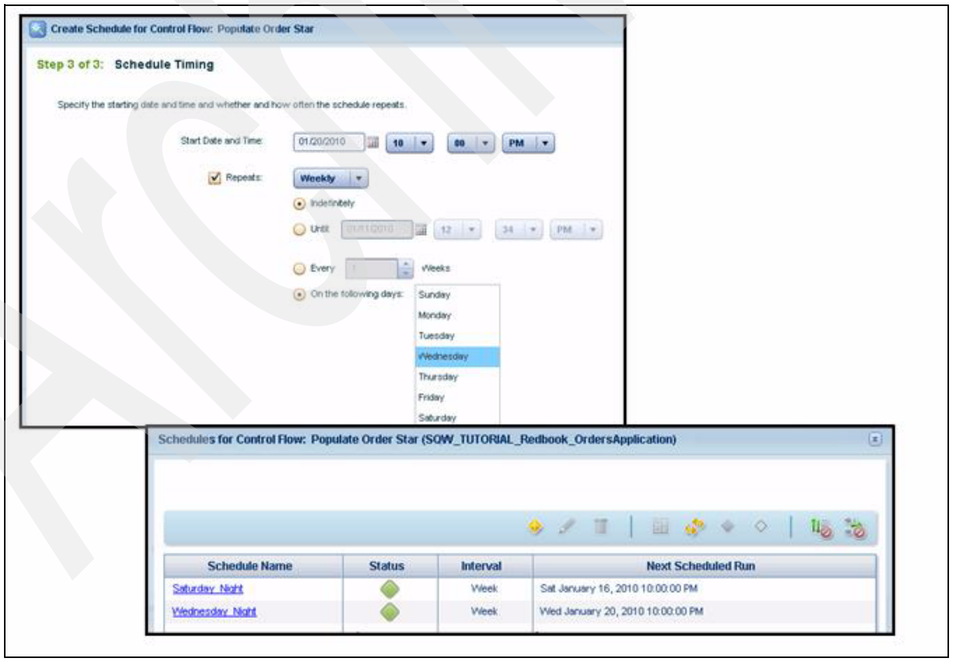
Create Schedule for Control Flow: Populate Order Star

Step 2 of 3: Update Variables

When the control flow runs, it will use the current value of the variable. Use the Value field to modify any of the following variable values if needed for whenever the schedule fires on the server.

Variable Name	Variable Type	Value
${SQW_Tutorial/EndDate}	Date	1999-01-01
${SQW_Tutorial/StartDate}	Date	1999-01-01
${SQW_Tutorial/ApplicationPath}	Directory	c:/OrdersApp

Figure 5-64 Adding a new schedule



Create Schedule for Control Flow: Populate Order Star

Step 3 of 3: Schedule Timing

Specify the starting date and time and whether and how often the schedule repeats.

Start Date and Time: 01/20/2010 10 00 PM

✓ Repeats: Weekly

[✓ Indefinitely]

Until 01/11/2010 12 34 PM

Every 1 Weeks

[✓ On the following days:] Sunday, Monday, Tuesday, [✓ Wednesday], Thursday, Friday, Saturday

Schedules for Control Flow: Populate Order Star (SQW_TUTORIAL_Redbook_OrdersApplication)

Schedule Name	Status	Interval	Next Scheduled Run
Saturday_Night	◆	Week	Sat January 16, 2010 10:00:00 PM
Wednesday_Night	◆	Week	Wed January 20, 2010 10:00:00 PM

Figure 5-65 Adding a new schedule: More steps

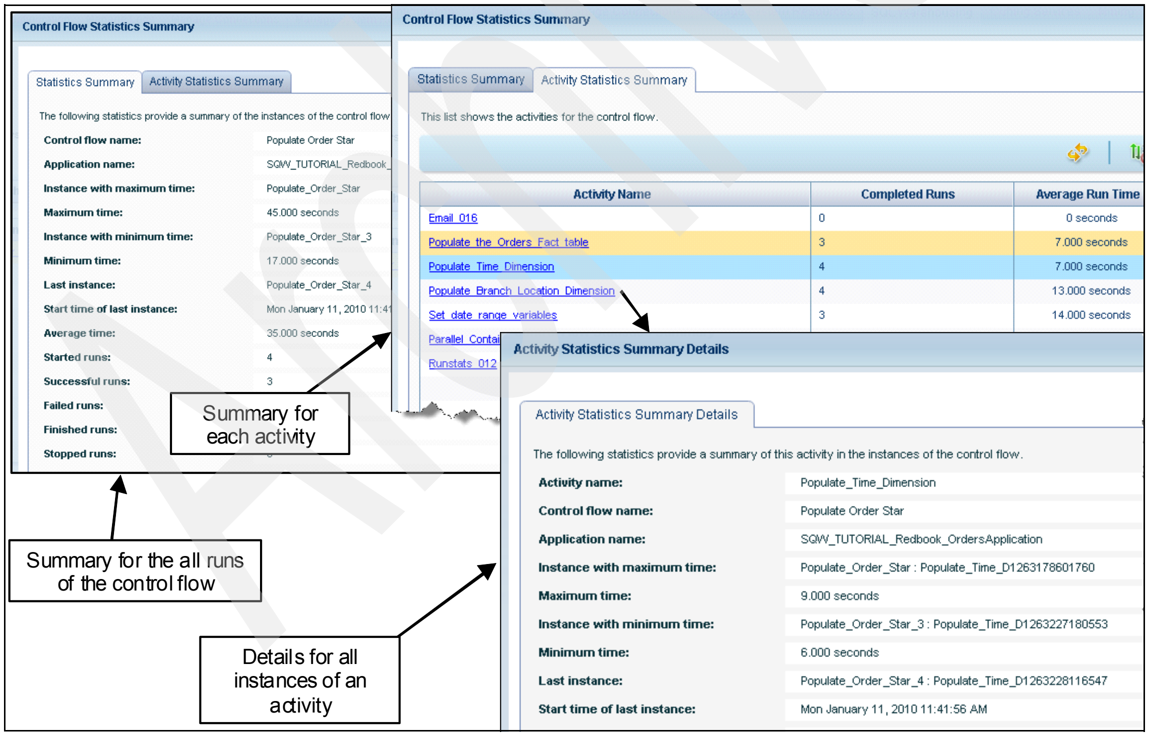
Execution profiles

In our sample control flow there are three variables. One is set at deployment time and does not vary with each execution. The other two are set dynamically during execution by reading the values from a file. What happens if there is a variable that needs to be changed for every execution but is not set dynamically during execution, and, we do not want to execute the control manually? For that you can create an execution profile that holds the variable for a specific schedule of a control flow.

From the Manage Control Flows page you can create one or more profiles specifying the values for the variables. Then when a schedule is created, you can specify to get the values for variables from an execution profile. Each execution of the control flow from this particular schedule will then obtain the variable values from the same profile.

Viewing statistics

It might be useful to monitor the execution statistics of the warehouse application. Statistics can help us understand how long activities should execute and how that changes over time. Statistics are kept at two basic levels. Statistics at the control flow level is a summary averaged over all execution instances of that control flow. Figure 5-66 shows the control flow level statistics for our sample control flow.



Control Flow Statistics Summary
Statistics Summary
Activity Statistics Summary
Summary for each activity
Summary for the all runs of the control flow
Details for all instances of an activity
Control Flow Statistics Summary
Statistics Summary
Activity Statistics Summary
Activity Statistics Summary Details
Activity Statistics Summary Details

The following statistics provide a summary of the instances of the control flow

Control flow name:	Populate Order Star
Application name:	SQW_TUTORIAL_Redbook_
Instance with maximum time:	Populate_Order_Star
Maximum time:	45.000 seconds
Instance with minimum time:	Populate_Order_Star_3
Minimum time:	17.000 seconds
Last instance:	Populate_Order_Star_4
Start time of last instance:	Mon January 11, 2010 11:41
Average time:	35.000 seconds
Started runs:	4
Successful runs:	3
Failed runs:	
Finished runs:	
Stopped runs:	

This list shows the activities for the control flow.

Activity Name	Completed Runs	Average Run Time
Email_016	0	0 seconds
Populate_the_Orders_Fact_table	3	7.000 seconds
Populate_Time_Dimension	4	7.000 seconds
Populate_Branch_Location_Dimension	4	13.000 seconds
Set_date_range_variables	3	14.000 seconds
Parallel_Contai		
Runstats_012		

The following statistics provide a summary of this activity in the instances of the control flow.

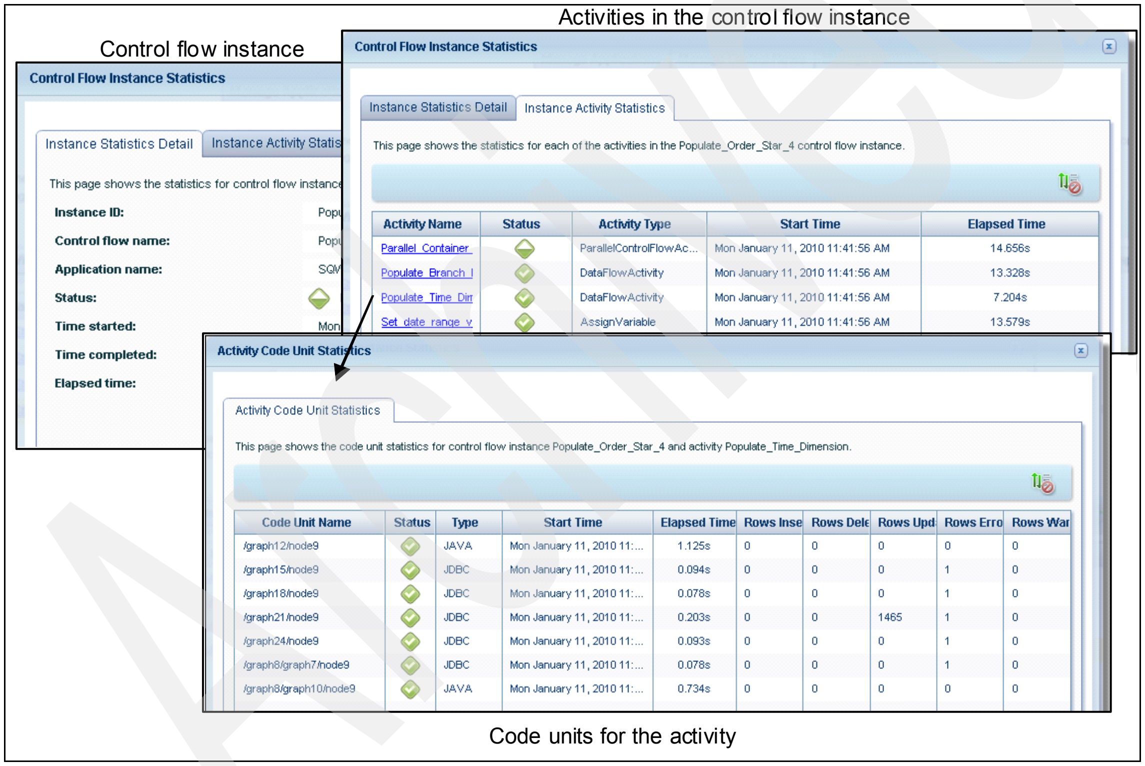
Activity name:	Populate_Time_Dimension
Control flow name:	Populate Order Star
Application name:	SQW_TUTORIAL_Redbook_OrdersApplication
Instance with maximum time:	Populate_Order_Star : Populate_Time_D1263178601760
Maximum time:	9.000 seconds
Instance with minimum time:	Populate_Order_Star_3 : Populate_Time_D1263227180553
Minimum time:	6.000 seconds
Last instance:	Populate_Order_Star_4 : Populate_Time_D1263228116547
Start time of last instance:	Mon January 11, 2010 11:41:56 AM

Figure 5-66 Control flow level statistics

The control flow was executed four times, with one failure. On average it took 35 seconds to execute. We can monitor the average to see if it is increasing, or to compare to instance execution times. We might want to investigate if instance elapsed times are out of bounds compared to the average.

We can also see averages for each of the activities in the control flow, for activity level analysis.

Instance level statistics contain the details for a particular execution instance as shown in Figure 5-67. We can view the statistics for the entire instance or for each activity, and even drill in to an activity for statistics at the code unit level (For example, the number of rows processed for the Populate_Time_Dimension data flow).


Activities in the control flow instance
Control flow instance
Code units for the activity

Control Flow Instance Statistics

Instance Statistics Detail | Instance Activity Statis

This page shows the statistics for control flow instance

Instance ID:	Popu
Control flow name:	Popu
Application name:	SQW
Status:	✓
Time started:	Mon
Time completed:	
Elapsed time:	

Control Flow Instance Statistics

Instance Statistics Detail | Instance Activity Statistics

This page shows the statistics for each of the activities in the Populate_Order_Star_4 control flow instance.

Activity Name	Status	Activity Type	Start Time	Elapsed Time
Parallel_Container_	✓	ParallelControlFlowAc...	Mon January 11, 2010 11:41:56 AM	14.656s
Populate_Branch_I	✓	DataFlowActivity	Mon January 11, 2010 11:41:56 AM	13.328s
Populate_Time_Dir	✓	DataFlowActivity	Mon January 11, 2010 11:41:56 AM	7.204s
Set_date_range_v	✓	AssignVariable	Mon January 11, 2010 11:41:56 AM	13.579s

Activity Code Unit Statistics

Activity Code Unit Statistics

This page shows the code unit statistics for control flow instance Populate_Order_Star_4 and activity Populate_Time_Dimension.

Code Unit Name	Status	Type	Start Time	Elapsed Time	Rows Inse	Rows Dele	Rows Upd	Rows Erro	Rows War
/graph12/node9	✓	JAVA	Mon January 11, 2010 11:...	1.125s	0	0	0	0	0
/graph15/node9	✓	JDBC	Mon January 11, 2010 11:...	0.094s	0	0	0	1	0
/graph18/node9	✓	JDBC	Mon January 11, 2010 11:...	0.078s	0	0	0	1	0
/graph21/node9	✓	JDBC	Mon January 11, 2010 11:...	0.203s	0	0	1465	1	0
/graph24/node9	✓	JDBC	Mon January 11, 2010 11:...	0.093s	0	0	0	1	0
/graph8/graph7/node9	✓	JDBC	Mon January 11, 2010 11:...	0.078s	0	0	0	1	0
/graph8/graph10/node9	✓	JAVA	Mon January 11, 2010 11:...	0.734s	0	0	0	0	0

Figure 5-67 Instance level statistics

5.6 Summary

In this chapter we discussed and described the data movement and transformation component of InfoSphere Warehouse, called SQL Warehousing Tool or SQW. SQW is basically an Extract-Load-Transform (ELT) tool that makes use of the power of the DB2 database engine to move and transform data. Developing data movement and transformation routines, called data flows, is done using a higher-level flow model graphical environment from which code, primarily optimized SQL, is generated, including the code to stage data as necessary.

There is also a powerful flow control and integration mechanism to create applications to run data flows and other types of non-SQL activities in a dynamic flow, handing failure conditions as well as other logical flow conditions. Developing flow control routines is accomplished using a graphical flow development tool called control flows.

A collection of related control flows are organized into a data warehouse application, which is deployed into the runtime environment managed by the web-based administration console. Warehouse application can be executed on-demand or through a schedule, and monitored. Various levels of statistics are also maintained to enable better management of the environment.

6

InfoSphere Warehouse Cubing Services

Online Analytical Processing (OLAP) is a popular and powerful data analytical method, and is a core component of data processing. It gives users the ability to interrogate data by intuitively navigating data from a summary level to a detail level.

InfoSphere Warehouse Cubing Services enables OLAP applications to access terabytes of data through industry-standard OLAP connectivity. This warehouse-based OLAP capability is a core pillar of the InfoSphere Warehouse analytics platform.

In this chapter we introduce the OLAP capabilities provided in InfoSphere Warehouse. An overview of modeling and deploying OLAP functionality using Design Studio is discussed and is referred to as *Cubing Services*.

6.1 Implementing OLAP

OLAP is a key component of many data warehousing and business intelligence projects and often is a core requirement from users. OLAP enables users (business analysts, managers, and executives) to gain insight into data through a fast, consistent, and interactive interface, enabling increased productivity and faster decision making.

6.1.1 Dimensional model

The OLAP style of reporting is typically performed using a specific type of data organization called multidimensional modeling. Multidimensional modeling organizes the data into facts, also called measures and dimensions. Facts are the data values that you want to analyze. Dimensions contain the values of how you want to view the data. In this example, a “sales amount “is a fact that we want to measure, while “store”, “region”, and “time” are the dimensions (or the way in which we want to see the data presented). This is called viewing sales by store, by geography, and by time and is depicted in Figure 6-1. There is likely one star schema for each business subject area.

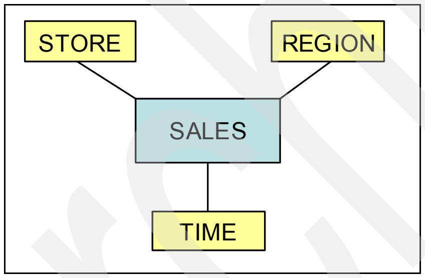


Figure 6-1 A simple star-schema

In Figure 6-1, all of the values from where we want to derive analytical information are contained in a relational table called SALES, which is the fact table. All of the other tables, such as STORE, REGION, and TIME, contain descriptive information, and define how you can view the data. These tables are the dimensions of the star. The data in the fact table refers to the data in a dimension table using the primary key of each dimension table. Fact tables are many times larger than dimension tables. In a star-schema design, the data is denormalized, which results in fewer tables that need to be joined to satisfy a user query.

In addition to the relationship between fact and dimensions, there are data relationships present in a dimension. Attributes in a dimension represent the way in which data is summarized or aggregated. These are organized in a hierarchy. Figure 6-2 shows a typical time hierarchy.

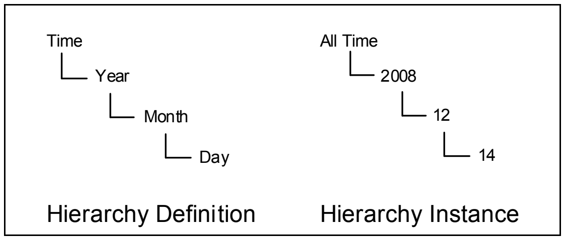


Figure 6-2 A simple time hierarchy

The levels of the hierarchy are where aggregations typically occur. The most common is a simple addition. In the time hierarchy, all of the day values for a month are summarized to become the value for the month, and all of the month values for a year are added to become the value for the year, and all of the year values are added to become the one value represented at the top of the hierarchy. This seems simple until you consider that a dimension is not used in isolation but in conjunction with other dimensions. A report showing sales by month for 2007 for all stores and all geographies might not be as useful as seeing the sales by month for the stores in San Jose, particularly if you are the sales manager for the stores in San Jose. Therefore, you might need to return a value for any combination of the values of any dimension. Although the San Jose city manager might be interested in the sales of San Jose stores for the last three months, the California region manager might want to compare the current month sales to same month sales last year by cities in California, which might require quite a lot of work if you were using a relational database.

6.1.2 Providing OLAP data

Given the requirement to do OLAP, what kind of database engine is needed? One characteristic of OLAP analysis is that the analysis usually begins by looking at aggregated data over a time period. Another important characteristic of OLAP analysis is the exploratory nature of the analysis. That is, we cannot predict the exact path a user might take through the data.

Although relational databases are good at ad hoc types of queries, the ability to deliver highly aggregated data effectively has traditionally been an issue. It is an issue that has led to the development of speciality database engines that delivered highly-aggregated data with good response times. These specialty databases used special storage mechanisms and pre-aggregated the data along all dimensions using highly-indexed systems to access specific cells of aggregated data across the various dimensional values directly. These types of databases were called *Multidimensional OLAP databases* (MOLAP).

Although MOLAP systems delivered highly-aggregated data across dimensions with impressive response times, there was definitely an associated cost. These databases required that data be extracted from the data warehouse and loaded into the MOLAP database, duplicating the data. MOLAP databases also required that the data access paths be pre-calculated across all combinations of dimension members, which resulted in an exponential growth rate in the required data storage. This calculation step was also expensive in terms of the CPU resource cost and elapsed time. However, the MOLAP engines did provide excellent response times and, for example, sophisticated financial analysis.

On the other end of the spectrum, there were products that used the relational database as the OLAP engine. These products as known as *Relational OLAP engines* or *ROLAP engines*. ROLAP products developed a way to provide aggregated data in an effective manner, usually using summary tables. The ROLAP engines determined, either automatically or through user choice, whether to use summary tables or the detail tables to satisfy a query.

The advantage of ROLAP is that it was relational-based, and the data was not copied into another storage mechanism that needed new and different operational procedures. ROLAP data was stored in the relational database and the normal operational procedures applied. However, there was an issue in the area of inconsistent response times.

Relational database technology did not stand still, however. There were dramatic improvements in their ability to handle aggregated data or summary tables. IBM DB2, for instance, has Materialized Query Tables (MQTs). MQTs are summary tables that can be used in multidimensional or OLAP analysis. The power of MQTs in DB2 is that they are transparent to the user or calling program. You can develop the query as though it is going to access the detail, or base tables. The DB2 Optimizer analyzes the query and decides if it costs less to satisfy the query by using the base tables or by using the available MQTs. Even updating the MQT as data is loaded into the data warehouse can be delegated to the relational database engine.

InfoSphere Warehouse Cubing Services takes a hybrid approach to delivering OLAP. From a user or application viewpoint, it appears to be a MOLAP engine using multidimensional expressions (MDX) as the query language. However, the data is not copied or persisted in another storage mechanism, but remains in the DB2 relational engine. Cubing Services then, is what we call a *multidimensional OLAP hot cache*. That is, InfoSphere Warehouse Cubing Services loads data into a multidimensional structure in memory.

That partial MOLAP structure can then deliver fast response times for the data that is in memory cache. If a user of query tool requests data that is not in the cache, the Cubing Services engine generates queries to the relational database to retrieve the data and builds a partial in-memory cube-like structure for only that data. Although the first request for that data sees a longer response time while the in-memory structures are being built, subsequent requests for that data are satisfied from the in-memory structures and experience MOLAP-style response times as long as that data is maintained in memory. Using Cubing Services as the OLAP engine results in a MOLAP-style response in a large percentage of queries without the need to copy the data into another storage mechanism. To improve the relational access, there is an optimization wizard that recommends the appropriate MQTs that can be implemented to improve the response of aggregated data when it is needed from the relational database.

6.1.3 Consuming OLAP data

Having data in an OLAP database is meaningless unless it can be retrieved and displayed to the user in a manner that is relevant and effective for the business requirement. This is an area that is constantly changing to add more capability to how data is turned into information that is timely and relevant for the user. One key aspect of this can be found in one of the terms used for the acronym OLAP, and that is “Online.”

You can certainly perform multidimensional analysis using batch reports. However, it is not uncommon for a user to receive a report and scan through it looking for items of interest based on their experience and knowledge of the business. Often they are looking for items that are out of line with expectations. Additional reports are then run to explore those deviations. Sometimes these reports already exist and can be returned in minutes or hours. In certain cases, a new report has to be defined, which could take days or even weeks, by which time the information might be irrelevant or just plain too late to act upon.

Business decisions these days might have to be made quickly. The user cannot afford to wait hours or days while the next report is developed. In fact, the user needs to see new views of the data while the thought is fresh. Remember, the goal of OLAP is to deliver information at the speed of thought. This does not imply sub-second response time to every data request but that the new view of the data has to be delivered to the user quick enough to keep the train of thought moving forward. This might mean seconds to certain users while minutes might be fine for other users.

Another aspect to OLAP-style analysis is that the user is not submitting report requests but directly interacting with data. This means that user click a data value or graphical element that represents data that might then return more detail behind that particular data value. Or a user might drag additional data attributes onto the OLAP display to add to or take away from values or to pivot the view of the data. The important point here is that the user is interactively engaged directly with the data.

How the OLAP data is displayed to the user and how much interaction is allowed depends entirely on the requirements of the user. Data for a business executive might be displayed entirely using maps and other highly graphical elements, and using color coding to represent business conditions. For example, the VP of sales for the US might see actual sales versus a sales target by using a map of the US with each state color coded to represent how that state is doing against targets. Using the color red to indicate that a state is missing their target would be a common example. The executive might then click a particular state of concern to see a view of the sales territories in that state and so on. A business analyst or knowledge worker might see line or pie graphs accompanied by a grid view of the data, with more freedom to explore by allowing pivoting, and drill down and drill up capabilities. There is a trend to push analytics out to the line-of-business workers giving line management a guided analysis capability of data scoped to be relevant to their part of the business.

In all of these cases, the OLAP display tool must be able to work with the underlying OLAP engine. It may also need to work directly with a relational database to integrate OLAP style analytics with the more traditional relational reporting, and perhaps even integrated in a web portal.

6.1.4 Pulling it all together

So what does all of this mean to the business? It is quite simple. Using multidimensional analysis through OLAP databases and tools gives you much better insight to your business than traditional reporting. It is all about delivering the right information to the right user at the right time in a manner that is relevant to the user to support the decision making process.

It takes a combination of the right multidimensional data model, the right OLAP database engine, and the right OLAP data delivery tooling to make OLAP effective for an organization. It might not always be easy, but when it is done right, the payoff is huge.

In this chapter we focus on the IBM software stack that enables you to implement OLAP using the Cubing Services component of InfoSphere Warehouse.

6.2 Cubing Services architecture

InfoSphere Warehouse Cubing Services is designed to provide a multidimensional view of data stored in relational databases. The Cubing Services architecture includes several core components that are used for designing, developing, testing, deploying, and administering cubes.

6.2.1 Cubing Services core components

In this section, we describe the individual components in the Cubing Services architecture, shown in Figure 6-3.

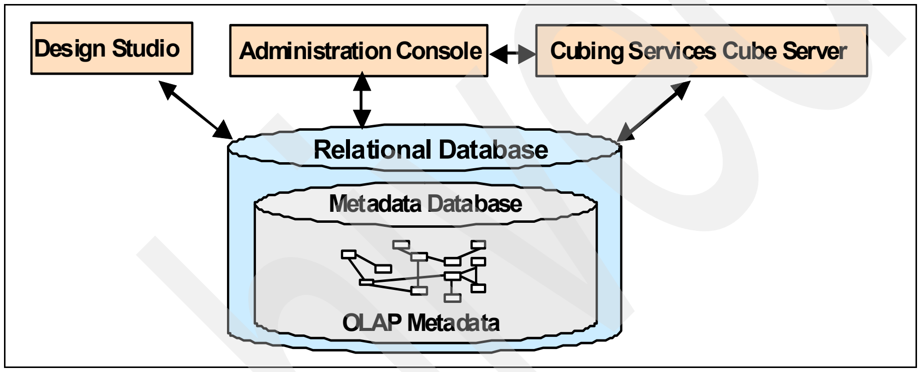


Figure 6-3 Cubing Services architecture: Core components

Design Studio

OLAP cube models and cubes in InfoSphere Warehouse Cubing Services are developed using the Design Studio Integrated Development Environment (IDE). Cube models and the dimensions and hierarchies that comprise the cube model are developed using Design Studio. This improves organizational productivity because the developers need not spend time getting trained on multiple tools for the design, development, and maintenance of data warehouse applications.

Design Studio can also be used to analyze the cube models and invoke the optimization advisor to recommend MQTs for the cube model. The developer needs to execute the SQL scripts generated by the optimization advisor to create the MQTs.

InfoSphere Warehouse Administration Console

The administration console is a web-based tool for managing and monitoring data warehouse applications. Installed on the WebSphere Application Server, the administration console uses web clients to access and deploy data warehouse applications that have been designed and modeled in Design Studio. The Cubing Services functions allow management of cube servers, including configuring caching, management of cube models including importing and exporting cube models, and using the OLAP Optimizer.

Metadata database

The Cubing Services metadata is stored in the metadata database (or repository). This metadata repository exposes the Cubing Services metadata at runtime and allows for unlimited scalability. Objects critical to the operation of Cubing Services are stored in this metadata repository, which provides for a more reliable and robust system implementation.

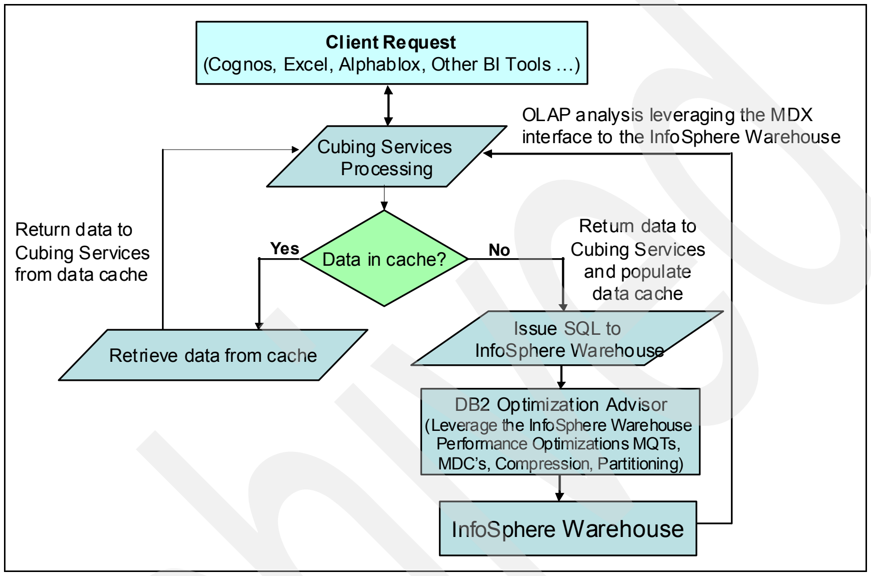
The metadata is accessible from the metadata database in read and write mode to all three clients, the Cubing Services server, Design Studio, and the administration console.

For Cubing Services, this metadata repository provides a relational repository that has the following characteristics:

- It can be accessed by multiple Cubing Services server installations
- It provides all of the industrial strength tools that are available in a database environment for activities such as transactional integrity, backup and restore operations, rollback to a consistent state, and database replication.

Cubing Services Cube Server

The Cubing Services Cube Server is a high performance scalable cubing engine that is designed to support high volumes of queries from many users across multiple cubes. The cube server enables fast multidimensional access to relational data that is referenced by the OLAP cubes defined in the metadata database. The logical flow of analytics in the cube server is shown in Figure 6-4.



Client Request
(Cognos, Excel, Alphablox, Other BI Tools …)
OLAP analysis leveraging the MDX interface to the InfoSphere Warehouse
Cubing Services Processing
Return data to Cubing Services from data cache
Yes
Data in cache?
No
Return data to Cubing Services and populate data cache
Retrieve data from cache
Issue SQL to InfoSphere Warehouse
DB2 Optimization Advisor
(Leverage the InfoSphere Warehouse Performance Optimizations MQTs, MDC's, Compression, Partitioning)
InfoSphere Warehouse

Figure 6-4   Logical flow of analytics in the cube server

The Cubing Services Cube Server relies on the relational data source to store the persistent low-level data. It uses the performance optimization features in InfoSphere Warehouse to improve the performance of data access.

This low-level and summarized data is fetched into the cube server memory as needed. This in-memory caching adds another performance optimization layer to the cubes.

When a cube server starts, it loads the cubes that are configured to start with the cube server. For each of the cubes registered in the cube server, all the dimensional metadata is stored either on the disk or in memory when the cube loads.

This metadata consists of the following information:

- High-level information about the cube, including the following elements:
  - its dimensions and hierarchies
  - levels, dimension members
  - member attributes
  - calculated members
  - calculated metrics
- The mapping of these multidimensional objects to relational database tables and columns

These concepts are shown in Figure 6-5.

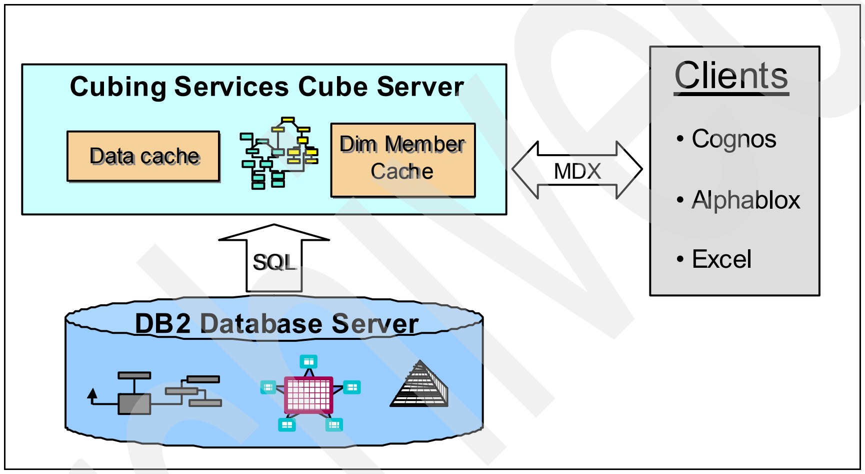


Figure 6-5 InfoSphere Warehouse multi-tier Cubing Services configuration

6.2.2 Software architecture

Each of the components in the Cubing Services architecture are physically located in either the data server, application server, or on the client machines. The Cubing Services Cube Server is a Java process that is used as the run-time for the cube model deployed onto the InfoSphere Warehouse. This Java process manages the caching layer and handles all user requests for data from the client tools.

The InfoSphere Warehouse Administration Console, a Web-based administration tool, runs on the WebSphere Application Server that comes with the InfoSphere Warehouse suite. Figure 6-6 on page 179 provides a pictorial view of the architecture.

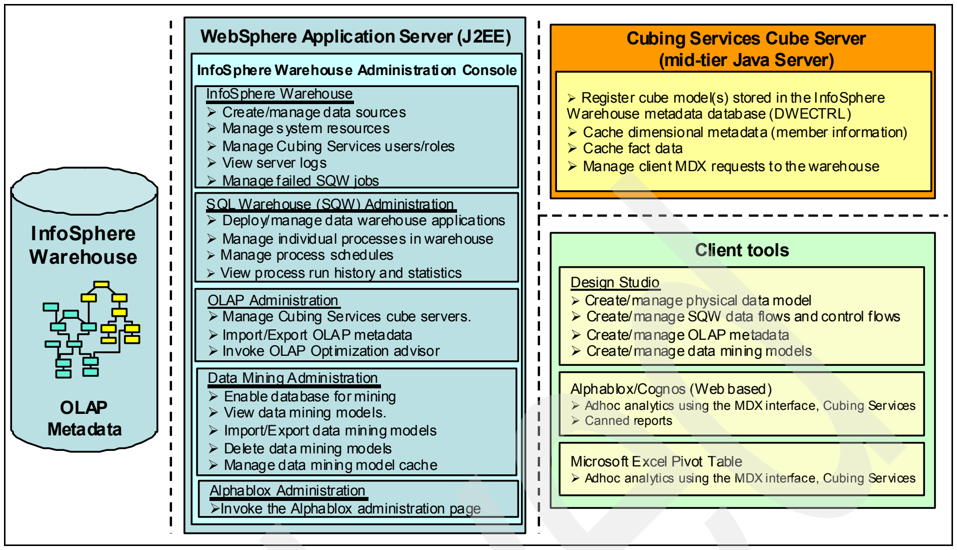

InfoSphere Warehouse
OLAP Metadata
WebSphere Application Server (J2EE)
InfoSphere Warehouse Administration Console
InfoSphere Warehouse
➢ Create/manage data sources
➢ Manage system resources
➢ Manage Cubing Services users/roles
➢ View server logs
➢ Manage failed SQW jobs
SQL Warehouse (SQW) Administration
➢ Deploy/manage data warehouse applications
➢ Manage individual processes in warehouse
➢ Manage process schedules
➢ View process run history and statistics
OLAP Administration
➢ Manage Cubing Services cube servers.
➢ Import/Export OLAP metadata
➢ Invoke OLAP Optimization advisor
Data Mining Administration
➢ Enable database for mining
➢ View data mining models.
➢ Import/Export data mining models
➢ Delete data mining models
➢ Manage data mining model cache
Alphablox Administration
➢Invoke the Alphablox administration page
Cubing Services Cube Server (mid-tier Java Server)
➢ Register cube model(s) stored in the InfoSphere Warehouse metadata database (DWECTRL)
➢ Cache dimensional metadata (member information)
➢ Cache fact data
➢ Manage client MDX requests to the warehouse
Client tools
Design Studio
➢ Create/manage physical data model
➢ Create/manage SQW data flows and control flows
➢ Create/manage OLAP metadata
➢ Create/manage data mining models
Alphablox/Cognos (Web based)
➢ Adhoc analytics using the MDX interface, Cubing Services
➢ Canned reports
Microsoft Excel Pivot Table
➢ Adhoc analytics using the MDX interface, Cubing Services

Figure 6-6 Software architecture

6.3 The Cubing Services life cycle

Based on the InfoSphere Warehouse Cubing Services architecture, the following sections describe the steps in the cubing life cycle consists of the following elements:

- Identify business requirements
- Designing cube models and cubes
- Validating cube with business users
- Deploying cube models

6.3.1 Identify business requirements

Before embarking on a business intelligence (BI) tool acquisition or an analytics solution, the enterprise needs to ask fundamental questions around the data they have available and their business needs. These questions help validate the effectiveness of an OLAP solution, and are a precursor to the actual implementation tasks.

With the assumption that a business problem or opportunity exists and that the organization has collected data over the years that can help them solve the business problem, of the following questions might be asked of the business users to help validate an OLAP solution:

- What type of reporting do you do?
- Do you provide more ad hoc or more canned reports?
- Do you have large amounts of data you use for your analysis?
- Are the current reports slow?
- Does the current process require you to go to IT to create new views of data?
- Does the current process constrain you from doing what-if analyses?
- Does the current process prevent you from inputting data and immediately reviewing the effects on the rest of the business?

If the answer to most or all of the above questions is yes, then OLAP is a good solution that can help businesses make use of their data and perform independent analyses.

OLAP helps organize data as dimensions and metrics or measures. This allows the user to create ad hoc reports quickly without needing to go back to IT for data preparation. That is, the user is able to manipulate the data, pivot the data, and aggregate the same data as different hierarchical views.

The performance of OLAP cubes is much faster than having to aggregate and merge data from multiple relational queries, and is realized by using the performance optimization features in the InfoSphere Warehouse.

The performance optimization features that can be used to improve the performance of Cubing Services cubes includes MQTs, multidimensional clustering (MDC), compression, and data partitioning.

6.3.2 Designing cube models and cubes

InfoSphere Warehouse Cubing Services allows the OLAP metadata to be organized as cube models and cubes in the metadata repository. Upon validating the need for an OLAP solution, the next step is to identify the dimensions and the metrics that are required for analysis by business users.

It is important to have the data organized in a usable format prior to defining the dimensions and measures required for reporting. Let us look at design considerations for the database schema in an OLAP solution. A typical database is made up of one or more tables. The relationships among all the tables in the database are collectively called the database schema. Although there are many different database schema designs, the databases used for querying and reporting on historical data are usually set up with a dimensional schema design.

A dimensional schema design is typically a star schema or a snowflake schema. There are two primary benefits for using these types of designs:

- Easily form queries that answer business questions. As an example, a typical query might calculate a measure of performance over several dimensions.
- Minimize the necessity to form these queries in the SQL language used by most RDBMS vendors.

A dimensional schema physically separates measures (also know as facts) that quantify a business from the descriptive elements (also known as dimensions) that describe and categorize the business. A physical schema is typically represented in the form or a star or snowflake schema where the objects in the star or snowflake are actually database tables or views.

Star schema

A star schema is a type of relational database schema that is made up of a single, centralized fact table surrounded by dimension tables. A star schema can have any number of dimensional tables. The dimension and fact tables have a one-to-many relationship, as shown in Figure 6-7.

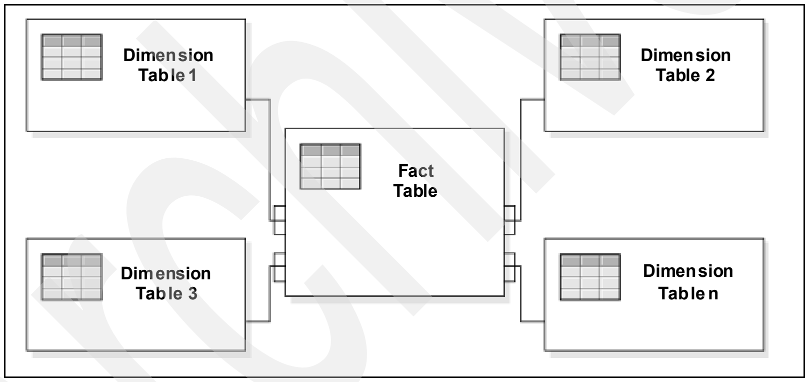


Figure 6-7 Illustration of a star schema

Snowflake schema

A snowflake schema, sometimes called a snowflake join schema, consists of one fact table connected to many dimension tables. These dimension tables can in turn be connected to other dimension tables. Individual levels in a hierarchy can be organized as separate tables in a snowflake schema. This is shown in Figure 6-8.

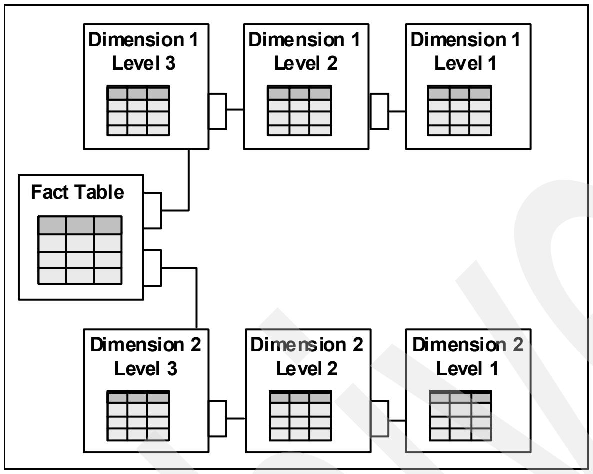


Figure 6-8 Illustration of a snowflake schema

Once the dimensional schema is finalized you can nominate the tables and columns to be used in defining dimensions and measures.

Note: Having a star schema or snowflake schema data warehouse not only simplifies the data model but it also allows you to achieve the desired performance and data quality in your analytics.

6.3.3 Validating the cube with business users

Now that we have a cube model, the next step is to validate the cube in the cube model with the business users. The following questions and checks need to be considered:

- Is the user able to create all the required canned reports using the dimensions in the cube?
- Do the cube dimensions have all the necessary hierarchies and levels?

  The hierarchy and level definitions are based on the type and content of reports being used by the business.
- Are the aggregations for the individual measures defined correctly?

  For example, sales might be summed across certain dimensions, whereas average sales is averaged across the another set of dimensions.

This level of validation can be performed by using client tools that can access InfoSphere Warehouse Cubing Services cubes, such as Cognos, Alphablox, and Microsoft Excel. These tools are discussed in IBM Redbook *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582.

6.3.4 Deploying cube models

When the structure of the cube has been validated by the business users in the test environment, the next step is to deploy the cube model into the production environment. However, all cube models need to be deployed to the InfoSphere Warehouse server before being exposed to the client access tools.

Cube models can be deployed by either using the Design Studio interface or the Web-based InfoSphere Warehouse Administration Console.

Deploying the cube model with Design Studio

If you choose to deploy the cube model using Design Studio, perform the following steps:

1. Analyze the cube model. This is the process of validating the cube model based on predefined rules. The developer can also selectively validate the cube model against a portion of the available rules.
2. Determine the InfoSphere Warehouse server to which the cube model is to be deployed.

Deploying the cube model with the administration console

If you choose to use the administration console to deploy a cube model, the cube model needs to be exported to an XML file using the Design Studio interface. This XML file can be stored either on client machine or on a server from which it can be deployed. The administration console allows the developer to select the XML file that is created and import the cube model onto the cube server.

6.4 Cube development process flow

The cube development process is all about defining the OLAP metadata objects in the InfoSphere Warehouse. OLAP metadata objects are used to describe relational tables as OLAP structures. This is different from working with traditional OLAP objects in that this describes where pertinent data is located and the relationships in the base data. This metadata is stored separately in the InfoSphere Warehouse metadata repository.

Metadata objects act as a base to access data in relational tables directly, while other metadata objects are used to describe relationships between the base metadata objects. All of the metadata objects can be grouped by their relationships to each other, in a metadata object called the *cube model*. Essentially, a cube model reflects a particular grouping and configuration of relational tables. An example is shown in Figure 6-9.

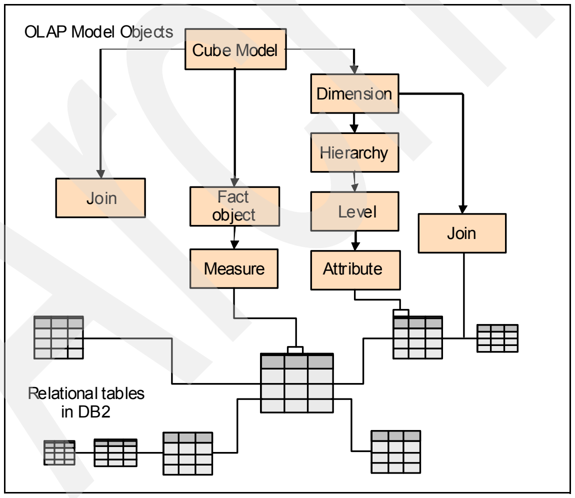


Figure 6-9 OLAP metadata objects in the InfoSphere Warehouse

A cube model in the InfoSphere Warehouse is a logical star schema or snowflake schema. It groups relevant dimension objects around a central fact object. Cube models define a complex set of relationships and can be used to expose relevant fact objects and dimensions to an application selectively.

The cube development process is centered around the definition of the cube model. With the assumption that the business requirements have been identified, the developer goes through the steps shown in the process flow of Figure 6-10, to define the OLAP metadata, optimize access to the underlying data using the OLAP interface, test the validity of the OLAP interface, and deploy it to the user for executing analytics.

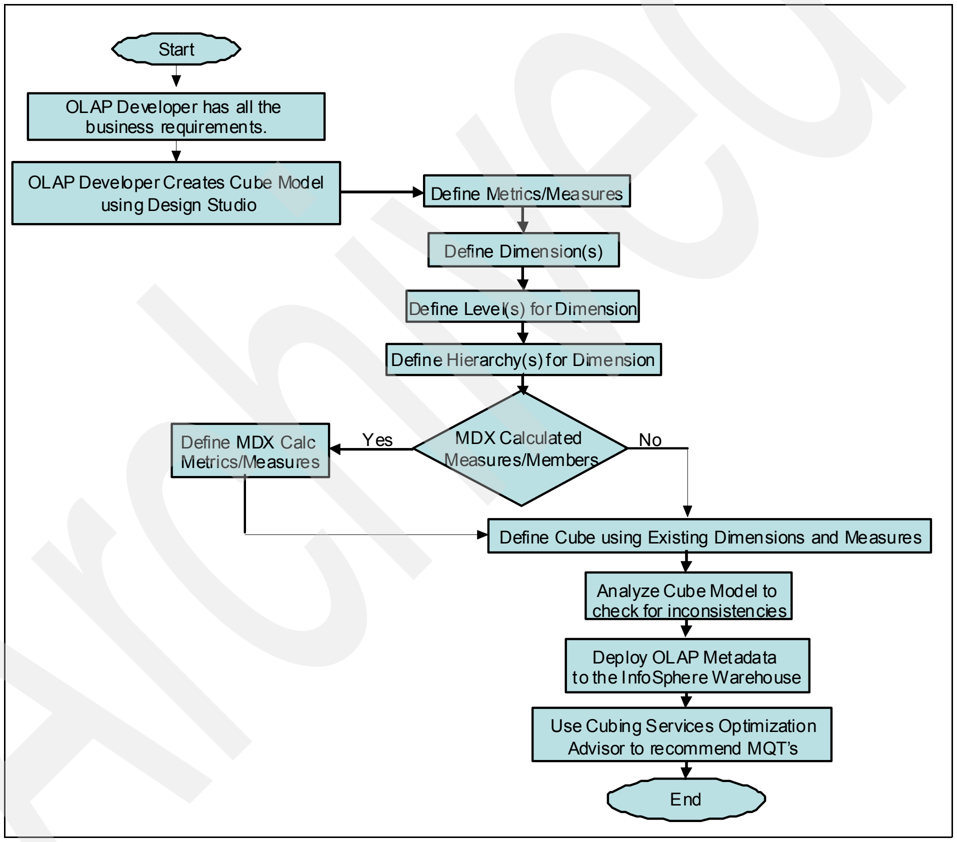

Start
OLAP Developer has all the business requirements.
OLAP Developer Creates Cube Model using Design Studio
Define Metrics/Measures
Define Dimension(s)
Define Level(s) for Dimension
Define Hierarchy(s) for Dimension
MDX Calculated Measures/Members
Yes
No
Define MDX Calc Metrics/Measures
Define Cube using Existing Dimensions and Measures
Analyze Cube Model to check for inconsistencies
Deploy OLAP Metadata to the InfoSphere Warehouse
Use Cubing Services Optimization Advisor to recommend MQT's
End

Figure 6-10 Cube development process flow

In the following sections, we discuss the components that make up a cube model, and the best practices around the development of each of these components.

6.4.1 Fact objects

In a cube model, a fact object is used as the center of the star schema. It groups related measures and metrics that are of interest to a particular application. A fact object contains a set of measures that describe how to aggregate data from the fact table, across dimensions. Measures describe data calculations from columns in a relational table. They are joined to create the fact object.

The fact object references the attributes that are used in fact-to-dimension joins. In addition to a set of measures, a fact object stores a set of attributes and a set of joins. This is shown in Figure 6-11.

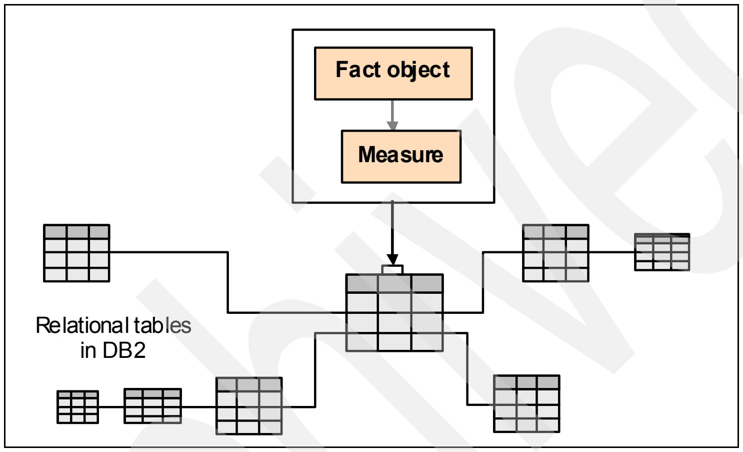


Figure 6-11 Fact object in the OLAP metadata

A fact object has the properties described in the following list:

- Set of measures

  This property is a set of all related measures in a fact object.

- Default measure

  This property is a measure that is selected as a default value for the fact object and that is displayed in vendor applications.

- Set of attributes

  This property is a set of all attributes that are used in the fact object for dimension-fact table joins, and fact-fact table joins.

- Set of joins

  This property is a set of relational database table joins that are needed to join all specified measures and attributes.

6.4.2 Dimensions

Dimensions in cube models are used to organize a set of related attributes that together describe one aspect of a measure. Dimensions in a cube model organize the data in the fact object according to logical categorizations (such as location and time).

Dimensions can reference zero or more hierarchies. Hierarchies describe the relationship and structure of the referenced attributes that are grouped into levels and provides a navigational and computational way to traverse the dimension.

Dimensions also have a type that defines a time-oriented dimension. Defining a dimension to be a time dimension allows for the use of Time Series MDX functions that rely on a time dimension.

The relationship between the dimension and fact objects needs to be defined for each dimension in a cube model. This is the facts-to-dimension join. If a dimension is based on data in a fact table, the dimension is called a degenerate dimension and does not require you to specify a facts-to-dimension join.

Dimensions are automatically detected based on primary key relationships with the specified fact table. Dimensions can be shared across cube models by using the Add Dimension wizard.

A dimension object has properties described in the following list:

- Set of attributes

  This property is a list of attributes that are used in the dimension. Attributes can either be pointers to columns in the dimension table, calculations based on columns in the dimension tables, or even calculations based on other attributes.

- Set of joins

  This property is a set of joins required to obtain all the specified attributes. Only the joins that are required to join the dimension tables are specified here.

- Set of hierarchies

  This property is the set of all the different ways in which the levels of this dimension are organized for analysis. A cube can have multiple hierarchies.

- Set of levels

  This property is the set of all the levels that are used by this dimension.

- Type

  This property identifies the type of this dimension.

6.4.3 Levels

Levels are elements of the dimension definition. A level is a set of attributes that work together as one logical step in the hierarchy definition for a dimension. A level can have four types of attributes:

- Level key attributes

  These key attributes are one or more attributes that uniquely identify each of the members in the level. This is the key column, or a group of columns, identified in the dimension table, that uniquely identifies every row in the dimension table.

- Default related attribute

  This is the attribute that is displayed by default in a reporting application to provide meaningful names for each member in the level. The default related attribute is a required field and must be functionally determined by the level key attributes. The cubing engine internally concatenates the default related attribute with the default related attributes of the higher level members in this hierarchy to generate a unique internal key for this member in the specified hierarchy. If this internally generated key is not unique you receive an error at load time. Keep this in mind as you define your default related attributes.

- Related attributes

  Related attributes are all the optional attributes that provide additional information about the members in a level. These attributes are functionally determined by the level key attributes.

Note: To use the Cognos drill through feature, attributes have to be created with specific names. To support Cognos drill-through, define a member key that uniquely identifies the member using a business key. This value must be defined as a related attribute and follow the naming convention of levelName.MEMBER_KEY. This should correspond to the level key. If the level key is a single attribute, define the member key-related attribute the same way. If the level key contains multiple attributes, define the member key related attribute as a compound expression (such as by using concatenation).

The following code is an example. Note that the Product.MEMBER_KEY is defined the same as the level key Product ID.

```
<attribute name="Product.MEMBER_KEY" schema="ATLAS"
businessName="Attribute">
<sqlExpression template="{$$1}">
<column name="PRODUCT" tableSchema="ATLAS" tableName="PRODUCT"/>
</sqlExpression>
</attribute>

<attribute name="Product ID" schema="ATLAS" businessName="Attribute">
<sqlExpression template="{$$1}">
<column name="PRODUCT" tableSchema="ATLAS" tableName="PRODUCT"/>
</sqlExpression>
</attribute>

<level name="Product" schema="ATLAS" businessName="Product" funcDep="no"
type="regular">
<levelKeyRef name="Product ID" schema="ATLAS"/>
<defaultAttributeRef name="Product" schema="ATLAS"/>
<relatedAttributeRef name="Product.MEMBER_KEY" schema="ATLAS"/>
<orderAttributeRef name="Product" schema="ATLAS" order="asc"/>
</level>
```

- Ordering attributes

  Ordering attributes are optional attributes that provide additional information about the ordering of members in a level, regardless of what hierarchy the level is in.

Note: Level keys play an important role in the performance of the cubes. Keep the size of the level keys as small as possible. In database terms, it is better to have a single column primary key in the dimension tables than to have a composite key to distinguish members in each level.

6.4.4 Hierarchies

A hierarchy defines the relationships among a set of attributes that are grouped by levels in the dimension of a cube model. More than one hierarchy can be defined for a given dimension in the cube model. Hierarchies are a grouping of levels in the dimension.

A hierarchy can have an All level as the top most level in the hierarchy, which is a single member that represents the aggregation of all of the members in the levels below in the hierarchy.

There are several types of hierarchies:

- Balanced
- Unbalanced
- Ragged

In a balanced hierarchy, every leaf member has exactly the same number of ancestors. In an unbalanced hierarchy, this is not the case because members at the leaf level, or interior levels, might be missing. Where interior level members are missing in the hierarchy, it is called a ragged hierarchy. For more detail on hierarchy types, see the IBM Redbook *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582.

6.4.5 Measures

Measures define a measurable entity and are used in fact objects. Measures can be defined as columns in the fact table or they can be calculated measures. A calculated measure could be an SQL expression with an existing column in the fact table or it could be an MDX expression.

An aggregate function is used to summarize the value of the measure for dimensional analysis. This could be any aggregate function, such as sum, average, maximum, and minimum.

Measures become meaningful only in the context of a set of dimensions in a cube model. As an example, a revenue of 300 is not meaningful. However, when you say the revenue for the clothing line in the eastern region is 300, it becomes meaningful.

A measure is defined by a combination of two properties, an SQL expression list and an aggregation list. Table columns, attributes, and measures are mapped to a template to build SQL expressions. The resulting SQL expression is subject to the first aggregation function of the measure. If a measure has more than one aggregation, the aggregation functions are performed in the order in which they are listed with each subsequent aggregation taking the previous aggregations result as the input. If a measure is defined as a calculation of another measure, the aggregation is optional for the measure as it inherits the aggregation from the other measure being used.

6.4.6 Attributes

An attribute represents the basic abstraction of a database column. It contains a SQL expression that can either be a simple mapping to a table column or a more complex expression. The more complex expressions can combine multiple columns or attributes and can use all SQL functions, including user-defined functions when necessary.

Design Studio hides much of the complexity of the attribute object definition. In Design Studio, you do not need to define the expression template or parameter list of the attribute explicitly. If you want to create an attribute that maps directly to a column in the table, select the source column when creating the attribute in the Dimension Properties window. To create a calculated attribute, use the SQL expression builder to create the source expression. The SQL expression builder provides a list of available attributes, columns, operators, functions, and constants.

6.5 Cubing Services and query optimization

Cubing Services cube models are a multidimensional overlay on relational data. The Cubing Services Cube Server does not provide any persistence of cube data objects in multidimensional form. Instead, it relies on the relational data source to store the persistent low-level and aggregated data, which is fetched into the Cube Server memory only as needed. This means that the performance of the cube models is dependant on both the caching layer in Cubing Services and the performance optimization layer in the InfoSphere Warehouse. With the dimensional metadata, when the cube server starts it queries the underlying relational data source and cache the metadata on the Cube Server.

The performance optimization layer in InfoSphere Warehouse needs to be built with consideration for the cube model that is designed and the data that is exposed using the Cubing Services cube model. The Cubing Services Optimization Advisor helps optimize the star or snowflake schema, and improve the performance of the OLAP-style SQL queries. This optimization process includes creating, implementing, and maintaining the MQTs recommended by the Optimization Advisor.

6.5.1 MQTs

The Cubing Services Optimization Advisor helps optimize cube models by recommending MQTs. MQTs can improve query performance because they contain precomputed results from one or more tables that can be used in a query. Costly table joins and complex calculations can be computed in advance and stored in a MQT so that future queries that use these aggregations can run much faster.

The DB2 Query Optimizer is able to recognize that a specific query requires an aggregate and, if it has a relevant MQT available, can rewrite the query to run against the MQT instead of the base data.

Figure 6-12 shows a DB2 graphical explain output for a query with no MQT and with an MQT. In both cases, the same SQL query is issued against the base fact table. The data comes from the fact table in the scenario with no MQT. When an MQT is present, DB2 automatically rewrites the query to use the MQT instead of the base fact table.

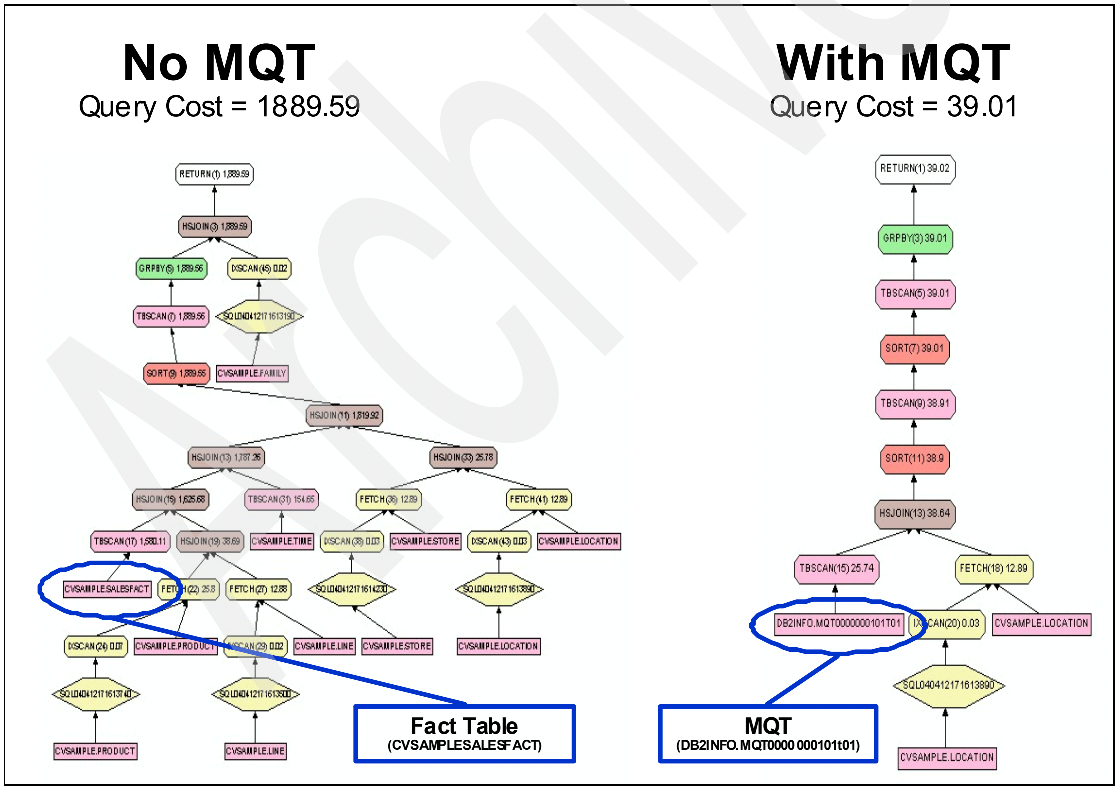


Figure 6-12 MQT explain comparison

The explain output in the figure is not readable because of the complexity of the output and the small size of the figure, but that is not really required. It is convincing enough just to see the difference in the explain structure and path. The structure is smaller and the path and options are much more simple when the MQT is used as opposed to not using the MQT. The result is that the query cost is dramatically lower when the MQT is used, as seen in the figure. The final result is that the query performance can be improved by orders of magnitude because values are already pre-calculated.

The DB2 Query Optimizer does not require the queries to be exact matches of an MQT. The optimizer might determine that a portion of the query can be resolved using an MQT. The optimizer can also use lower level aggregate MQTs to calculate higher level aggregates.

Important Tip: After MQTs are defined and built in DB2, users and BI tools should not construct queries to run directly against them. The SQL queries should be constructed against the base tables and not the MQTs. DB2 determines how the query should be rewritten and the optimal access path. In this manner, database administrators can drop and recreate new MQTs to reoptimize the environment as the data warehouse evolves with new tables and more data and different query workloads. When a major update in the OLAP metadata occurs, re-run the Cubing Services Optimization Advisor to determine whether new MQTs are recommended.

6.5.2 Cubing Services Optimization Advisor

Cubing Services provides an optimization advisor that recommends MQTs based on the OLAP metadata and environmental information, as shown in Figure 6-13. This relieves the database administrator from having to spend time and effort trying to determine the most effective MQTs to create.

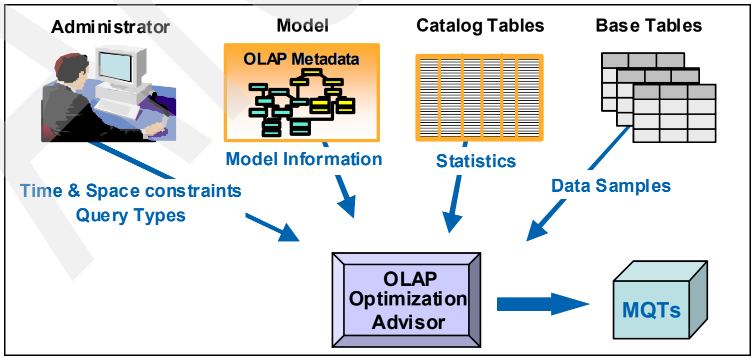


Figure 6-13 Cubing Services Optimization Advisor

Several factors are taken into consideration by the Cubing Services Optimization Advisor to determine the best MQT (or set of MQTs) that can be recommended for a given environment:

- Time: How long do you want the advisor to think?
- Space: How much disk space do you want to provide for the MQTs?
- Cube model: What dimensions, hierarchies, attributes, and measures are there?
- Data: The advisor gets useful information from the DB2 statistics for each table, so statistics must be up-to-date. The advisor can also optionally get useful information from sampling.

Important: The optimization advisor is affected by the catalog statistics and sample data. Therefore, MQT recommendations are different between a test and production system if the catalog statistics and data vary greatly.

After running the optimization advisor, scripts to create and update the MQTs are generated. These can be run immediately or scheduled to run during a preferred time period. Database administrators (DBAs) do not need to understand and write any SQL. However, a DBA can choose to modify the scripts before they are run.

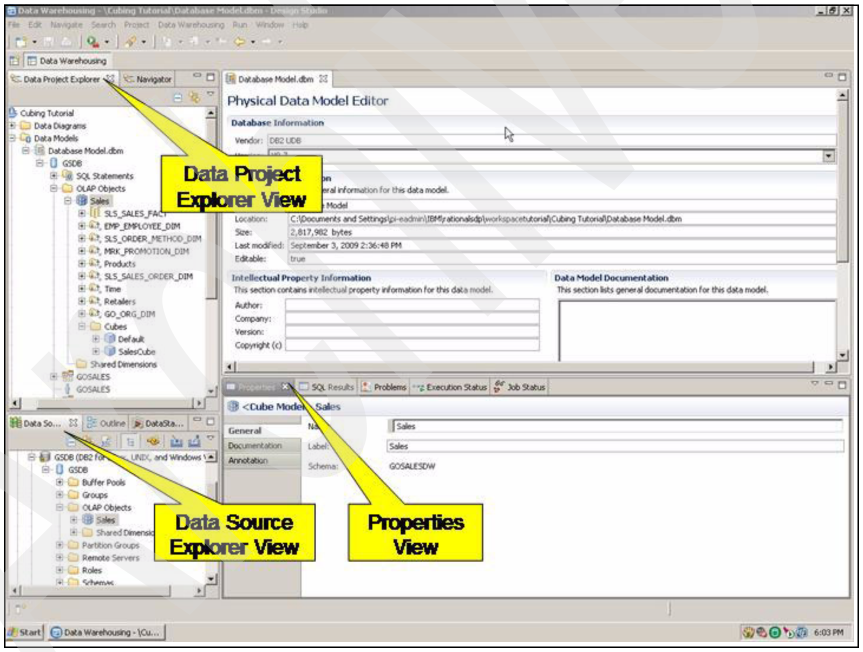
Important: The initial MQT build can take several hours and consume many computing resources. Run the scripts during periods of low activity.

6.6 Modeling OLAP with InfoSphere Warehouse Design Studio

In this section we discuss modeling OLAP with InfoSphere Design Studio, which provides a common integrated data warehousing design environment.

Note: Review and be familiar with Chapter 3, "InfoSphere Warehouse Design Studio" on page 29 and Chapter 4, "Developing the physical data model" on page 53 before you begin working with Cubing Services.

In Design Studio, creating an OLAP model primarily takes place using the Data Source Explorer, Data Project Explorer, and the Properties View, as seen in Figure 6-14. This is a busy figure, and you are not expected to read the contents. It is to give you a perspective of the segments in Design Studio. The details of those segments are presented in subsequent figures.



Data Project Explorer View
Data Source Explorer View
Properties View
Physical Data Model Editor

Figure 6-14 Design Studio OLAP views

6.6.1 Data Source Explorer

The Data Source Explorer provides a tree view that allows the user to browse the catalog contents of a relational database. Initially a connection to a database is created, then the connection node can be expanded to reveal a database icon, which can be further expanded to display nodes representing the catalog contents of the database, as shown in Figure 6-15.

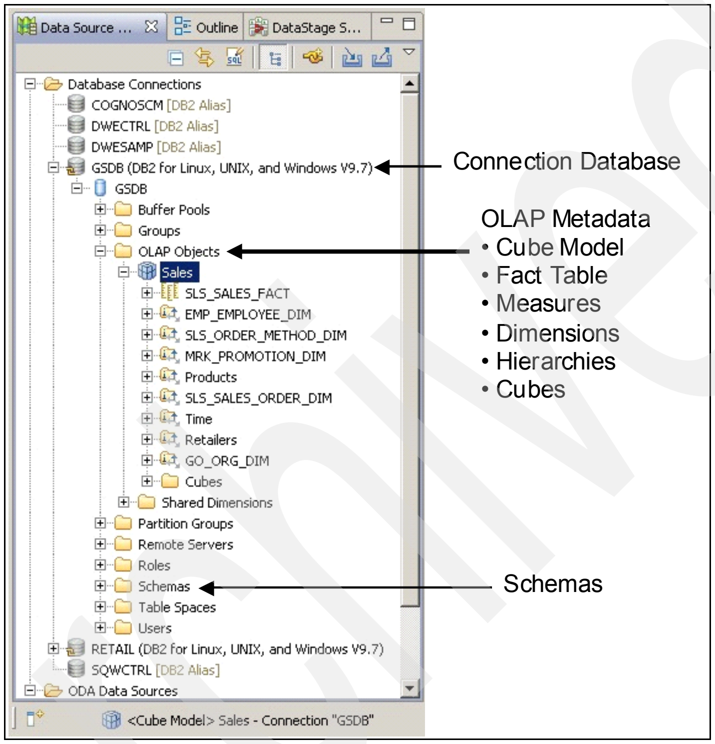


Figure 6-15 Data Source Explorer: OLAP metadata

in the Data Source Explorer, existing OLAP objects can only be viewed. However, OLAP objects can be imported from an XML file and created as new physical objects in the Data Source Explorer. Otherwise, creating new OLAP objects takes place in the Data Project Explorer.

6.6.2 Properties View

The Properties View, as shown in Figure 6-16, allows the users to view the properties of the selected object. The Properties View is applicable to objects in either the Data Source Explorer or the Data Project Explorer. Properties for objects selected in the Data Source Explorer are read only. Properties for objects selected in the Project Explorer are editable.

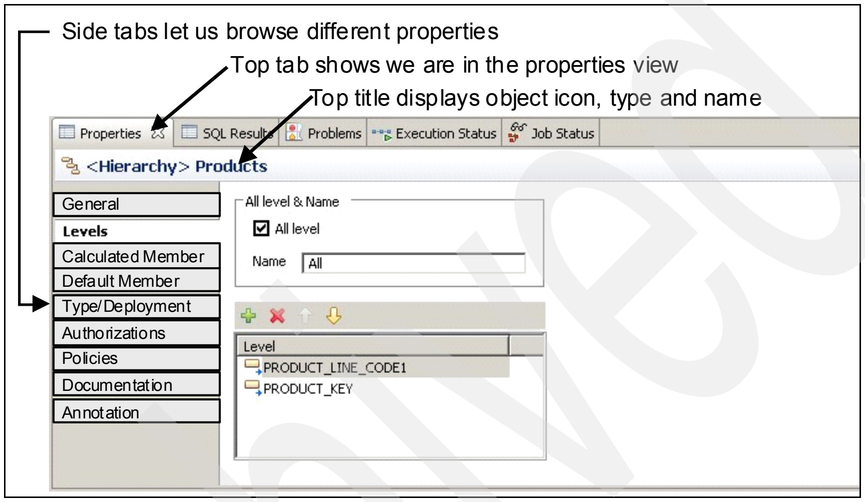


Figure 6-16 Properties view

6.6.3 Data Project Explorer View

The Data Project Explorer View in Design Studio is used to view projects and data models. A project can contain one or more physical data models, and each physical data model is a model of either a complete relational database or a part of a relational database such as a schema and its contents. Physical data models can contain OLAP objects as well as the traditional relational objects such as tables and views, as shown in Figure 6-17. As new projects are created, each project maps to a folder in the Windows file system, and each physical data model maps to a file in the folder.

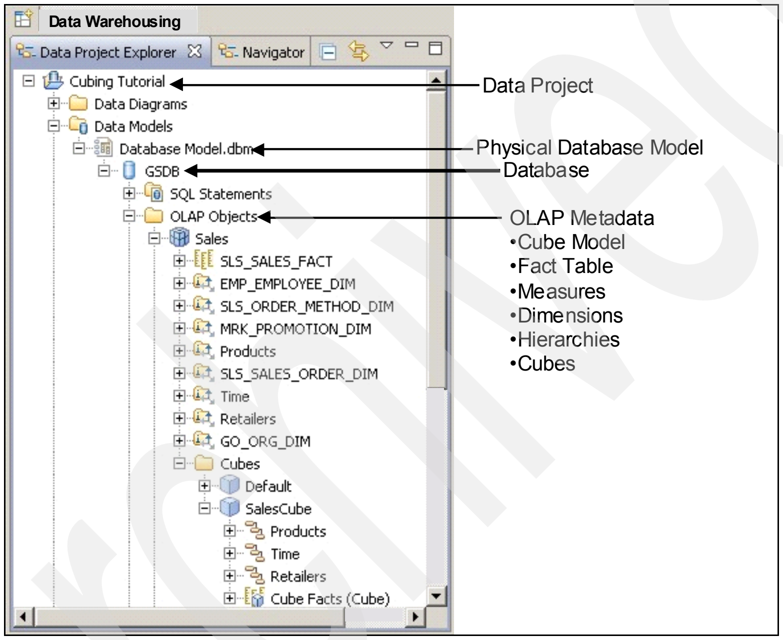


Figure 6-17 Data Project Explorer: OLAP metadata

Using the right mouse button, context menus are available from selected objects in the view to create, delete, and alter the objects. A physical data model is first needed to model OLAP metadata objects.

A physical data model containing OLAP objects can be created with the following approaches:

- Reverse engineering an existing DB2 database containing OLAP objects
- Creating an empty physical model and creating OLAP objects by:
  - Importing objects from an OLAP XML file
  - Using the quick start wizard
  - Using the context menus (right-click)
  - Manually copying objects from a database in the Data Source Explorer using either the clipboard or drag and drop

6.6.4 Creating a new data project

A data project needs to be defined before an OLAP model can be created. There are several types of data projects, but for Cubing Services use the data design project (OLAP).

1. From Design Studio, select **File** → **New** → **Data Design Project (OLAP)**, as shown in Figure 6-18.

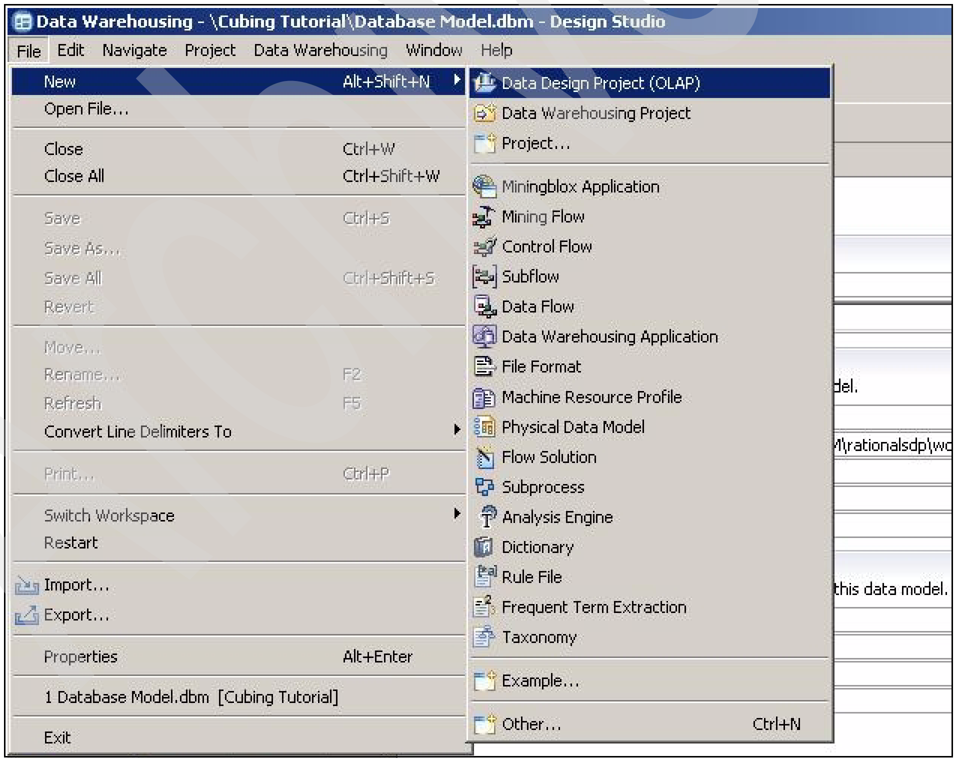


Figure 6-18 New data project

2. Type `Cubing Tutorial` as the project name and click **Finish**.

6.6.5 Creating a new physical data model

Before the OLAP model is created, the physical data model of the underlying base tables (facts and dimensions) should be defined.

1. Expand the Cubing Tutorial project.
2. Right-click the **Data Models** folder and use the context menu to select **New** → **Physical Data Model**, as shown in Figure 6-19.

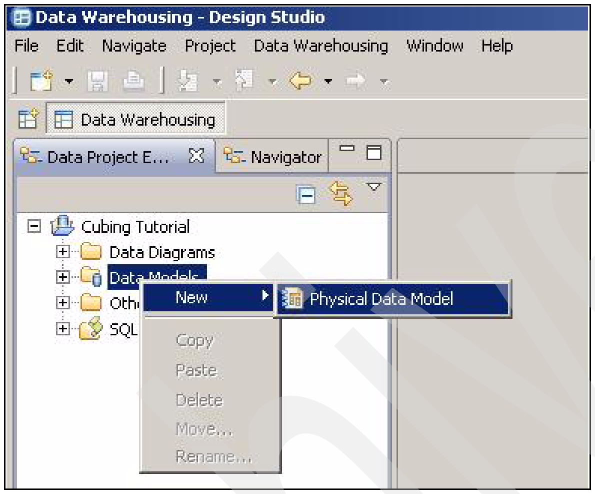
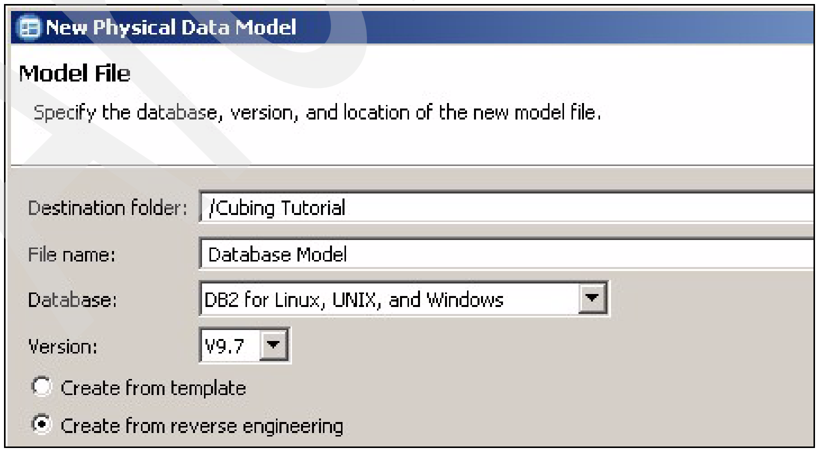


Figure 6-19 New physical data model

3. In the File Name box, give the data model a name.
4. Select the appropriate database and version (DB2 and v9.7 in this example), as shown in Figure 6-20.


New Physical Data Model

Model File

Specify the database, version, and location of the new model file.

Destination folder: /Cubing Tutorial

File name: Database Model

Database: DB2 for Linux, UNIX, and Windows

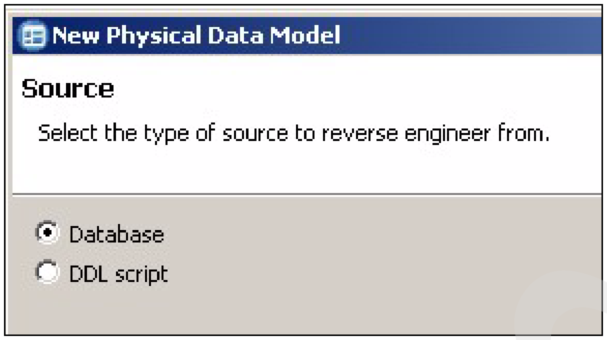
Version: V9.7

○ Create from template

● Create from reverse engineering

Figure 6-20 New physical data model options

5. Select the **Create from reverse engineering** radio button and click **Next**.
6. From the “New Physical Data Model” window (Figure 6-21), select the **Database** radio button. This specifies that an existing database will be used.



New Physical Data Model

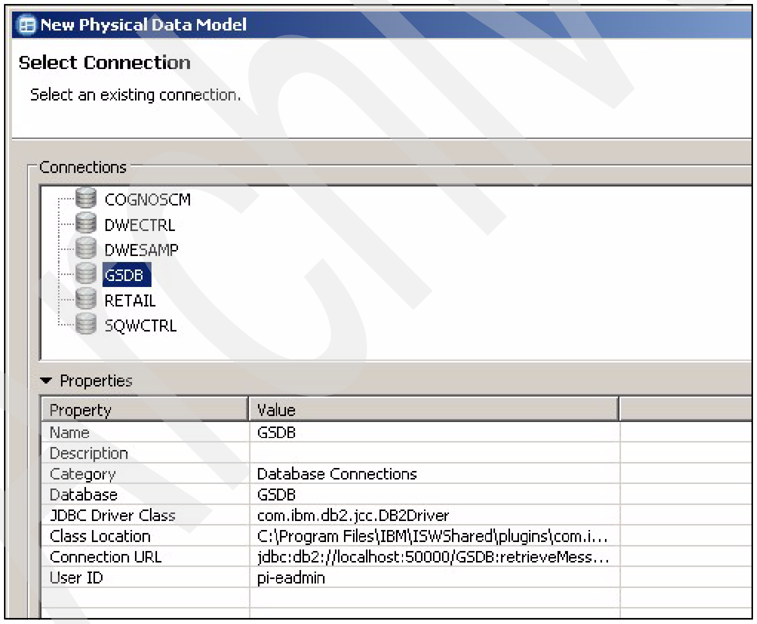
Source

Select the type of source to reverse engineer from.

(● Database) ○ DDL script

Figure 6-21 Reverse engineering options

7. On the “Select Connection” window (Figure 6-22), select **Use an existing connection**, select the **GSDB** connector, and click **Next**.



New Physical Data Model

Select Connection

Select an existing connection.

Connections

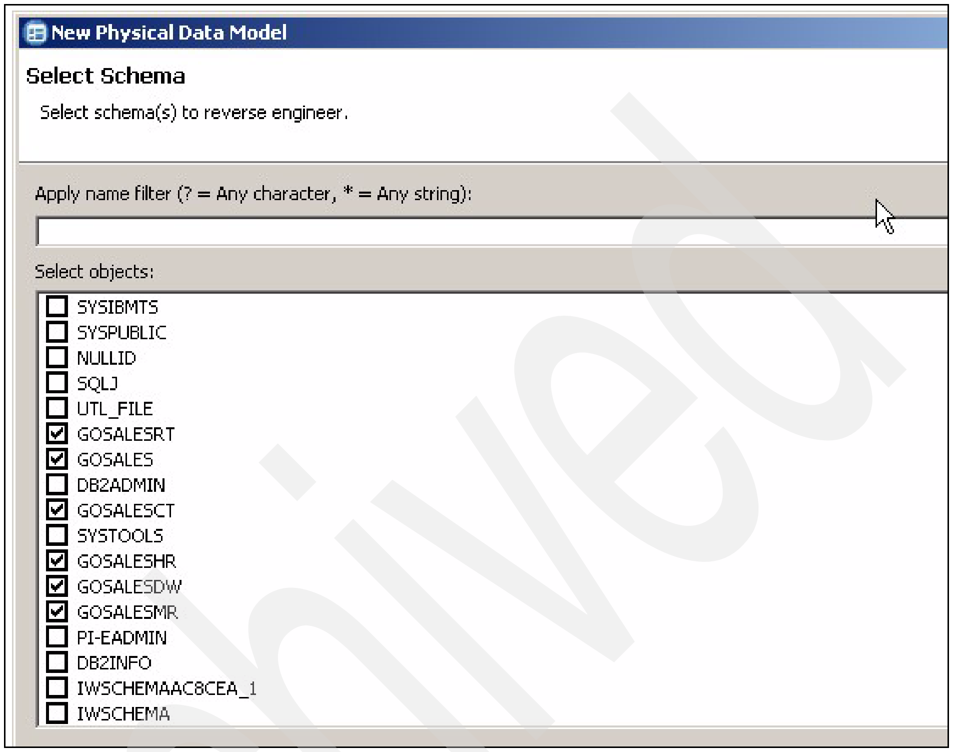
- COGNOSCM
- DWECTRL
- DWESAMP
- GSDB
- RETAIL
- SQWCTRL

Properties

Property	Value
Name	GSDB
Description	
Category	Database Connections
Database	GSDB
JDBC Driver Class	com.ibm.db2.jcc.DB2Driver
Class Location	C:\Program Files\IBM\ISWShared\plugins\com.i...
Connection URL	jdbc:db2://localhost:50000/GSDB:retrieveMess...
User ID	pi-eadmin

Figure 6-22 New physical data model connections

8. On the “Select Schema” window (Figure 6-23), click **Select All** to import all schemas, and click **Next**.


New Physical Data Model

Select Schema

Select schema(s) to reverse engineer.

Apply name filter (? = Any character, * = Any string):

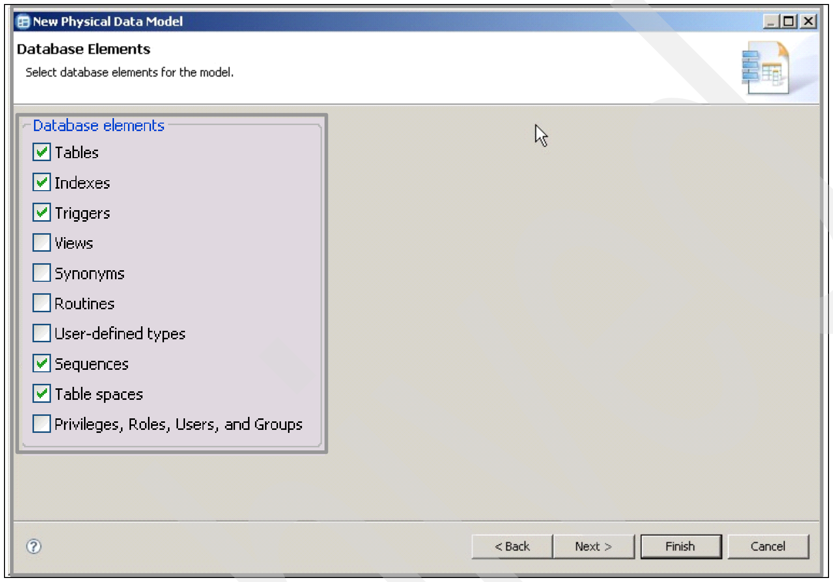
Select objects:

- [ ] SYSIBMTS
- [ ] SYSPUBLIC
- [ ] NULLID
- [ ] SQLJ
- [ ] UTL_FILE
- [✓] GOSALESRT
- [✓] GOSALES
- [ ] DB2ADMIN
- [✓] GOSALESCT
- [ ] SYSTOOLS
- [✓] GOSALESHR
- [✓] GOSALESDW
- [✓] GOSALESMR
- [ ] PI-EADMIN
- [ ] DB2INFO
- [ ] IWSCHEMAAC8CEA_1
- [ ] IWSCHEMA

Figure 6-23 Select schemas

9. On the “Database Elements” window (Figure 6-24), accept the default database elements, and click **Next**.

> **Exception:** If your models are based on views, check the Views box.


New Physical Data Model

Database Elements

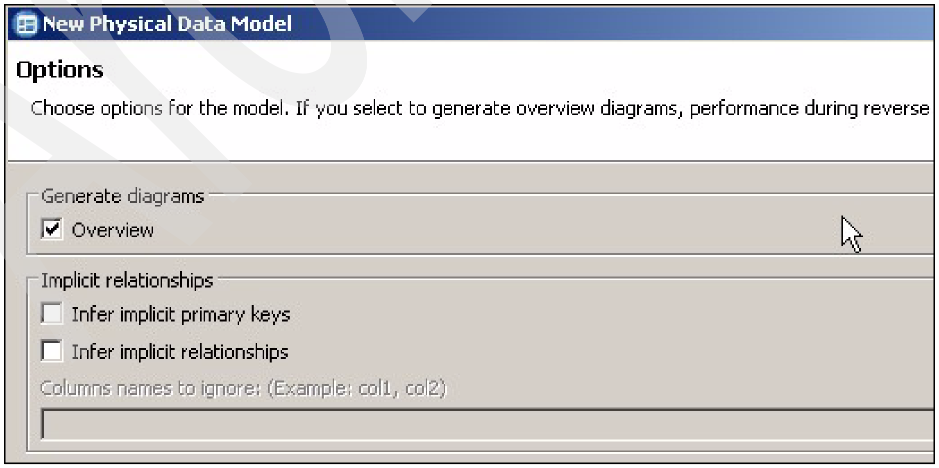
Select database elements for the model.

Database elements

- [✓] Tables
- [✓] Indexes
- [✓] Triggers
- [ ] Views
- [ ] Synonyms
- [ ] Routines
- [ ] User-defined types
- [✓] Sequences
- [✓] Table spaces
- [ ] Privileges, Roles, Users, and Groups

Figure 6-24 Database elements options

10. On the “Options” window (Figure 6-25), select **Overview** in the Generate Diagrams field, and click **Finish**.


New Physical Data Model

Options

Choose options for the model. If you select to generate overview diagrams, performance during reverse

Generate diagrams

- [✓] Overview

Implicit relationships

- [ ] Infer implicit primary keys
- [ ] Infer implicit relationships

Columns names to ignore: (Example: col1, col2)

Figure 6-25 Generate overview diagram

6.6.6 Cube Model wizard

Cubing Services provides a wizard to define OLAP objects in a physical data model. This can be opened by using the context menu, as shown in Figure 6-26.

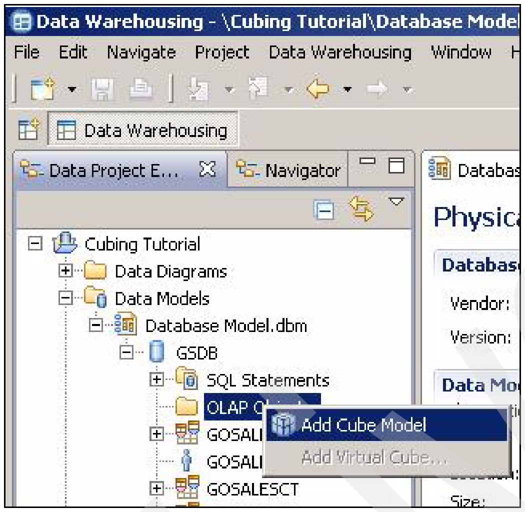


Figure 6-26 Cube model wizard

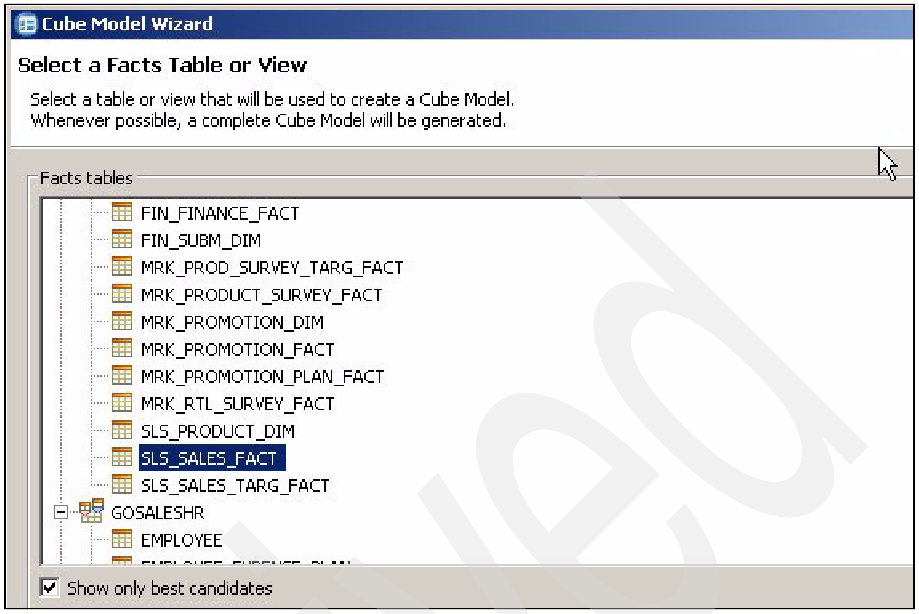
The Cube Model wizard creates the OLAP objects that it can logically infer from the relational star schema or snowflake schema. Based on the fact table, the wizard detects the corresponding dimensions, joins, attributes, and measures.

Before using the Cube Model wizard, make sure that the star schema database is well formed with primary keys and foreign key pairs and referential integrity in place, either enforced or informational. The wizard completes the cube model by creating the dimensions, attributes, and joins using the RI constraints.

6.6.7 Defining the fact table

When you have started the Cube Model wizard (as in Figure 6-26), the first step is to identify the fact table.

1. To start the Cube Model Wizard, expand the Data Model folder until you reach the OLAP Objects folder. Right-click the **OLAP Objects** folder and select **Add Cube Model** (see Figure 6-26**)**.
2. The "Select a Facts Table or View" window (Figure 6-27 on page 205) opens. Select the **Show only best candidates** check box, expand the GOSALESDW schema and select the **SLS_SALES_FACT** table. Click **Next**.



Cube Model Wizard

Select a Facts Table or View

Select a table or view that will be used to create a Cube Model.
Whenever possible, a complete Cube Model will be generated.

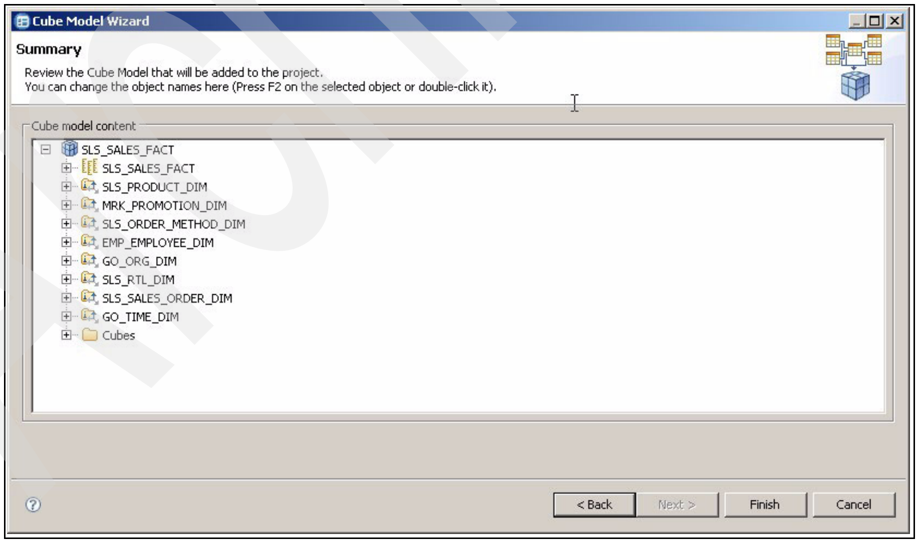
Facts tables

- FIN_FINANCE_FACT
- FIN_SUBM_DIM
- MRK_PROD_SURVEY_TARG_FACT
- MRK_PRODUCT_SURVEY_FACT
- MRK_PROMOTION_DIM
- MRK_PROMOTION_FACT
- MRK_PROMOTION_PLAN_FACT
- MRK_RTL_SURVEY_FACT
- SLS_PRODUCT_DIM
- SLS_SALES_FACT
- SLS_SALES_TARG_FACT
- GOSALESHR
  - EMPLOYEE

☑ Show only best candidates

Figure 6-27 Cube model wizard: Fact table selection

3. The "Summary" window (Figure 6-28) lists the dimensions and hierarchies that have been created for the cube model. You can rename an object by double-clicking it to give them more descriptive names.



Cube Model Wizard

Summary

Review the Cube Model that will be added to the project.
You can change the object names here (Press F2 on the selected object or double-click it).

Cube model content

- SLS_SALES_FACT
  - SLS_SALES_FACT
  - SLS_PRODUCT_DIM
  - MRK_PROMOTION_DIM
  - SLS_ORDER_METHOD_DIM
  - EMP_EMPLOYEE_DIM
  - GO_ORG_DIM
  - SLS_RTL_DIM
  - SLS_SALES_ORDER_DIM
  - GO_TIME_DIM
  - Cubes

Figure 6-28 Cube model summary

4. Click **Finish**.

You can now expand the OLAP Objects folder in the Data Project Explorer View to review the new objects that have been created, as shown in Figure 6-29. You can rename any objects here by highlighting them and changing their name in the Properties View.

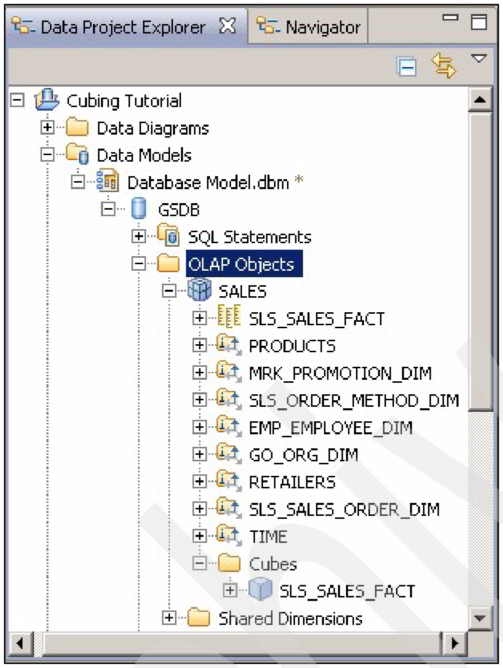


Figure 6-29 Data Project Explorer: OLAP Objects

6.6.8 Creating a new cube

The default cube created by the wizard contains all of the metadata from all of the schemas that make up the cube model. You might want to create a cube with only a subset of this metadata to optimize the response time for your queries.

To create a new cube, perform the following steps:

1. Right-click the **Cubes** folder of the new cube model, and select **Add Cube,** as shown in Figure 6-30.

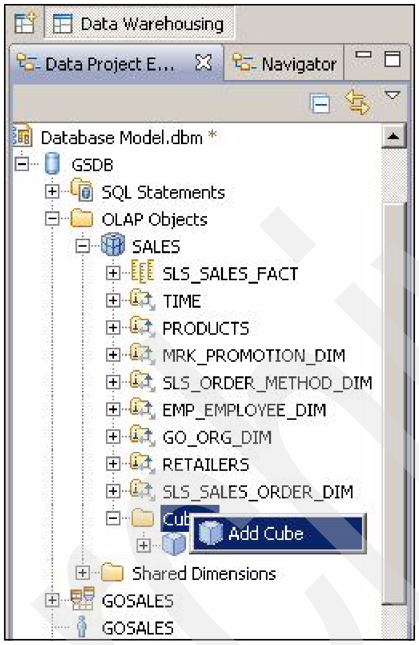


Figure 6-30 Add Cube

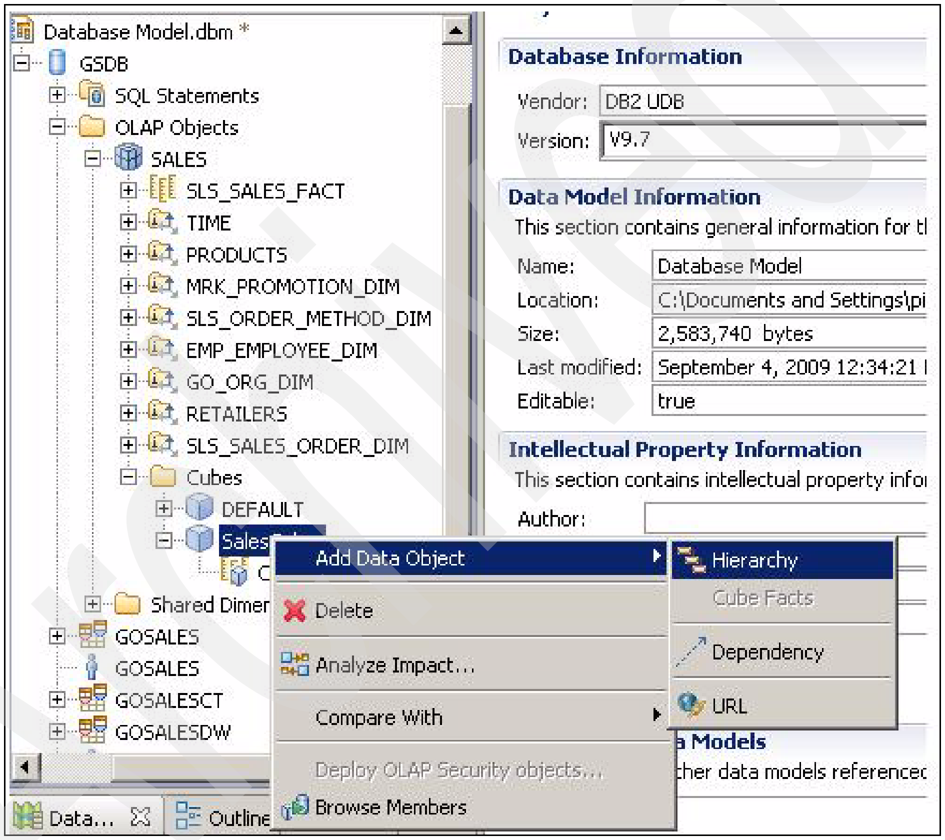
2. Highlight the cube and change its name in the Properties View to `SalesCube`.

6.6.9 Adding hierarchies and measures to a cube

SalesCube currently contains no measures or hierarchies, and the cube is empty. Hierarchies and measures must be added to the cube so that it can be made available for users to query.

To add hierarchies to a cube, perform the following steps:

1. Right-click the **SalesCube** and select **Add Data Object** → **Hierarchy** as shown in Figure 6-31.



Database Information

Vendor: DB2 UDB

Version: V9.7

Data Model Information

This section contains general information for tl

Name: Database Model

Location: C:\Documents and Settings\pi

Size: 2,583,740 bytes

Last modified: September 4, 2009 12:34:21 I

Editable: true

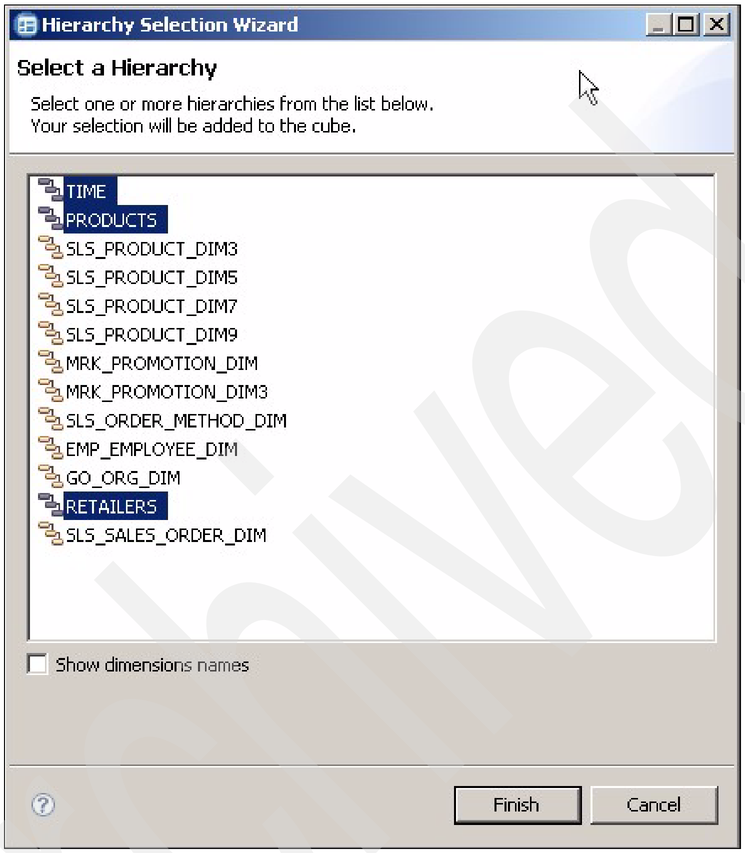
Intellectual Property Information

This section contains intellectual property infoı

Author:

Figure 6-31 Add hierarchy

2. Select **Time**, **Products**, and **Retailer hierarchies** from the list of available hierarchies, as shown in Figure 6-32.



Select a Hierarchy

Select one or more hierarchies from the list below.
Your selection will be added to the cube.

TIME
PRODUCTS
SLS_PRODUCT_DIM3
SLS_PRODUCT_DIM5
SLS_PRODUCT_DIM7
SLS_PRODUCT_DIM9
MRK_PROMOTION_DIM
MRK_PROMOTION_DIM3
SLS_ORDER_METHOD_DIM
EMP_EMPLOYEE_DIM
GO_ORG_DIM
RETAILERS
SLS_SALES_ORDER_DIM

Figure 6-32 Hierarchy Selection Wizard

In the Data Project Explorer you can now see the hierarchies that have been added to the SalesCube, as shown in Figure 6-33.

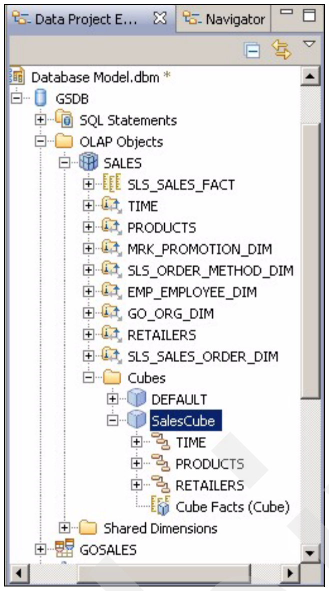
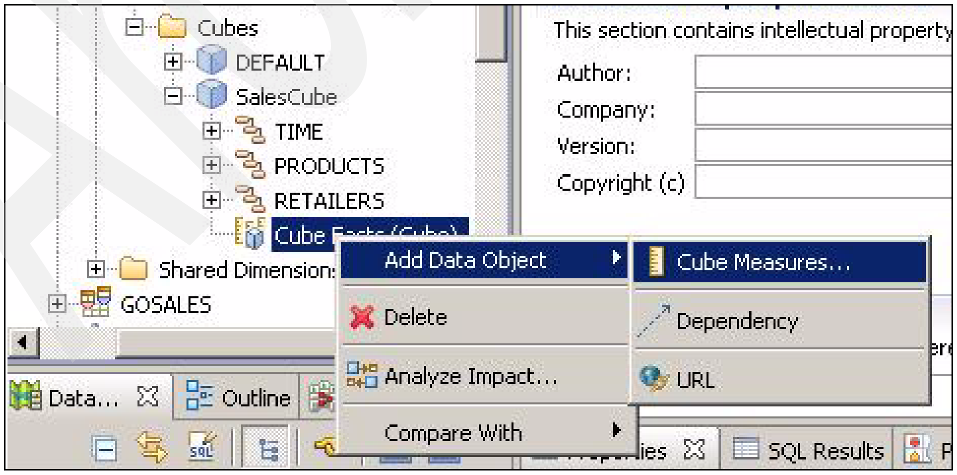


Figure 6-33 Added hierarchies

To add measures to a cube, perform the following steps:

1. Expand the SalesCube to locate the Cube Facts object. Right-click the **Cube Facts (Cube)** object as shown in Figure 6-34 and select **Add Data Object → Cube Measures**.

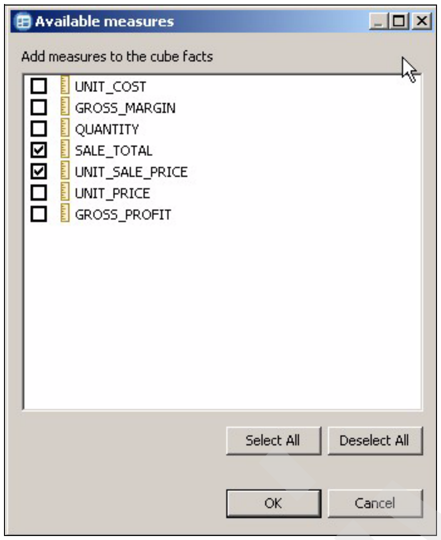


This section contains intellectual property

Author:	
Company:	
Version:	
Copyright (c)	

Figure 6-34 Adding measures

2. Select the **SALE_TOTAL** and **UNIT_SALE_PRICE** check boxes, as shown in Figure 6-35. Click **OK**.



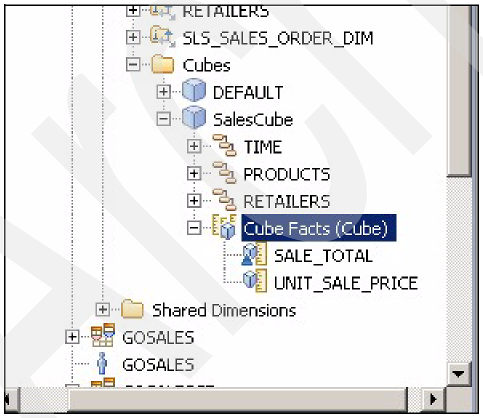
Available measures

Add measures to the cube facts

- [ ] UNIT_COST
- [ ] GROSS_MARGIN
- [ ] QUANTITY
- [x] SALE_TOTAL
- [x] UNIT_SALE_PRICE
- [ ] UNIT_PRICE
- [ ] GROSS_PROFIT

Figure 6-35 Available measures

The measures display under the Cube Facts (Cube) folder, as shown in Figure 6-36.

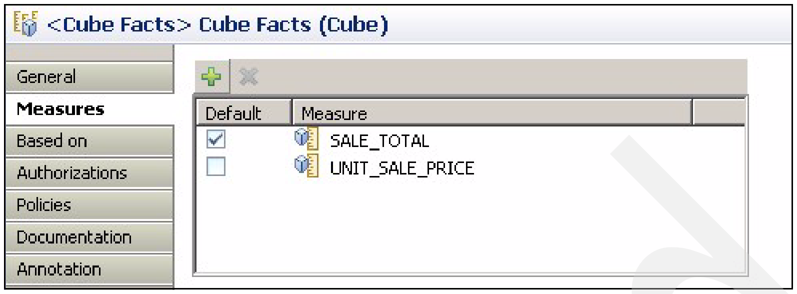


```
RETAILERS
SLS_SALES_ORDER_DIM
Cubes
  DEFAULT
  SalesCube
    TIME
    PRODUCTS
    RETAILERS
    Cube Facts (Cube)
      SALE_TOTAL
      UNIT_SALE_PRICE
Shared Dimensions
GOSALES
GOSALES
```

Figure 6-36 Added measures

3. Highlight the **Cube Facts (Cube)** object in the Data Project Explorer View. In the Properties View, select the **Measures** tab.

4. Make sure that the **SALE_TOTAL** check box is selected, as in Figure 6-37.



Default	Measure
✓	SALE_TOTAL
☐	UNIT_SALE_PRICE

Figure 6-37 Default measure

Setting SALE_TOTAL as the default measure means that SALE_TOTAL displays by default when you run a query against the cube, unless you explicitly specify another measure.

6.6.10 Modifying a hierarchy

A hierarchy is an entity that is defined in the cube model. Hierarchies are available to all cubes in the cube model. However, you cannot modify the hierarchies directly in the cube. You must modify them at the model level, and add them to your cubes.

To create new levels for a hierarchy, you need to add attributes that can be used to define the level. Attributes map to a column in the physical tables. To add attributes, perform the following steps:

1. Locate the Attributes folder in the Retailers dimension (Figure 6-38) . Right-click the **Attributes** folder and select **Add attributes from columns**.

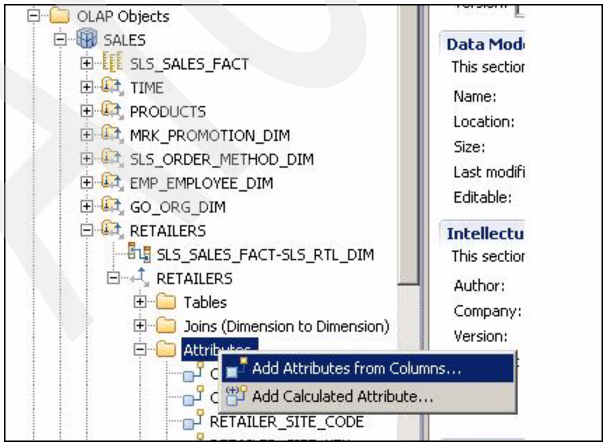


Figure 6-38 Add attributes from columns

2. Add the RETAILER_KEY and RETAILER_NAME columns from the SLS_RTL_DIM table in the GOSALESDW schema, as shown in Figure 6-39.

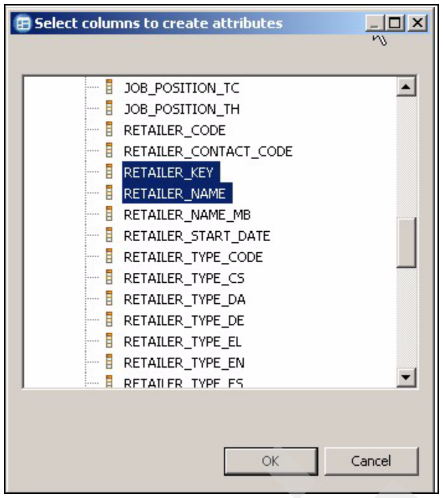


Figure 6-39 Available attributes

3. The attributes are created under the attributes folder. These attributes can now be used to create levels.

To add a level to a hierarchy, perform the following steps:

1. Right-click the **Levels** folder under the Retailers hierarchy and select **Add Level**, as shown in Figure 6-40.

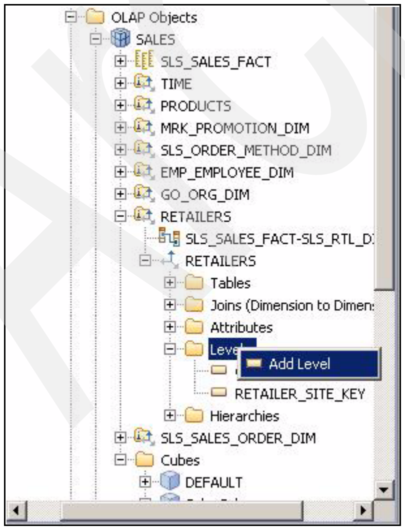
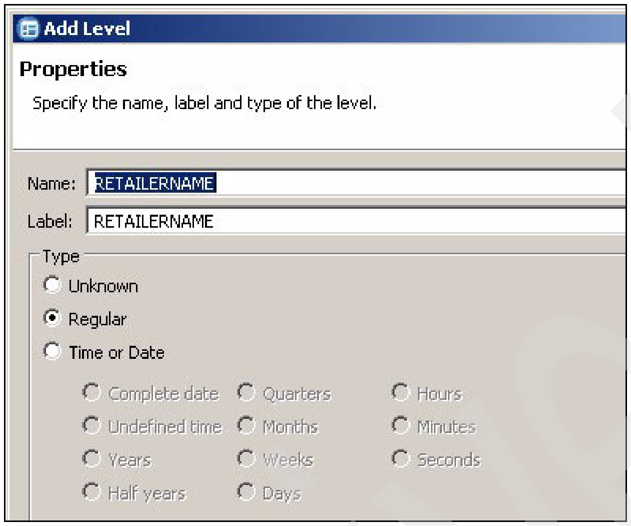


Figure 6-40 Add level

2. In the “Properties” window (Figure 6-41), specify the name as `Retailer Name`, and select the **Regular** radio button.


Add Level

Properties

Specify the name, label and type of the level.

Name: RETAILERNAME

Label: RETAILERNAME

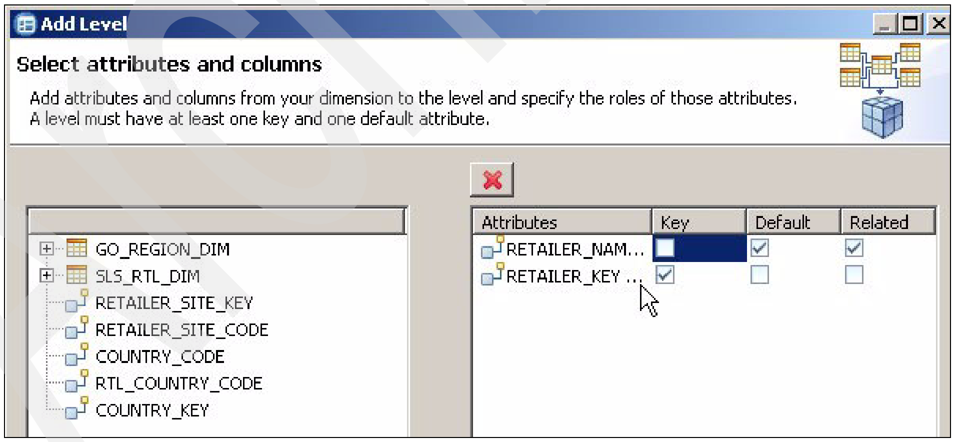
Type

○ Unknown
● Regular
○ Time or Date

○ Complete date	○ Quarters	○ Hours
○ Undefined time	○ Months	○ Minutes
○ Years	○ Weeks	○ Seconds
○ Half years	○ Days	

Figure 6-41 Level Properties

3. Use the arrow icon to select the **RETAILER_NAME** and **RETAILER_KEY** attributes. Make RETAILER_KEY the key attribute, and RETAILER_NAME the default and related attribute, as shown in Figure 6-42.


Add Level

Select attributes and columns

Add attributes and columns from your dimension to the level and specify the roles of those attributes.
A level must have at least one key and one default attribute.

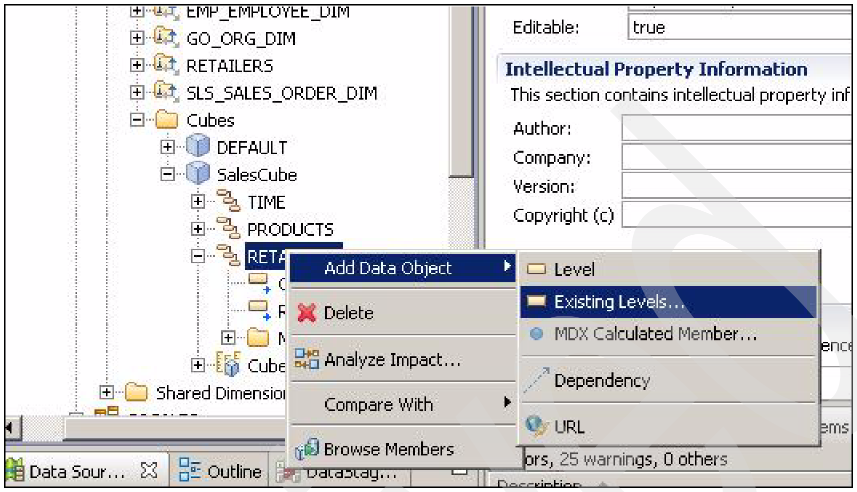
- GO_REGION_DIM
- SLS_RTL_DIM
  - RETAILER_SITE_KEY
  - RETAILER_SITE_CODE
  - COUNTRY_CODE
  - RTL_COUNTRY_CODE
  - COUNTRY_KEY

Attributes	Key	Default	Related
RETAILER_NAM...	☐	✓	✓
RETAILER_KEY ...	✓	☐	☐

Figure 6-42 Level attributes

4. Click **Next** and click **Finish**. The levels that have been created can now be added to a hierarchy.

5. Right-click the **Retailers** hierarchy and select **Add Data Object** → **Existing Levels** as shown in Figure 6-43.



Editable: true

Intellectual Property Information

This section contains intellectual property inf

Author:

Company:

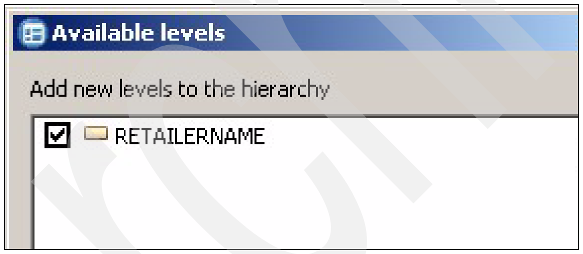
Version:

Copyright (c)

ors, 25 warnings, 0 others

Figure 6-43 Add level to hierarchy

6. Select **RETAILER_NAME** from the list of available levels, as shown in Figure 6-44, and click **OK**.



Available levels

Add new levels to the hierarchy

☑ RETAILERNAME

Figure 6-44 Available levels

The levels in the hierarchy next must be moved into the correct order.

7. Highlight the Retailers hierarchy, and select the **Levels** tab in the Properties View, as shown in Figure 6-45. Highlight RETAILER NAME and use the Up arrow to move it to the top of the list.

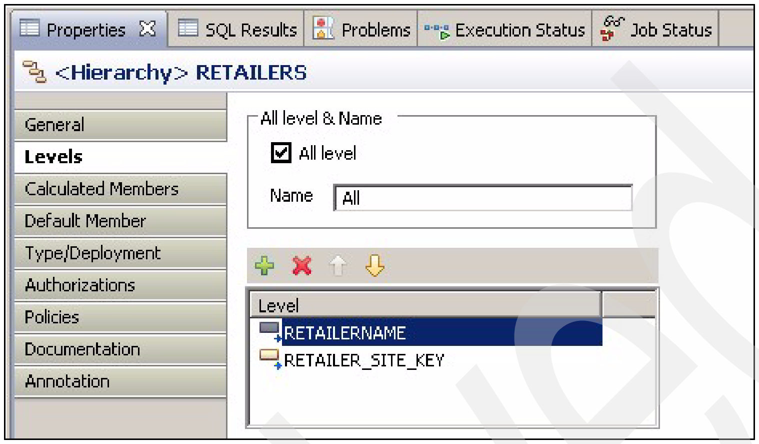


Figure 6-45 Rearrange hierarchy levels

6.6.11 Adding tables to a cube model

Levels can be created based on dimension-to-dimension table joins. First, the table needs to be added to the cube model, and the join can be created.

To add an additional table to a cube model, perform the following steps:

1. Locate the Tables folder in the Retailers dimension. Right-click the **Tables** folder and select **Add Existing Table**, as shown in Figure 6-46.

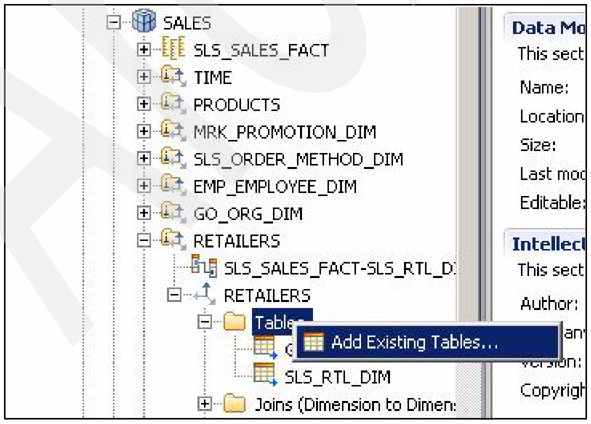
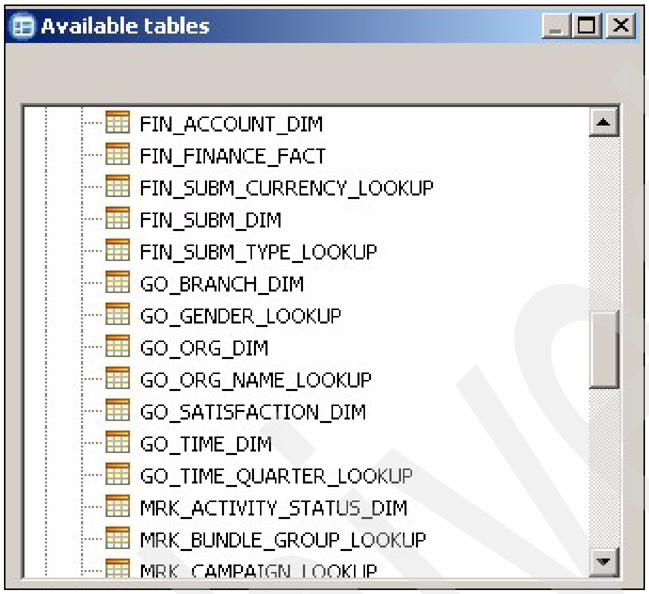


Figure 6-46 Add table to model

2. Expand the GOSALESDW schema and select the **GO_REGION_DIM** table from the list of options, as shown in Figure 6-47. Click **OK**. The table has now been added to the cube model.



FIN_ACCOUNT_DIM
FIN_FINANCE_FACT
FIN_SUBM_CURRENCY_LOOKUP
FIN_SUBM_DIM
FIN_SUBM_TYPE_LOOKUP
GO_BRANCH_DIM
GO_GENDER_LOOKUP
GO_ORG_DIM
GO_ORG_NAME_LOOKUP
GO_SATISFACTION_DIM
GO_TIME_DIM
GO_TIME_QUARTER_LOOKUP
MRK_ACTIVITY_STATUS_DIM
MRK_BUNDLE_GROUP_LOOKUP
MRK_CAMPAIGN_LOOKUP

Figure 6-47 Available tables

6.6.12 Creating a dimension-to-dimension join

To create a dimension-to-dimension join, perform the following steps:

1. Locate the Joins folder in the Retailers dimension. Right-click the **Joins** folder and select **Add Join** (Figure 6-48). The new join is created under the Joins folder as the Join object.

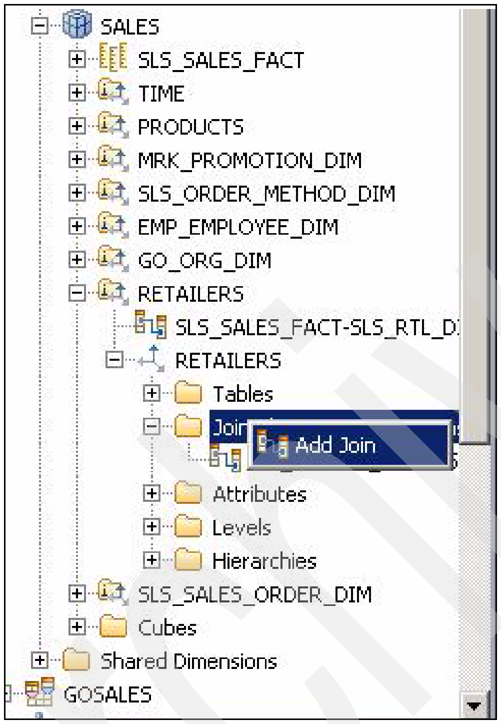
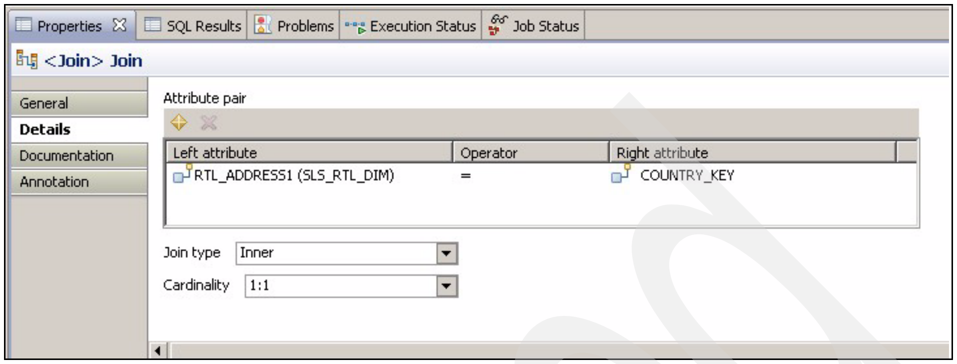


Figure 6-48 Add Join

2. Select the **Join object** to display in the Properties View.
3. On the general tab, change the name of the join to `SLS_RTL_DIM-GO_REGION_DIM`.
4. Select the **Details** tab of the Properties View. Click the **Add Attribute Pair** icon to open the wizard.
5. In the left column pane, expand SLS_RTL_DIM and select the **RTL_COUNTY_CODE** column.
6. In the right column pane, expand GO_REGION_DIM and select the **COUNTRY_CODE** column.

7. The attribute pair displays on the details tab, as shown in Figure 6-49. Select **Many:1** from the Cardinality menu.



Left attribute	Operator	Right attribute
RTL_ADDRESS1 (SLS_RTL_DIM)	=	COUNTRY_KEY

Join type: Inner

Cardinality: 1:1

Figure 6-49 Join Properties

6.6.13 Analyzing OLAP objects for validity

After creating a physical data and OLAP model, the models can be validated using the Analyze Model option available from the context menus displayed from the Data Project Explorer, as shown in Figure 6-50 on page 220.

A cube model validates successfully only after you add the following mandatory components:

- At least one facts object
- At least one dimension
- A hierarchy defined for at least one dimension
- Joins between the existing facts objects and dimensions
- Attributes that reference existing table columns

To validate a cube model in the Data Project Explorer:

1. Right-click the **SalesCube model** and select **Analyze Model** (Figure 6-50).

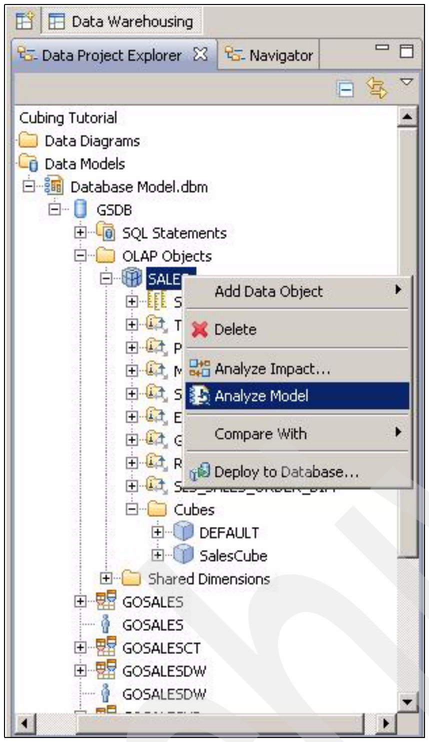


Figure 6-50 Analyze Model

2. Select the **Physical Data Model** check box. All of the rules beneath the Physical Data Model object are also selected (Figure 6-51).

   Any errors should be resolved before you continue working on your model or cube. Warnings might be resolved when the Cubing Services Optimization Advisor Wizard is run to resolve any issues with indexes and constraints.

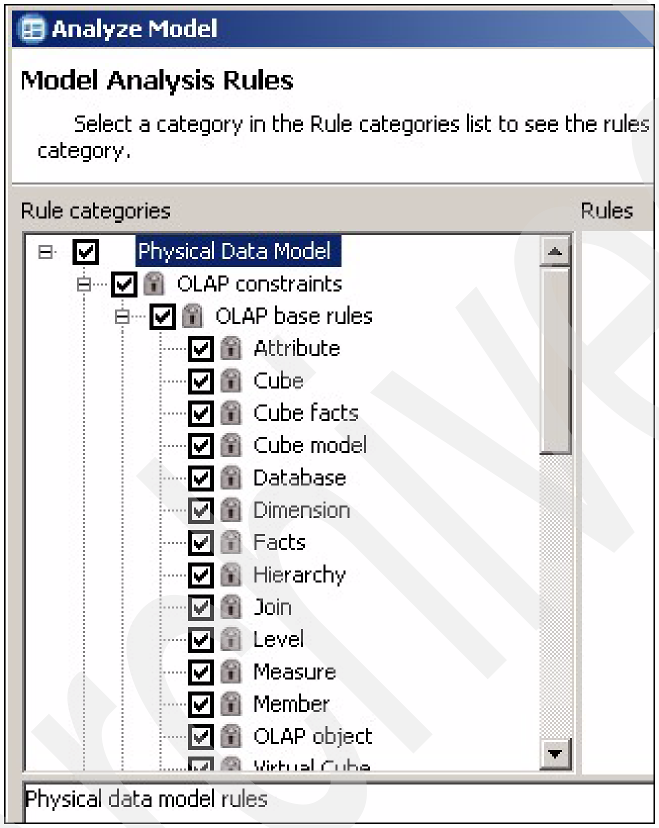


Figure 6-51 Model Analysis Rules window

6.6.14 Exporting a cube model from Design Studio

Objects that are created in Design Studio exist only in the local system until they are deployed. The cube server uses cube models that are stored in a metadata repository in a DB2 database. You can export your cube model from Design Studio to an XML file, which can then be imported into the metadata repository by using the administration console.

To export a cube model to an XML file, perform the following steps:

1. Click **File** → **Export**.
2. Expand **Data Warehousing**, and select **OLAP metadata**. Click **Next**.
3. Specify the file name. Browse to C:\temp and save the file as `tutorialmodel.xml`.
4. Expand the GSDB container and select the **SalesCube** model. Click **Finish**.

The OLAP objects have now been exported, and can be deployed using the administration console. For information about cube deployment, see Chapter 9, “Deploying and managing solutions” on page 353. After a cube has been deployed, it can be access by reporting tools.

6.6.15 Running the Cubing Services Optimization Advisor

The Optimization Advisor can be launched from the context menu of a cube model in the Data Source Explorer, as shown in Figure 6-52.

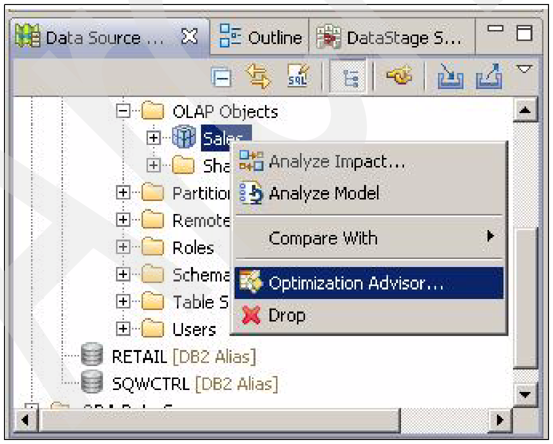
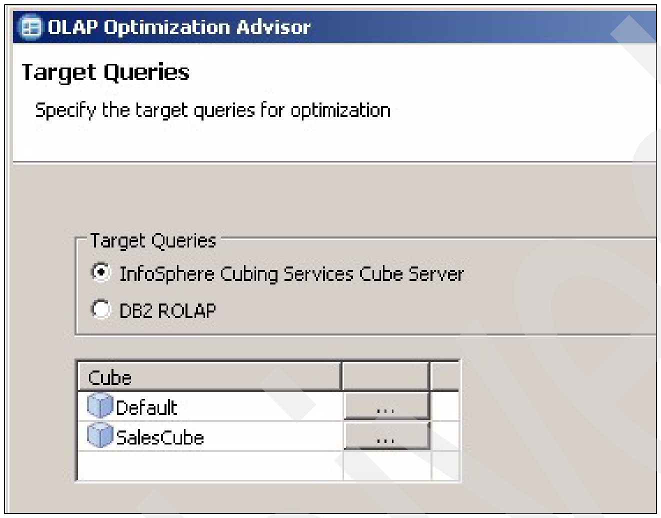


Figure 6-52 Launching the Optimization Advisor

1. Right-click the **Sales** cube model in the Data Source Explorer and select **Optimization Advisor**.
2. Ensure that the **InfoSphere Cubing Services Cube Server** radio button is selected as the target query, as shown in Figure 6-53. Click **Next**.



OLAP Optimization Advisor

Target Queries

Specify the target queries for optimization

Target Queries

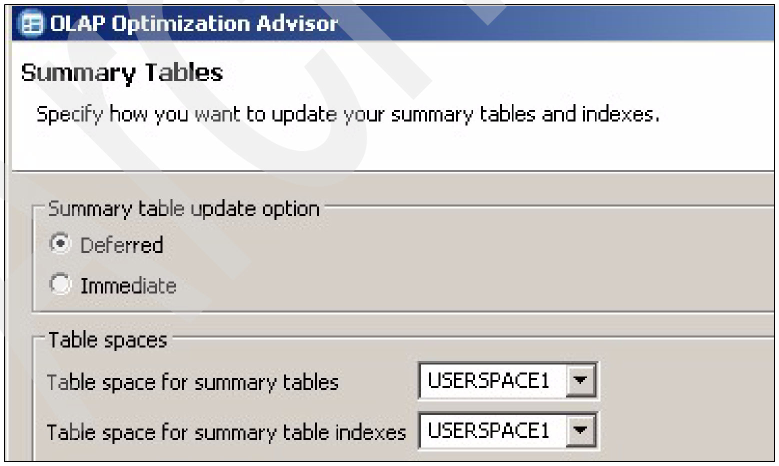
(●) InfoSphere Cubing Services Cube Server

( ) DB2 ROLAP

Cube	
Default	...
SalesCube	...

Figure 6-53 Target Queries

3. Ensure that the **Deferred** radio button in the summary table is selected, as shown in Figure 6-54.



OLAP Optimization Advisor

Summary Tables

Specify how you want to update your summary tables and indexes.

Summary table update option

(●) Deferred

( ) Immediate

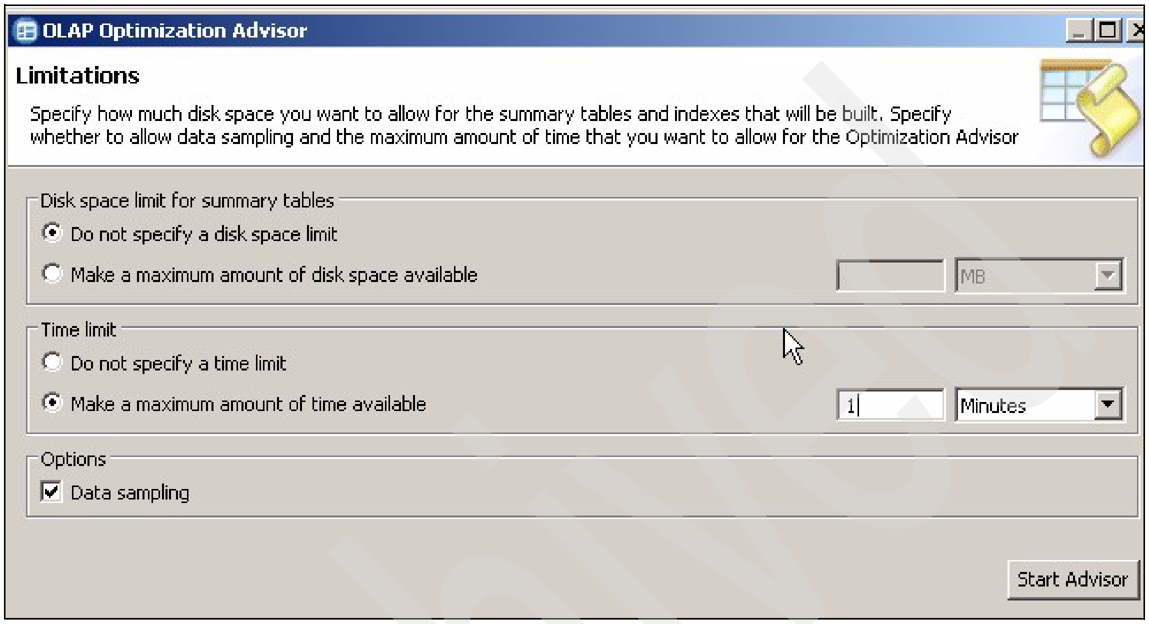
Table spaces

Table space for summary tables: USERSPACE1

Table space for summary table indexes: USERSPACE1

Figure 6-54 MQT refresh

4. Specify a maximum time limit of 1 minute, as shown in Figure 6-55. In a real production scenario, you will experience better cube performance if you choose to use more disk space, time, and data sampling.


OLAP Optimization Advisor

Limitations

Specify how much disk space you want to allow for the summary tables and indexes that will be built. Specify whether to allow data sampling and the maximum amount of time that you want to allow for the Optimization Advisor

Disk space limit for summary tables
- [● Do not specify a disk space limit]
- Make a maximum amount of disk space available — MB

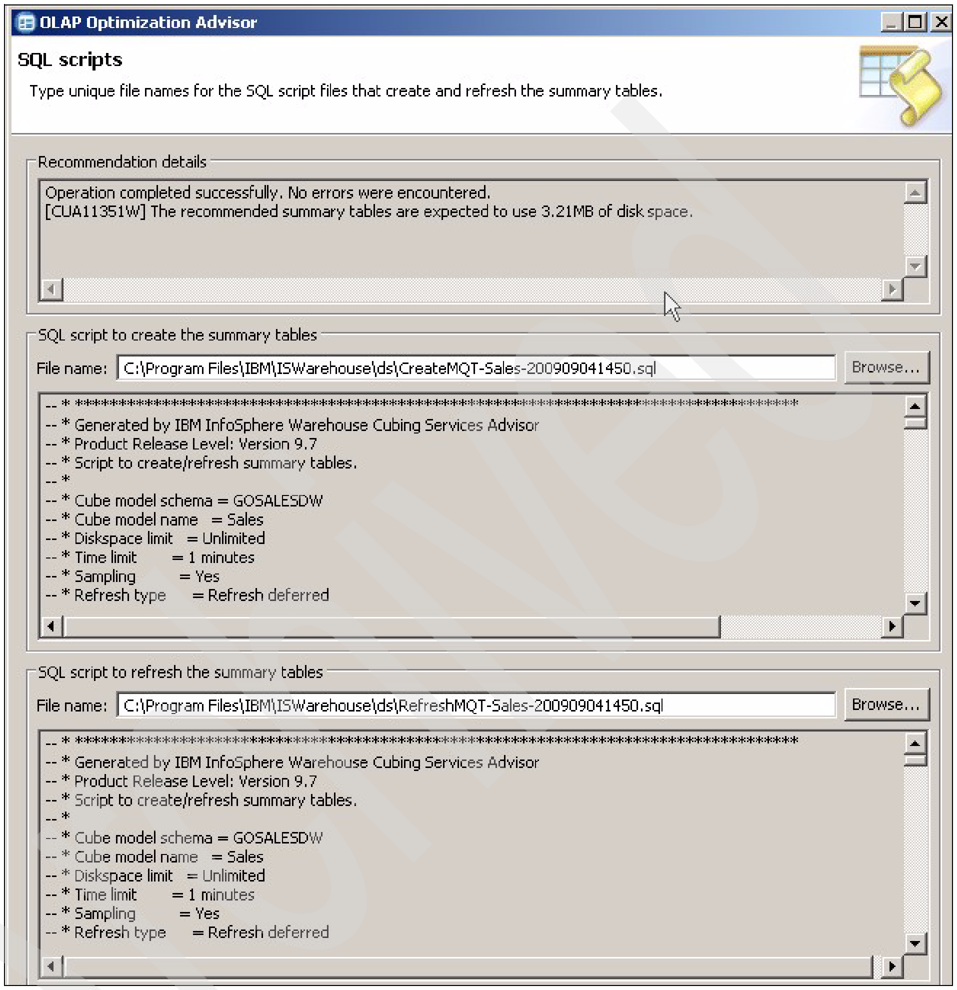
Time limit
- Do not specify a time limit
- [● Make a maximum amount of time available] 1 Minutes

Options
- [✓ Data sampling]

Figure 6-55 MQT limitations

5. Click **Start Advisor**.
6. After the Advisor finishes, click **Next**.

7. Specify the file path and name of the SQL script that creates the recommended MQTs shown in Figure 6-56. Click **Finish** to save the scripts.



SQL scripts

Type unique file names for the SQL script files that create and refresh the summary tables.

Recommendation details

```
Operation completed successfully. No errors were encountered.
[CUA11351W] The recommended summary tables are expected to use 3.21MB of disk space.
```

SQL script to create the summary tables

File name: C:\Program Files\IBM\ISWarehouse\ds\CreateMQT-Sales-200909041450.sql

```
-- * *****************************************************************************
-- * Generated by IBM InfoSphere Warehouse Cubing Services Advisor
-- * Product Release Level: Version 9.7
-- * Script to create/refresh summary tables.
-- *
-- * Cube model schema = GOSALESDW
-- * Cube model name   = Sales
-- * Diskspace limit   = Unlimited
-- * Time limit        = 1 minutes
-- * Sampling          = Yes
-- * Refresh type      = Refresh deferred
```

SQL script to refresh the summary tables

File name: C:\Program Files\IBM\ISWarehouse\ds\RefreshMQT-Sales-200909041450.sql

```
-- * *****************************************************************************
-- * Generated by IBM InfoSphere Warehouse Cubing Services Advisor
-- * Product Release Level: Version 9.7
-- * Script to create/refresh summary tables.
-- *
-- * Cube model schema = GOSALESDW
-- * Cube model name   = Sales
-- * Diskspace limit   = Unlimited
-- * Time limit        = 1 minutes
-- * Sampling          = Yes
-- * Refresh type      = Refresh deferred
```

Figure 6-56 MQT scripts

The MQTs must be deployed to the database before they can be used against the cube.

6.7 Cubing Services features

In this section we detail the features that are specific to InfoSphere Warehouse 9.7 for LUW, including virtual cubes and dimensional security.

6.7.1 Virtual cubes

Virtual cubes are one of the new Cubing Services features in IBM InfoSphere Warehouse 9.7. A virtual cube provides a way to merge different cubes together to allow a single query destination that returns merged results from the composing cubes.

A virtual cube is a logical cube that is defined in terms of exactly two existing cubes: either two real cubes, two other virtual cubes, or one virtual cube and one real cube. The resulting virtual cubes can be merged with other cubes, as shown in Figure 6-57.

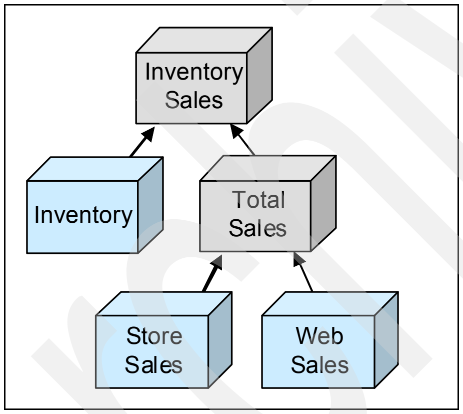


Figure 6-57 Cubes aggregated to form virtual cubes

The merging is done based on the dimension names. Dimensions with the same names in both cubes are merged. Dimensions from one cube that do not have a corresponding dimension with the same name in the other cube are added to the virtual cube. The two cubes that are merged can belong to different cube models, so they might have completely different internal structures. The main requirement is that they share at least one dimension. The virtual cubes and the depending cubes must reside in the same database. A virtual cube can be defined using only cubes from the same cube server.

When a query is issued against a virtual cube, it is routed to the dependent cubes, which produces two intermediary results that are aggregated according to the virtual cube's merge operator. The following list details the merge operators:

- SUM
- MINUS
- PRODUCT
- DIVIDE
- MAX
- MIN
- NOP (returns data from the first cube in the Virtual Cube definition)

Working with virtual cubes

Virtual cubes are defined using Design Studio, and the administrative tasks for the virtual cubes are performed using the administration console. To create a virtual cube, you must have at least two cubes in the physical data model.

To create a virtual cube, perform the following steps:

1. In the Data Project Explorer, right-click the **OLAP Objects** folder, and select **Add Virtual Cube** (Figure 6-58). The Virtual Cube Wizard opens.

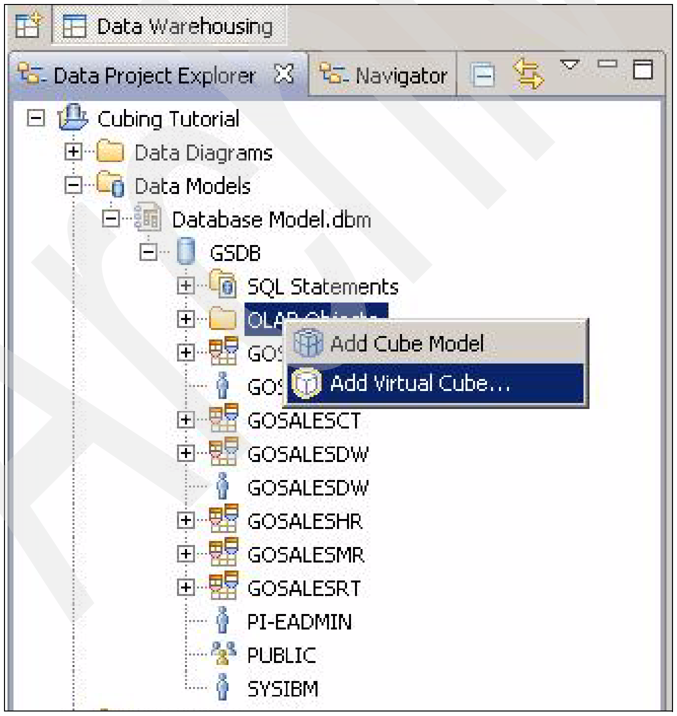
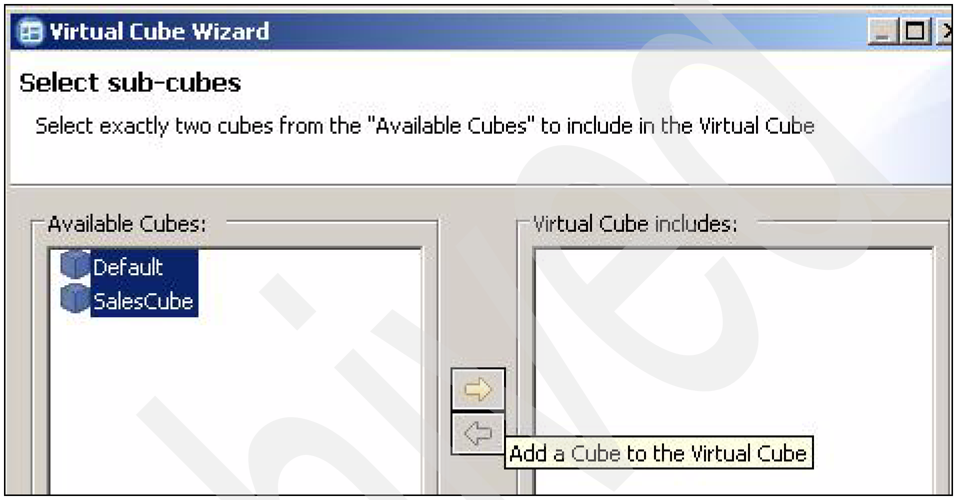


Figure 6-58 Creating a virtual cube

2. Select the two cubes on which that virtual cube is based from the Available Cubes box, and click **Finish**.

> **Note:** The **Finish** button is greyed-out in Figure 6-59 because the window was captured before the selected cubes had not yet been moved to the right pane of the window.


Virtual Cube Wizard

Select sub-cubes

Select exactly two cubes from the "Available Cubes" to include in the Virtual Cube

Figure 6-59 Select sub-cubes

3. Expand the **OLAP Objects** folder in the Data Project Explorer and highlight the virtual cube. In the Properties View, specify the name, change the merge operator, or add virtual measures or MDX calculated measures to the cube.

After a virtual cube is created, you can create virtual dimensions, virtual measures, virtual members, or MDX calculated members for this virtual cube. If you make changes to one of the two cubes on which the virtual cube is based, the virtual cube is updated. You do not need to recreate the virtual cube.

You can deploy the virtual cube to a database, or export it to a metadata file, which can be imported into the metadata repository by using the administration console. When you deploy a virtual cube to a database, the two dependent cubes are automatically deployed.

6.7.2 Dimensional security

In InfoSphere Warehouse v9.5, Cubing Services enabled you to secure cubes using the administration console. You could either grant or deny a role the privilege to query a cube. In v9.7, you can still restrict access to a cube, but you use Design Studio to do so.

A role that is allowed to access a cube is allowed to access all the members of a dimension by default. Starting with v9.7, you can limit the members that a role can access to a subset of the members of a dimension. You can need to limit the members that a role can access to ensure that only legitimate users have access to the information they need.

For example, you can have a sales cube that contains information about retailers in different countries. The sales managers in one country might need access to information about the retailers in their own country but not to information about the retailers outside their country. With dimension security, you can limit the members of the retailers dimension that sales managers can access to only those in the countries that they manage.

To limit the cubes and members of a dimension that a role can access, you need to design and deploy a security model. The security model contains the rules for accessing OLAP objects. The security model does not contain the actual definition of these OLAP objects. The cube model does. Because the security model depends on the cube model, you should design and deploy the cube model before designing and deploying the security model.

The high level steps to design and deploy a security model, depicted in Figure 6-60 are as follows:

1. Create a security model in Design Studio.
2. Export the security model as an XML file.
3. Import the XML file into the metadata repository using the administration console.
4. View the security model in the administration console.
5. Instruct the Cube Server to refresh security and start enforcing the security model.

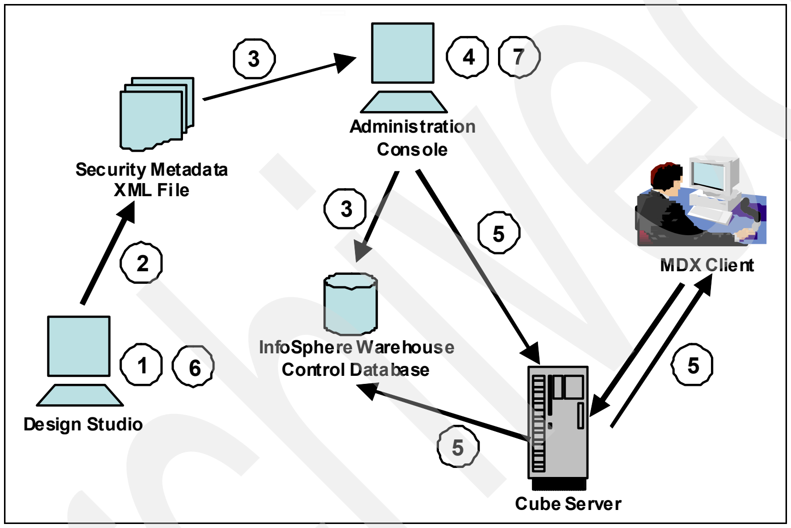
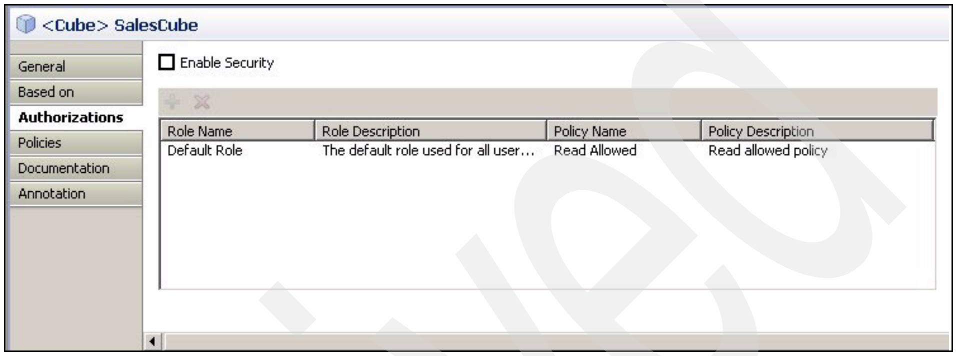


Figure 6-60 Designing and deploying a security model

Creating roles and securing a cube

To create roles and secure the SalesCube, perform the following steps:

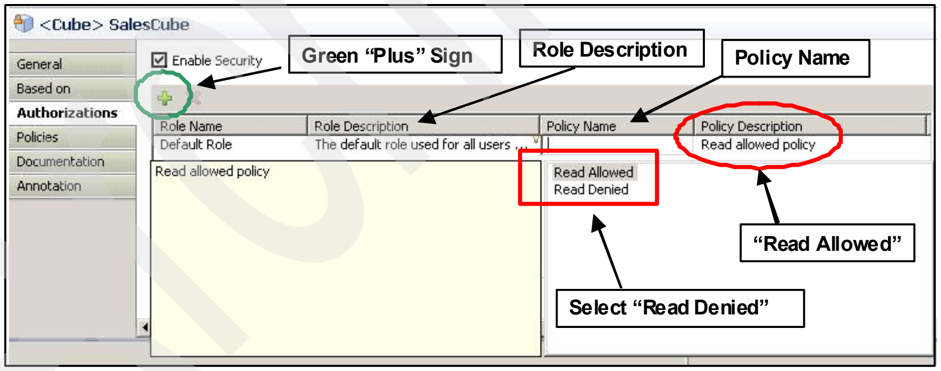
1. Locate the SalesCube cube in the Data Project Explorer. Highlight the cube to display its properties in the Properties View.
2. Select the **Authorizations** tab. Select the **Enable Security** check box. The Default role is automatically created, as in Figure 6-61.



Role Name	Role Description	Policy Name	Policy Description
Default Role	The default role used for all user...	Read Allowed	Read allowed policy

Figure 6-61 SalesCube security properties

3. Click the **Read Allowed** policy name of the **Default Role** as shown in Figure 6-62, and delete the text that is shown.



Role Name	Role Description	Policy Name	Policy Description
Default Role	The default role used for all users ...		Read allowed policy

Figure 6-62 Change permissions

4. Press CTRL + Space to see a list of options. Select **Read Denied**, as highlighted in Figure 6-62. New users now have no access by default.
5. Create a new Managers role. On the Authorizations page, click the green plus sign. Give the new role a name of `Managers`.
6. Click in the Role Description field and type `Manager Only Role`.
7. Make sure that the Policy Name field is set to **Read Allowed**.

The roles have now been created and the cube secured, as shown in Figure 6-63. A lock icon appears on the cube icon to indicate that the cube has security policies enabled. Any users who want to query this cube must now be assigned to one of the roles.

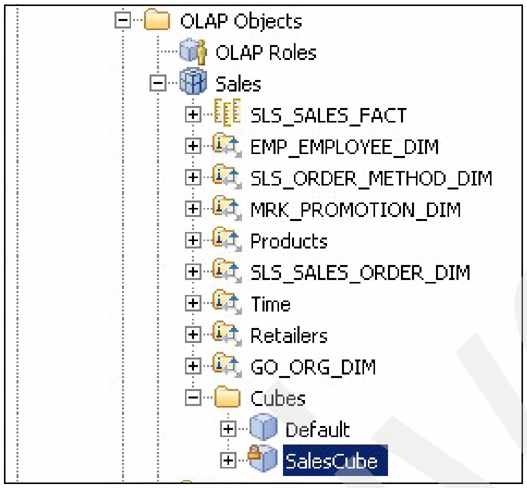


Figure 6-63 Cube locked

Dimension security

The next step is to create roles that are restricted to accessing certain members of a dimension. The following list of actions shows how to restrict access to members in the Retailers hierarchy.

To enable security for a hierarchy, perform the following steps:

1. Expand the **SalesCube** and highlight the **Retailers** hierarchy. Its properties will display in the Properties View.
2. Open the **Authorizations** tab and check the **Enable Security** check box.
3. Click the **green plus sign** to add a new role. Give it a name of `Americas Sales Managers`.
4. Change the role description to **Sales managers for the Americas**.
5. Open the **Policies** tab of the Properties View.
6. Click the green plus sign to add a new policy, and call it `Americas Region`.
7. Restrict the policy to include only members that are in the Americas. Select the text in the **Allowed Members** column and click the ellipsis push button.
8. Specify the following text in the MDX Expression Builder window: `Descendants([Retailers].[Amercias])`.

9. Click **Validate**, and **OK** to create the expression.
10. Set the policy for the Americas Sales Managers role. Open the **Authorizations** tab and press CTRL + Space on the policy name, and select **Americas Region** policy. The Retailers hierarchy has now been secured.

Deploying the security metadata

The security metadata currently resides in the cube that exists locally on your server. The security metadata needs to be exported from Design Studio and imported into the cube model that is administered in the administration console. When the security metadata is imported, the roles that you created in Design Studio are created in the administration console. Users can then be added to these roles. To export the security metadata from Design Studio, perform the following steps:

1. Click **File** → **Export** to open the export wizard.
2. Choose **Data Warehousing**, and **OLAP Security**, as shown in Figure 6-64.

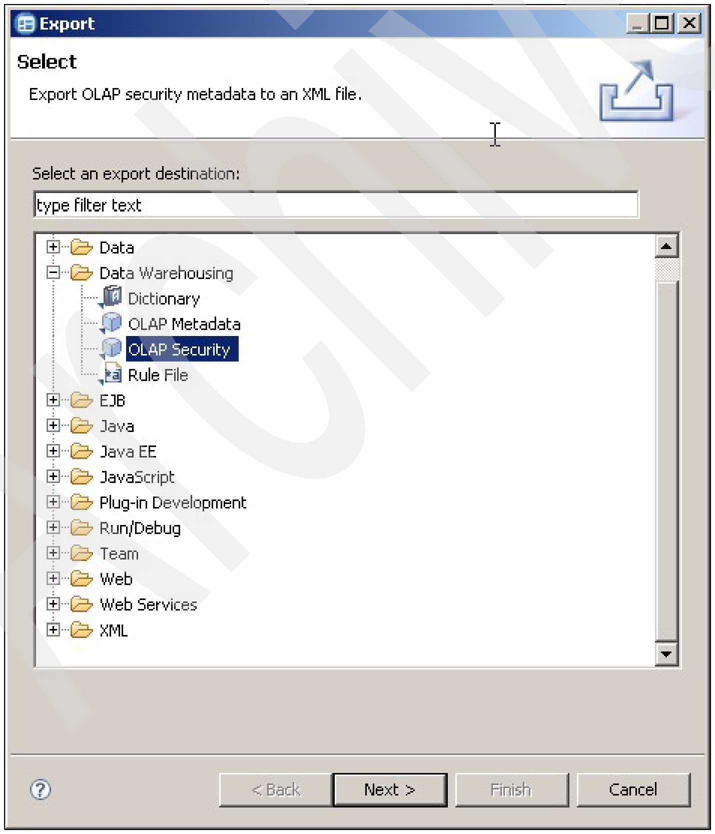
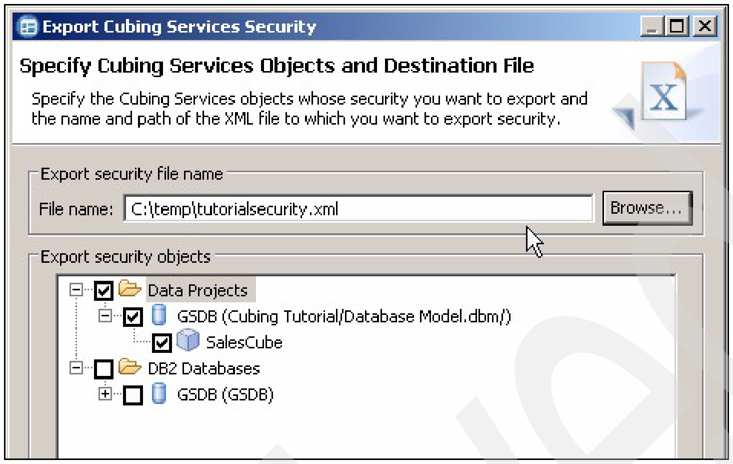


Figure 6-64 OLAP Security

3. Choose a file name and path, and select the **SalesCube** check box, as shown in Figure 6-65.


Export Cubing Services Security

Specify Cubing Services Objects and Destination File

Specify the Cubing Services objects whose security you want to export and the name and path of the XML file to which you want to export security.

Export security file name

File name: C:\temp\tutorialsecurity.xml

Export security objects

- [✓] Data Projects
  - [✓] GSDB (Cubing Tutorial/Database Model.dbm/)
    - [✓] SalesCube
- [ ] DB2 Databases
  - [ ] GSDB (GSDB)

Figure 6-65 Export XML

4. Click **Finish**. The OLAP objects have now been exported.

The next step in the process is to import the metadata to the Cube Server using the administration console.This process is detailed in “Importing the OLAP metadata” on page 404.

6.8 Cubing Services for System z

In this section we describe the Cubing Services features that are specific to InfoSphere Warehouse 9.5.2 for System z. As on LUW platforms, Cubing Services on System z is a multidimensional database server. It takes requests in the form of multidimensional expressions (MDX), obtains the data from a cube and returns the results. The architecture of Cubing Services on System z is similar to Cubing Services on LUW as seen in Figure 6-66 on page 235. On System z, the cube server runs in the Linux partition and the back end database is DB2 for z/OS V8 or V9.

Cubing Services on System Z is primarily the same as Cubing Services on LUW. The cube modeling techniques and practices documented in this book, and in the existing IBM Redbook, *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582, generally apply to Cubing Services on System z. However, there are differences, and we discuss those in the subsequent sections of this chapter.

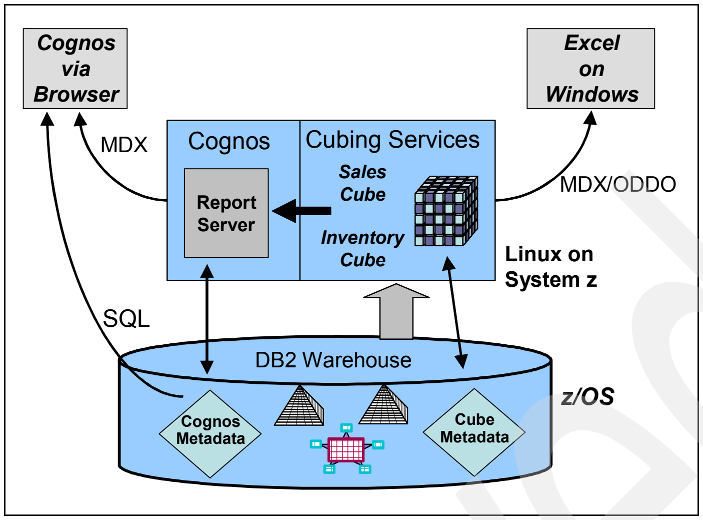
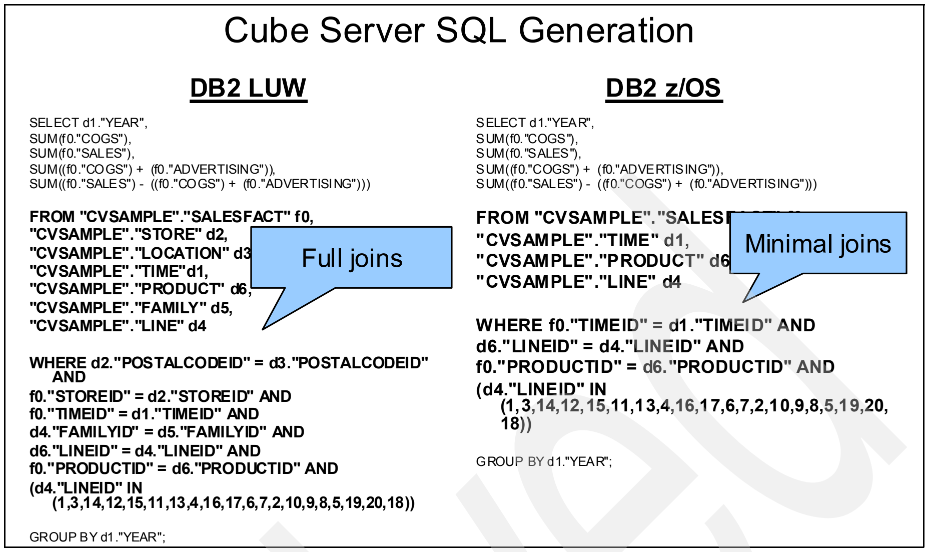


Figure 6-66 Architecture of Cubing Service on System z

6.8.1 Cubing Services differences on System z

There are two primary differences between Cubing Services on LUW and on the System z platforms due to the differences in the DB2 for z/OS optimizer and the MQT support of DB2 for z/OS.

The DB2 for z/OS optimizer does not eliminate lossless joins. Therefore, the Cube Server SQL generator must generate different SQL for DB2 for z/OS, as can be seen in Figure 6-67 on page 236. This is handled by the SQL generation routines of the Cube Server and the Optimization Advisor, and is transparent to the users of Cubing Services.



Cube Server SQL Generation

DB2 LUW

```
SELECT d1."YEAR",
SUM(f0."COGS"),
SUM(f0."SALES"),
SUM((f0."COGS") + (f0."ADVERTISING")),
SUM((f0."SALES") - ((f0."COGS") + (f0."ADVERTISING")))

FROM "CVSAMPLE"."SALESFACT" f0,
"CVSAMPLE"."STORE" d2,
"CVSAMPLE"."LOCATION" d3
"CVSAMPLE"."TIME"d1,
"CVSAMPLE"."PRODUCT" d6,
"CVSAMPLE"."FAMILY" d5,
"CVSAMPLE"."LINE" d4

WHERE d2."POSTALCODEID" = d3."POSTALCODEID"
    AND
f0."STOREID" = d2."STOREID" AND
f0."TIMEID" = d1."TIMEID" AND
d4."FAMILYID" = d5."FAMILYID" AND
d6."LINEID" = d4."LINEID" AND
f0."PRODUCTID" = d6."PRODUCTID" AND
(d4."LINEID" IN
    (1,3,14,12,15,11,13,4,16,17,6,7,2,10,9,8,5,19,20,18))

GROUP BY d1."YEAR";
```

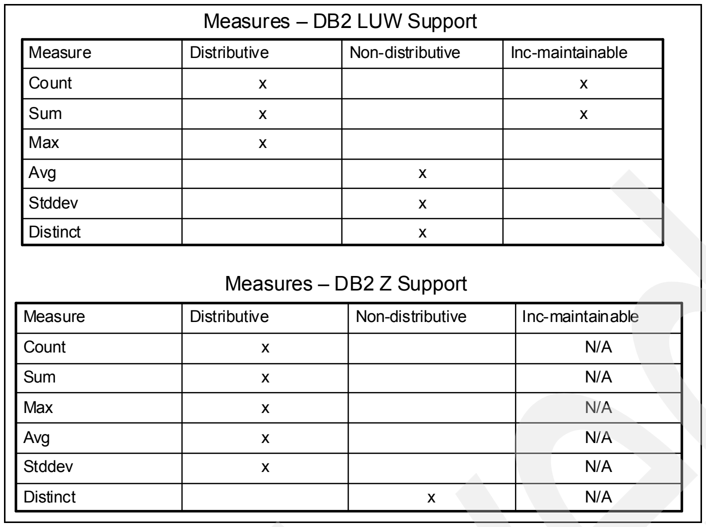
DB2 z/OS

```
SELECT d1."YEAR",
SUM(f0."COGS"),
SUM(f0."SALES"),
SUM((f0."COGS") + (f0."ADVERTISING")),
SUM((f0."SALES") - ((f0."COGS") + (f0."ADVERTISING")))

FROM "CVSAMPLE"."SALESF
"CVSAMPLE"."TIME" d1,
"CVSAMPLE"."PRODUCT" d6
"CVSAMPLE"."LINE" d4

WHERE f0."TIMEID" = d1."TIMEID" AND
d6."LINEID" = d4."LINEID" AND
f0."PRODUCTID" = d6."PRODUCTID" AND
(d4."LINEID" IN
    (1,3,14,12,15,11,13,4,16,17,6,7,2,10,9,8,5,19,20,
    18))

GROUP BY d1."YEAR";
```

Figure 6-67 Differences in SQL generation

The DB2 optimizer can use an MQT for a distributive measure if the query matches the MQT slice or the one above it, while a non-distributive measure must match the MQT slice exactly. As depicted in Figure 6-68 on page 237, all but distinct measures are distributive in DB2 for z/OS, which allows more queries to be routed to MQTs.

Incremental refresh for MQTs is not supported on DB2 for z/OS, and as the third column of Figure 6-68 on page 237 indicates, measures on z/OS can not be incrementally maintained. MQTs must be maintained with a full refresh for system-maintained MQTs, or by the user for user-maintained MQTs.



Measures – DB2 LUW Support

Measure	Distributive	Non-distributive	Inc-maintainable
Count	x		x
Sum	x		x
Max	x		
Avg		x	
Stddev		x	
Distinct		x	

Measures – DB2 Z Support

Measure	Distributive	Non-distributive	Inc-maintainable
Count	x		N/A
Sum	x		N/A
Max	x		N/A
Avg	x		N/A
Stddev	x		N/A
Distinct		x	N/A

Figure 6-68 Distributive versus non-distributive measures

The administration console for InfoSphere Warehouser for System z has been re-written using Adobe® Flex. An example is shown in Figure 6-69 on page 238. For Cubing Services, the administration console is used perform the following tasks:

- Manage cube servers

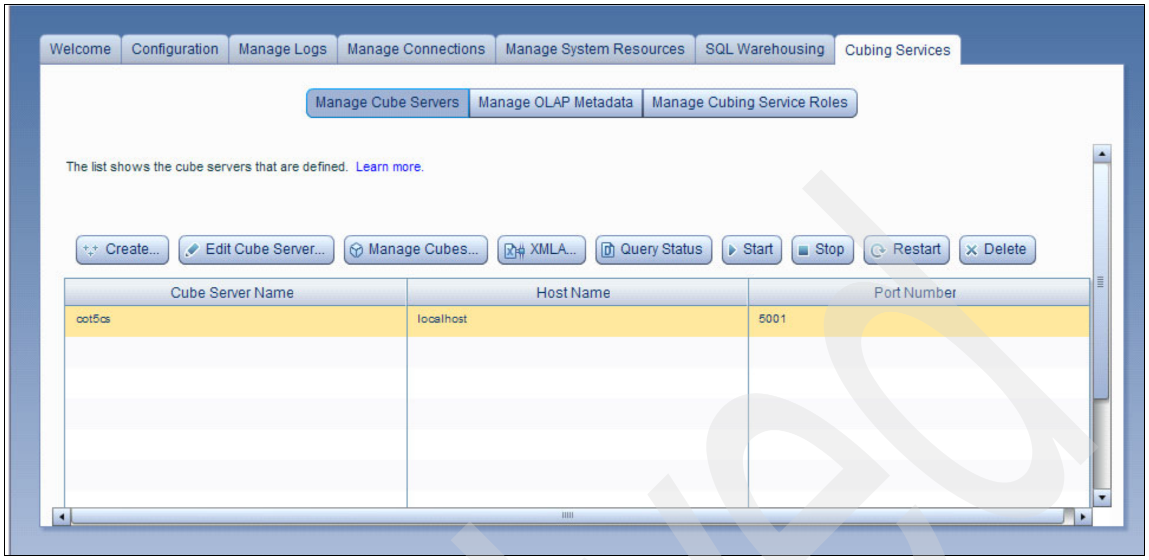
  Create, start, stop cube servers and cubes, and assign cubes to cube servers.

- Manage OLAP metadata

  Import OLAP metadata from XML files, map cubes to data sources, and assign cubes to cube servers.

- Manage Cubing Services roles

  Create user-defined roles and assign users to those roles.



The list shows the cube servers that are defined. Learn more.

Cube Server Name	Host Name	Port Number
cot5cs	localhost	5001

Figure 6-69 Cubing Services administrative services

While the user interface is new, the functions for Cubing Services are basically the same as in the LUW version, with the exception of two functions that did not make it into this release. The Cubing Services Optimization Advisor cannot be invoked from the administration console, and SQL scripts cannot be executed from the administration console. At this time, the Optimization Advisor has to be invoked from Design Studio and might require the runtime administrator to install the Design Studio client. The function to run SQL scripts from the administration console was there to execute the scripts generated by the Optimization Advisor. These scripts can be run from other tools, such as the DB2 command line, QMF™, or SPUFII.

6.8.2 Standard practices

The standard practices and design techniques documented in other parts of this book, and in IBM Redbook, *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582, generally apply to Cubing Services on System z. However, there are settings, database design, and troubleshooting recommendations that are specific to System z.

In general, the best performance experience is when the MDX query can be serviced from the cube server cache. If the query cannot be satisfied from the cache, the cube server generates SQL to retrieve the data from DB2 to add to the cache. At cube start, and after specifically clearing the data cache, most of the MDX requests result in SQL queries to DB2 until such time that the cache is adequately populated such that it satisfies most request from the cache. Therefore, the performance of queries to DB2 affects the overall performance of the cube server.

The primary tuning capability for a cube server comes when you ensure that the appropriate MQTs exist to support the cube. The recommendations of the Cubing Services Optimization Advisor is the starting point for overall cube performance. In addition, the recommendations in the following sections should also be considered.

Recommended DB2 ZPARM settings

CDSSRDEF enables query parallelism. Partitioning any large fact tables and setting the degrees of parallelism reduces the elapsed time when the base tables have to be accessed.

Enable star joins by setting STARJOIN and, depending on workload, adjust the star join enablement threshold from the default of 10 by using SJTABLES.

The routing of queries to MQTs is key to overall Cubing Services performance. The REFSHAGE and MAINTYPE parameters must be set to allow automatic query rewrite to be considered by the DB2 optimizer.

Data warehouse database design suggestions

The following sections describe a number of design suggestions for the data warehouse database.

Adjust the MQT definitions

The MQT recommendations suggested by the Cubing Services Optimization Advisor might not cover all possible slices, with a result that certain queries might not be routed to an MQT. If this happens to be a path that is common for the users, additional MQTs might need to be created to improve the performance of the cube.

Use separate buffer pools for tables and indexes

In a star schema database where the fact table size is usually large, it is suggested to place the data and indexes for the fact table into separate buffer pools. This improves the query elapsed time by increasing the buffer pool hit ratio. This also has the advantage of making it easier to use the DB2 statistics and accounting traces to identify performance bottlenecks and to perform diagnostics.

Adjust work file buffer pools and datasets

If the query issued by the cube server must access the base fact table, it might result in intensive sort activity. To improve sort performance, put the work files database, DSNDB07, into separate buffer pools and increase the size of the buffer pools as system resources allow. In addition, to reduce file contention for parallel sorting, additional work file datasets might be required.

Compress data and indexes

While the usual impetus for compression is to decrease the cost of storage, an increase in performance might also be obtained. This is due to the significant reduction of I/O suspension time with minimal CPU overhead. Decoding of compressed data for reading is much less costly that compressing the data when writing data. In a data warehouse, most access to the databases is for reading, with a relatively smaller amount of updates that reduces the comparative of data compression.

Partition data

Using classic partitioned tablespaces for large tables and data partitioned secondary indexes (DPSI) improves SQL query response time.

Troubleshooting cube performance

There might be occasions when certain paths through the data have performance issues. In those cases, it might be useful to apply the following methodology to troubleshoot the SQL performance issues.

1. Use the administration console to turn on the SQL trace and, optionally, the MDX trace for this cube server.
2. Execute the workload that is having performance issues to capture the MDX SQL statements in the SQL log file, `SQLLog.txt`, found in `<installation directory>/CubingServices/<cube server name>/Logs`.
3. Extract the SQL statements from the SQL trace log.
4. Run the DB2 Explain facility on the extracted SQL statements. Check to see if any queries are not matching MQTs and determine which MQTs are being used by the queries.
5. Assess whether any of the queries not matching MQTs might benefit by manually creating one or more MQTs.
6. If the extracted workload is enough to represent a typical workload, it could be helpful to drop MQTs that are not being used. This saves storage, as well as CPU costs for refreshing.
7. For any new MQTs and SQL queries that can not be helped with an MQT, use Optimization Expert to get recommendations for indexes.

8. Use RUNSTATS to update catalog statistics on any new MQTs. For more information, see the following resources
   - IBM Redbooks publication IBM DB2 9 for z/OS: New Tools for Query Optimization, SG24-7421, available from the following location:

     `http://www.redbooks.ibm.com/abstracts/sg247421.html?Open`
   - The following URL will take you to the directory entry for the DB2 Optimization Expert for z/OS. From there you can follow each of the links for more information on each of those related topics.

     `http://www-01.ibm.com/software//data/db2imstools/db2tools/opti-expert-zos/`

7

Data mining

Data mining used to be the exclusive province of highly-trained statisticians and mathematicians. Today, most businesses need to enable many business analysts to pose business questions and answer them using methodologies based in analytics. Embedded data mining is an approach to making sophisticated data mining methodology and technology available to large numbers of business users and enabling them to solve business problems or take better advantage of business opportunities.

In this chapter we discuss data mining and its implementation in the InfoSphere Warehouse Design Studio. This includes a description of the data mining process and the concept of embedded data mining for deploying mining-based business intelligence (BI) solutions to large numbers of business users. We also discuss the DB2-based components of InfoSphere Warehouse Data Mining for building, applying, and visualizing models in the database. We also show how mining flows are developed and used in conjunction with other flows and functions in Design Studio to support advanced analytics in BI applications for solving high-value business problems.

7.1 Data mining overview

Data mining means different things to different people. Traditional statistics, Online Analytical Processing (OLAP), and database query are sometimes referred to as data mining in the sense that they are tools or techniques for data analysis. These methods, however, are hypothesis driven, implying that we understand the system well enough to formulate precise queries or specify and test explicit hypotheses about relationships in the data.

In the context of the discussion here, we begin with the premise that we do not know what patterns or relationships exist in the data. Rather, we might ask broad questions such as “What do my clients look like?” and leave it up to the data mining algorithms to tell us what the patterns are.

Thus, we define data mining as the process of discovering and modeling non-trivial, potentially valuable patterns and relationships hidden in data. Data mining is discovery driven, meaning that these techniques can find and characterize relationships that are unknown and therefore cannot be expressed explicitly.

Data mining techniques can be divided into two broad groups:

- Discovery techniques
- Predictive techniques

For more information, see the IBM Redbooks publication SG24-7418, *Dynamic Warehousing: Data Mining Made Easy*.

7.1.1 Discovery data mining

Discovery methods are data mining techniques that find patterns that exist in the data, but without any prior knowledge of what those patterns might be. That is, the objective is to discover the relationships that are inherent in the data. There are three discover methods:

- Clustering

  The clustering method groups data records into segments by how similar they are based on the attributes of interest. Clustering could be used, for example, to find distinct profiles of clients with similar behavioral and demographic attributes to create a client segmentation model.

- Associations

  Associations is a type of link analysis that finds links or associations among the data records of single transactions. A common use of the associations method is for market basket analysis. This is to find what items tend to be purchased together in a single market basket, such as chips and soda, for example.

- Sequences

  Another type of link analysis, sequential patterns, finds associations among data records, but across sequential transactions. A store could use sequential patterns to analyze purchases over time and, at checkout time, use that model to print customized discount coupons for the customers to use on their next visit.

7.1.2 Predictive data mining

The predictive methods are data mining techniques that can help predict categorical or numeric values. There are three predictive methods:

- Classification

  Classification is used to predict values that fall into predefined buckets or categories. For example, they can predict whether a particular treatment will cure, harm, or have no effect on a particular patient.

- Regression

  In contrast, regression is used to predict a numerical value on a continuous scale. For example, predicting how much each customer will spend in a year. If the range of values is 0 to 1, this becomes a probability of an event happening, such as the likelihood of a customer leaving, for example.

- Time series forecasting

  Time series forecasting predicts values of a numerical data field for a future period in time. You can use a time series model to predict future events based on known past events. For example, forecasting can be used to create stock level forecasts to reduce warehouse costs.

For more detailed descriptions on the discovery and predictive mining methods available in InfoSphere Warehouse (ISW), see the IBM Redbooks publication SG24-7418, *Dynamic Warehousing: Data Mining Made Easy*.

7.2 The data mining process

For many people, data mining is a strange and almost mythical phenomenon. It is, therefore, a topic that is not well understood by most. This is because it can involve many technical mathematical algorithms and might require a detailed and complex analysis of the results.

For these reasons, data mining has not been as widely used as it could be, and the value that can be gained from data mining has not been fully realized. That is about to change. One way to start that change is to make the use of data mining easier. This can be done by embedding much of the data mining process in applications. Another is to de-mystify the process, making it more understandable to people.

7.2.1 Understanding the business problem

The data mining process begins not with data but with a problem to be solved. Because the purpose of this process is to generate results (discoveries, insights, information, models) that helps to solve the business problem, you must begin by clearly stating the business problem, translating it into one or more questions that data mining can address, and understand how the data mining results are used in a BI solution to improve the business.

Data mining has many high-value uses, but it is not always the right tool for a particular purpose. For example, if the business need is to make the sales force more productive by reducing the time required to gather certain information, then the right solution might be a new reporting system based on real-time, multidimensional queries. One the other hand, if the need is to understand client behaviors and preferences to improve targeting or promotional campaigns, the right solution might be based on client segmentation and link analysis (associations and sequences). Understanding the business problem and how the data mining results are deployed into the business enable us to determine the best approach to solving the problem.

Stating the business problem clearly and concisely forces us to focus on the root issue of what needs to change to enable us to accomplish a specific goal. After you identify the root issue and determine that data mining is appropriate, you can formulate one or more specific questions to address that issue. For example, to design a new targeted promotion, you can ask the following questions:

- What are the distinct profiles of behavioral and demographic attributes of our clients?
- Which of these client segments looks best to target for the product that you want to promote?
- For the target segment, what are the right client attributes to use in designing an appealing promotion?

Data mining is used successfully in many different business and scientific fields including retail, banking, insurance, telecommunications, manufacturing, health care, and pharmaceuticals. For example, in the retail, financial, and telecom industries, well-known uses of data mining include client segmentation for targeted marketing and fraud detection, store profiling for category and inventory management, associations for cross-selling and upselling, and predictive techniques for client retention and risk management. In manufacturing, data mining applications include profiling and predictive methods for quality assurance, warranty claims mitigation, and risk assessment. In health care and pharmaceutical research, high-value uses include associations, sequences, and predictive techniques for disease management, associations and prediction for cost management, and patient segmentation for clinical trials. In every case, the data mining results must be conveyed in such a way that someone can make a better decision or formulate an action to solve the business problem.

7.2.2 Identifying the data mining approach

Understanding the business problem and how the data mining results are used guides us in identifying the data mining approach and techniques that are most suitable to address the problem. In certain cases, a combination of data mining techniques is most appropriate. For example, if the business issue is to improve the effectiveness of a marketing campaign for an upcoming back-to-school promotion, we might decide to perform client segmentation using the clustering technique to identify a target group of clients, followed by a market basket analysis using the association's technique with the transactions only for the target group to find item affinities on which to base the promotion.

In practice, the choice of data mining approach might be driven in part by the data available or the cost of transforming the available data to a suitable format. For example, if the data are available in transactional format and a client segmentation to identify cross-selling opportunities is the technique, then the transactional data must be aggregated and transformed to the client level (one record per client in demographic format).

If this data preparation procedure is not practical in the scope or budget of the data mining project, a mining technique that uses transactional data (associations or sequences) might be selected instead. In this sense, the steps of identifying the data mining approach and identifying and understanding the data are closely interrelated.

7.2.3 Understanding the data

Understanding the data resources that are available to support data mining is integral to identifying the right analytical approach to solve a business problem. In most enterprises, vast amounts of data are collected and stored, typically in data warehouses and often in dedicated data marts, to support various applications. As a central repository of data resources, the data warehouse can improve the efficiency of preparing data for data mining. Additional demographic data might be available from third-party or governmental sources to supplement the existing data and enrich the data mining analysis.

For any application, the data warehouse or specific data mart contains information (metadata) about the data, how it was derived, cleansed, and transformed, how it was formatted, and so on. The data and metadata form a data model that supports the application and defines the data sources, types, content, description, usage, validity, and transformations.

Like any application, data mining requires a data model to specify the data structure for each data mining technique. In most cases, the data model calls for a single denormalized table as the primary data table with relational tables containing additional information (such as name mappings or hierarchies). The denormalized table might be either a physical table or a view of joined tables. The following list details the two most common data types used in data mining:

- Transactional data

  Operational data generated each time an interaction occurs with the individual (target). This typically contains a time stamp, target identifier, and item identifier (where an item might represent a product, event, or other attribute). It is data suitable for link analysis (associations and sequences) where you want multiple records per entity (such as per transaction or per person/event).

- Behavioral/Demographic data

  This is data about individuals' characteristics, such as age, gender, location, quantities purchased, amount spent, and number of interactions. This data is suitable for analysis where the entity of interest is an individual person, event, or system, and there should be one record per entity with many attributes describing the entity. Demographic data refers to non-behavioral characteristics.

The first task in data preparation is to identify what data elements are needed to accomplish the analysis using the chosen mining method. For example, if you are doing customer profiling, you need basic information about the customer along with other relevant data appropriate to the problem, perhaps the total of their purchases over the last twelve months.

When the required data elements are identified, an inventory of those data elements needs to be generated. Do the data elements exist, and if so, where? If not, can they be obtained from an external source? If the data elements exist in the data warehouse that reduces the data preparation time. Even so, additional data preparation is required.

If the data required for the chosen mining method does not exist and cannot be obtained, the chosen analytic approach might have to be revisited and modified.

7.2.4 Preparing the data

Data preparation for data mining addresses an important aspect of data mining projects. Without adequate preparation of data, the return on the resources invested in mining projects is uncertain and could be disappointing. The data preparation process creates the input variables for data mining models. These variables represent the data in a context than can be explored by data mining tools, which are used to produce insightful information to the business.

Understanding of data should not be limited to elements such as data models, tables, field names, and join keys. The data analyst should also be concerned with, and understand, data domains for elements such as categorical fields, missing values, taxonomies, outliers, data integrity, and data granularity. All these aspects should be evaluated and considered in transformation tasks for data preparation.

InfoSphere Warehouse Design Studio has a set of operators to support data transformations for data mining and also offers specific functions for data exploration (for example, univariate, bivariate, and multivariate distributions) to assist analysts in assessing data quality and gaining deep insights about the data. Details about InfoSphere Warehouse Design Studio functions to support data preparation and exploration are described in 7.5.2, “Data exploration” on page 259 and 7.5.3, “Data preparation” on page 266.

7.2.5 Building the data mining model

After the data have been prepared, the data mining model is created. The model is built to execute the desired technique with appropriate parameters. Modeling is an interactive and iterative process as the initial results are reviewed, model parameters are adjusted to produce a better model, and any additional data preparation is performed.

For a predictive technique, training and testing sets are used for model building and validation, respectively. Overfitting is a potential problem when building a predictive model. A high degree of overfitting means that the model closely fits the training data but is not accurate when applied to new data. Because you want to use the model to predict outcomes that have not happened yet, there is particular interest in how accurate the model is when applied to new data. Thus, you can train the model with a training data set and validate the model using a testing data set that is statistically the same as the training data.

Both the training and testing sets consist of historical data, meaning that they both must contain the outcome (target field) to construct and validate a predictive model. InfoSphere Warehouse Data Mining has a function to split the input data randomly into training and testing sets and to perform the modeling and validation steps automatically. In Data Mining, the mining algorithms are implemented with internal checks to avoid overfitting a model during training.

The modeling process produces a data mining model that is stored in Predictive Model Markup Language (PMML) format in a designated DB2 table. PMML is an XML-based language for defining statistical and data mining models so that they can be shared and used by PMML-compliant applications.

Each DB2 database used for data mining must be enabled for mining. To enable a database for mining means that the DB2 extenders for data mining are activated for the database and the necessary tables to support the mining are created. When a database is enabled for data mining, a set of tables under the schema IDMMX is created to contain information needed by DB2 to execute the data mining procedures. Four of these tables contain the PMML models by type. The PMML models in these tables are available to InfoSphere Warehouse for deployment, application, or visualization. They also can be accessed by PMML-compliant applications such as an Alphablox application or a third-party application.

7.2.6 Interpreting and evaluating the data mining results

InfoSphere Warehouse Data Mining allows a model to be visualized for interpretation and evaluation. Tailored to each type of data mining model, the visualizations present information about model quality, specific results such as association rules or clusters, and other information about the data and results pertinent to the particular model. This information enables the data mining analyst to assess model quality and determine whether the model fulfills its business purpose. Improvements to the input data, model parameters, and modeling technique can be made to obtain a model that meets the objective.

For example, a clustering model intended to find high-potential clients to target for a new family plan cellular phone offer has one large cluster that contains 90% of all the clients with the remainder distributed across several small clusters. Because it provides little distinctive information about the clients, the model is of poor quality and does not meet the objective of identifying high-potential clients with distinct behavioral and demographic profiles to support designing a promotional offer. Adjusting certain parameters and re-executing the mining run yields a model with clusters ranging in size from 30% to 1%, providing a set of distinct profiles, each with enough clients to be meaningful in terms of the business objective.

As another example, a classification model intended to screen a hospital patient database to identify those at high risk of developing a certain disease fits the training data extremely well and is highly predictive of the propensity to develop the disease. But one of the key predictors in the model is a costly test not covered by insurance except for patients already diagnosed with the disease. Thus, this variable is unknown for most patients in the database. Although this model is of high quality, it does not meet the objective of an early warning indicator for patients who have not developed the disease. Eliminating that predictor from the model yields a less accurate but more useful model that can be widely deployed to screen patients and identify those who would benefit from preventive treatment before they develop the disease.

As a third example, a regression model intended to predict the likelihood that a client will default on a new bank loan fits the training data well, is highly accurate with the testing data, and contains only those variables that are readily attainable at the time of loan application. But when the model predicts that a loan applicant has a high propensity to default, the loan officer is unable to state precisely why the loan application must be declined. Although this model is accurate and useful, the bank legal department points out that the model does not meet the business need to cite specific reasons for declining a loan.

Building a decision tree classification model using the same data and variables yields a model of slightly lower accuracy, but has the advantage of explicit decision paths for predicting the likelihood that a loan applicant will default, thereby enabling the loan officer to meet the legal requirement of stating exactly why a loan application is declined.

The ultimate measure of model quality is the usefulness of a particular model for addressing the business problem at hand. For example, a clustering model having 10 clusters might be of high quality in a technical sense (per a calculated measure of how closely the records fit the assigned cluster and how well-separated the different clusters are) but be of limited usefulness to a business analyst who needs a simpler model having only four clusters on which to base a new promotional campaign. A better model in this case might have a lower calculated quality metric but would have the required number of clusters and, therefore, would better meet the business requirements.

7.2.7 Deploying the data mining results

The final step in the data mining process is to deploy the data mining results in the business as part of a BI solution. Deployment might be the most important step in the data mining process because how and where the results are deployed are crucial to realizing the maximum value from the data mining.

Data mining results can be deployed in various ways to diverse business processes or systems, depending on the business needs:

- Ad hoc insight

  Use data mining on an ad hoc basis to address a specific, nonrecurring question. For example, a pharmaceutical researcher might use data mining techniques to discover a relationship between gene counts and disease state for a cancer research project.

- Interactive insights

  Incorporate data mining into a BI application for interactive analysis. For example, a business analyst might regularly use a targeted marketing application to segment the client base, select a segment suitable for the next promotion, and perform market basket analysis specific to that segment. The item affinities and client profile then analytical and reporting functions that identify the highest-value item relationships for the promotion and the best promotional channel given the characteristics of the target group.

- Scoring

  Apply a data mining model to generate a prediction for each record, depending on the type of model. For a clustering model, the score is the best-fit cluster for each individual. For an associations model, the score is the highest-affinity item, given other items. For a sequences model, the score is the most likely action to occur next. For a predictive model, the score is the predicted value or response.

  – Batch scoring

    Embed a scoring function in a BI application for periodic scoring of a database using a data mining model and generating an action or alert when required. An application might be set up to periodically score a database of clients and proactively trigger an action in response to a change in an individual’s score. For example, when an insurance client’s life situation changes, such as having a child, the application registers a change in that client’s scored propensity to purchase life insurance. An informational letter is automatically sent to the client, and the agent is notified to follow up with an offer for a new or upgraded life insurance policy.

  – Real-time scoring

    Embed a scoring function in a BI application for real-time, on-demand scoring of an individual during an interaction and immediately generate a recommended offer or action. For example, a call center client service representative might call up a client’s record during a phone conversation, enter the current order, and receive a recommendation for a suitable item to offer the client to complement the item just ordered.

7.3 Embedded data mining

Leading-edge companies have profited greatly from employing data mining to help them make better business decisions. But this technology is complex and has traditionally been the domain of expert statisticians working in an environment separate from the central data repository and BI infrastructure.

Businesses today are looking for ways to make data mining accessible to large numbers of line-of-business analysts and decision makers throughout the enterprise. *Embedded data mining* is the concept of delivering data mining to business analysts through portals, reporting tools, and customized applications tailored to specific business areas, all in the database environment.

With embedded data mining components implemented as DB2 extenders for model building and application, InfoSphere Warehouse facilitates the development and enterprise-wide deployment of data mining-based solutions by enabling data mining functions to be integrated into BI tools. Because data mining capabilities are implemented as DB2 extenders, any reporting or query tool that can export SQL can become a data mining platform. Business analysts who are not accomplished statisticians can bring the power of data mining to bear on many different business problems.

Embedded mining is based primarily on discovery data mining methods (associations, sequential patterns, and clustering) that are relatively easy to execute and interpret and so lend themselves well to incorporation into business applications for business analysts. Predictive data mining methods (classification, value prediction) also can be implemented through embedded mining, but these methods are more complex to execute (particularly regarding parameter specifications) and interpret.

For predictive methods, embedded mining might be most useful to business analysts for scoring using models created by statisticians. For more information about embedded data mining, see the articles *Accessible Insight: Embedded Data Mining* and *Embedded Data Mining: Steps to Success*, listed in "Related publications" on page 597.

For embedded data mining, the data warehouse is the single, centralized, and comprehensive data source for integrated BI analytical and related functionality. Through centralized management of data resources, many of the problems, errors, and costs inherent in moving and duplicating data can be avoided. Working directly in the database also enables the use of high-throughput, scalable data warehouse platforms for BI analytics.

BI analytics can be effectively supported by domain-specific or project-specific data structures in the overall data warehouse, reducing data manipulation costs. For example, to support an ongoing client segmentation project, a data mart could be established to perform the required data transformations to populate and refresh a table containing the client data in the correct structure for the analysis. Another data mart could likewise support market basket analysis, which requires a different data structure than client segmentation. By maintaining and refreshing the data for different analytical uses, data marts can greatly reduce the overall analysis time.

The need to integrate mining into the database, with both embedded data mining and traditional data mining (working outside the central database environment) has led to the creation of standards establishing PMML and a standard SQL interface to DB2's data mining capabilities. PMML provides a way to express a data mining model as an XML object that can be ported between analytical environments without having to rebuild or recode the model.

This capability enables analysts to create data mining models and deploy them in their native forms in the data warehouse where they can be used by applications. PMML also facilitates model refreshing without recoding and retesting, greatly reducing the time to implement refreshed models and ensuring that business analysts are able to use up-to-date models in their decision making. Many data mining vendors have embedded DB2-based mining in their analytical and reporting tools. For more information about PMML see the *The Data Mining Group* listed in “Related publications” on page 597.

7.4 InfoSphere Warehouse data mining components

InfoSphere Warehouse has three components for DB2-based data mining. These are called Modeling, Visualization, and Scoring. For more information about these components see the IBM Redbook *Enhance Your Business Applications: Simple Integration of Advanced Data Mining Functions,* SG24-6879, listed in section, “Related publications” on page 597. Also see the Data Mining section of the DB2 Information Center installed as part of InfoSphere Warehouse.

7.4.1 Modeling

Modeling is a DB2 SQL application programming interface (API) implemented as a DB2 extender. Modeling is accessed graphically through the InfoSphere Warehouse Design Studio to build data mining models from information residing in DB2 databases.

Modeling offers six data mining methods. Most of the methods have multiple algorithms to provide more flexibility in model development. The data mining methods are:

- Associations: find item affinities or links in transactions or events.
- Sequences: find item affinities or links across transactions or events.
- Clustering: find distinct groups or segments.
- Classification: predict a categorical response or outcome.
- Regression: predict a numeric outcome or likelihood of response.
- Time series: forecast future values of a numeric time series

InfoSphere Warehouse Modeling, a DB2 extender, provides a set of SQL stored procedures and user-defined functions to build a model and store it in a DB2 table. These procedures and functions are collectively referred to as *easy mining procedures*. As the model is set up through graphical wizards in Design Studio, the easy mining procedures use the wizard inputs automatically to create mining tasks that specify the type of model to be built, the parameter settings, the data location, and data settings (for example, which columns to use in the model) and to call the appropriate mining kernel in DB2.

A DB2 table containing columns representing the behaviors and other attributes of the records (such as clients, stores, accounts, and machines), including the response or outcome, if any, is used as the data source for building (training) the model and, for predictive mining, validating (testing) the model as well. As described in section 7.2.5, "Building the data mining model" on page 249, the new model is stored in PMML format in a DB2 table where it is accessible for deployment.

7.4.2 Visualization

After a data mining model has been created, the analyst can explore the model using visualization. Visualization is a Java application that uses SQL to call and graphically display PMML models, enabling the analyst to assess a model's quality, decide how to improve the model by adjusting model content or parameters, and interpret the final model results for business value.

Visualization has visualizers for all six mining methods in the modeling. The visualizers are tailored to each mining method and provide a variety of graphical and tabular information for model quality assessment and interpretation in light of the business problem.

Visualization can also display PMML models generated by other tools if the models contain appropriate visualization extensions such as quality information or distribution statistics as produced by modeling. These model types do not contain much extended information and do not present well in visualization.

7.4.3 Scoring

Like modeling, scoring is implemented as a DB2 extender. It enables application programs using the SQL API to apply PMML models to large databases, subsets of databases, or single records. Because the focus of the PMML standard is interoperability for scoring, scoring supports all the model types created by modeling as well as selected model types generated by other applications (for example, SAS and SPSS) that support PMML models:

- Associations
- Sequences
- Clustering (distribution-based, center-based)
- Classification (decision tree, neural, logistic regression)
- Regression (linear, polynomial, transform, neural)

Scoring also supports the radial basis function (RBF) prediction technique, which is not yet part of PMML. RBF models can be expressed in XML format, enabling them to be used with scoring.

Scoring includes scoring JavaBeans, enabling the scoring of a single data record in a Java application. This capability can be used to integrate scoring into client-facing or e-business applications, for example.

A PMML model is either created and automatically stored by modeling or created by another application and imported into DB2 using Scoring's SQL import function. Accessed through Design Studio, scoring applies the PMML model to new data. The score assigned to each record depends on the type of mining model. For example, with a clustering model, the score is the best-fit cluster for a given record. The results are written to a new DB2 table or view, where they are available to other applications for reporting or further analysis such as OLAP. Because it exploits DB2 parallel processing, scoring is faster than single-threaded scoring routines.

7.5 Data mining in Design Studio

Design Studio is the interface to data exploration and data mining in InfoSphere Warehouse. In this section we discuss the process and functions for developing mining flows in Design Studio. We also discuss useful functionality for data exploration as a way to better understand the data as it is prepared for mining.

7.5.1 Database enablement

As explained in 7.2.5, “Building the data mining model” on page 249, each database used for data mining must be enabled for mining, which is easily done in Design Studio. To enable a database, locate the database in the Connections folder in the Data Source Explorer on the Design Studio window and connect to the database. As illustrated in Figure 7-1, right-click the blue database icon, and select **Enable the database for Data Mining**.

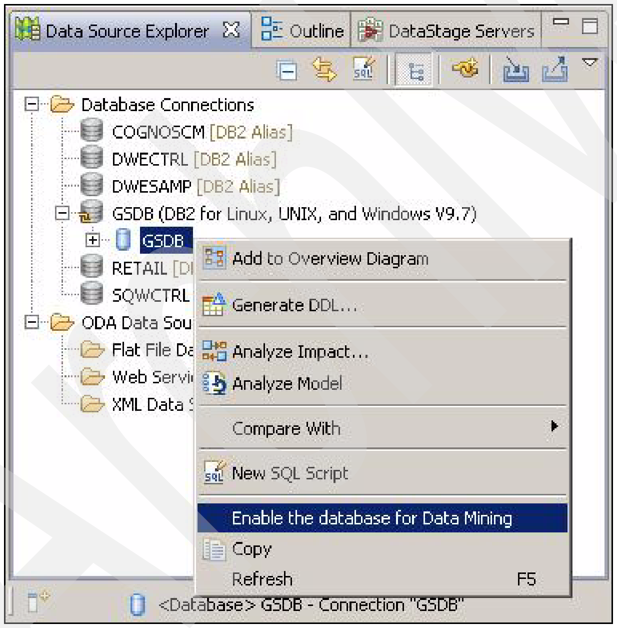


Figure 7-1 Enabling a database for data mining in the Data Source Explorer

When a database is enabled, the schema IDMMX and a set of tables used by the mining functions is automatically created, as illustrated in Figure 7-2. Most of these tables are repositories for settings and specifications used by modeling and scoring. Four of the tables are used to store PMML models:

- IDMMX.RULEMODELS: associations and sequences models
- IDMMX.CLUSTERMODELS: Clustering models
- IDMMX.CLASSIFMODELS” Classification models
- IDMMX.REGRESSIONMODELS: Prediction models

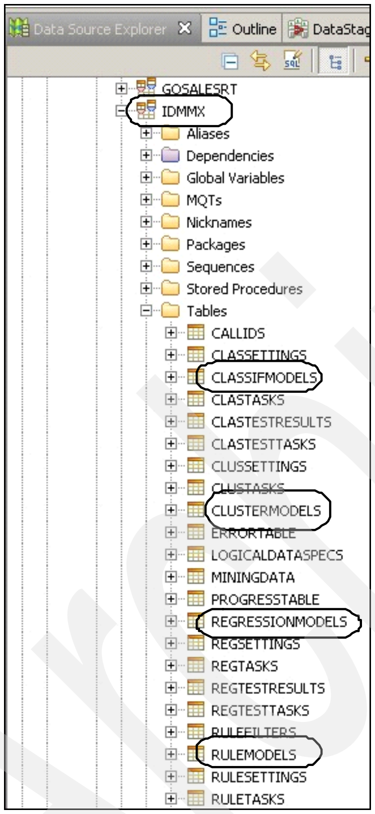


Figure 7-2 Tables created under the schema IDMMX

7.5.2 Data exploration

InfoSphere Warehouse has the functionality to facilitate data exploration as part of the data preparation process. Understanding the data and validating its quality and content are essential to obtaining meaningful results from data mining analysis. In this section, we discuss four capabilities that enable the analyst to investigate data values and distributions quickly and easily through the Data Source Explorer:

- Table sampling
- Univariate distribution analysis. See page 261.
- Bivariate distribution analysis. See page 264.
- Multivariate distribution analysis. See page 265.

Table sampling

Sampling the contents of a table in the Data Source Explorer is illustrated in Figure 7-3. Right-clicking the desired table brings up a menu for selecting **Data → Sample Contents**.

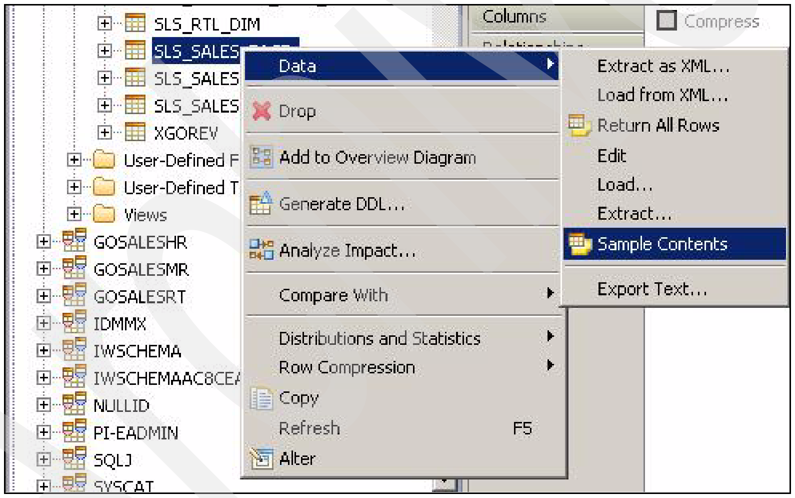
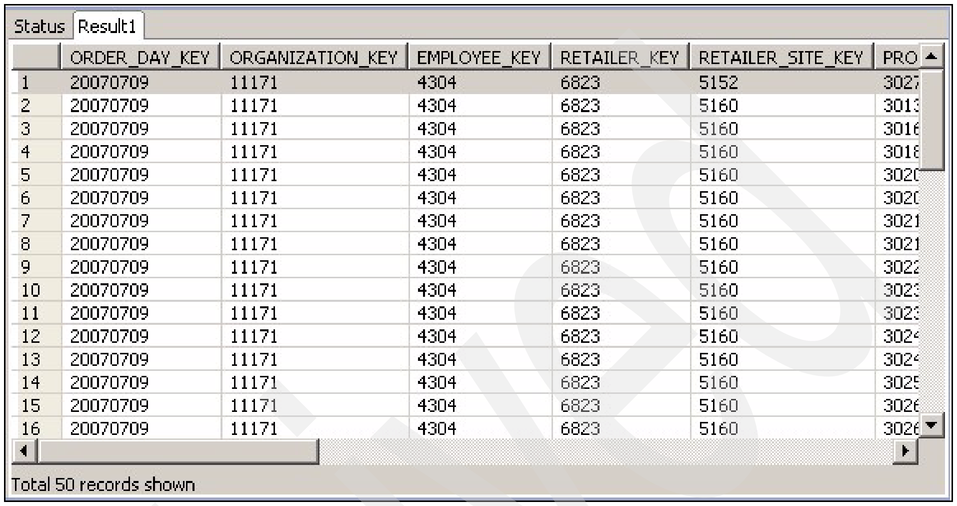


Figure 7-3 Table sampling from the Data Source Explorer

This selection produces a table of all the columns and a sample of the rows in the table, as shown in Figure 7-4. This view appears in the lower right corner of the Design Studio window. The analyst can inspect the sampled rows to get an initial feel for how the data looks.


Status | Result1

	ORDER_DAY_KEY	ORGANIZATION_KEY	EMPLOYEE_KEY	RETAILER_KEY	RETAILER_SITE_KEY	PRO
1	20070709	11171	4304	6823	5152	3027
2	20070709	11171	4304	6823	5160	3013
3	20070709	11171	4304	6823	5160	3016
4	20070709	11171	4304	6823	5160	3018
5	20070709	11171	4304	6823	5160	3020
6	20070709	11171	4304	6823	5160	3020
7	20070709	11171	4304	6823	5160	3021
8	20070709	11171	4304	6823	5160	3021
9	20070709	11171	4304	6823	5160	3022
10	20070709	11171	4304	6823	5160	3023
11	20070709	11171	4304	6823	5160	3023
12	20070709	11171	4304	6823	5160	3024
13	20070709	11171	4304	6823	5160	3024
14	20070709	11171	4304	6823	5160	3025
15	20070709	11171	4304	6823	5160	3026
16	20070709	11171	4304	6823	5160	3026

Total 50 records shown

Figure 7-4 Sample contents of a table

Univariate distribution analysis

More illuminating insights about the data can be gained by inspecting the distributions of the columns in the table. The function to select univariate, bivariate, or multivariate distributions is illustrated in Figure 7-5. Right-clicking the desired table brings up a menu from which you select. **Distributions and statistics**. Select the desired view (Univariate, Bivariate, Multivariate) of the data distributions from the menu that displays.

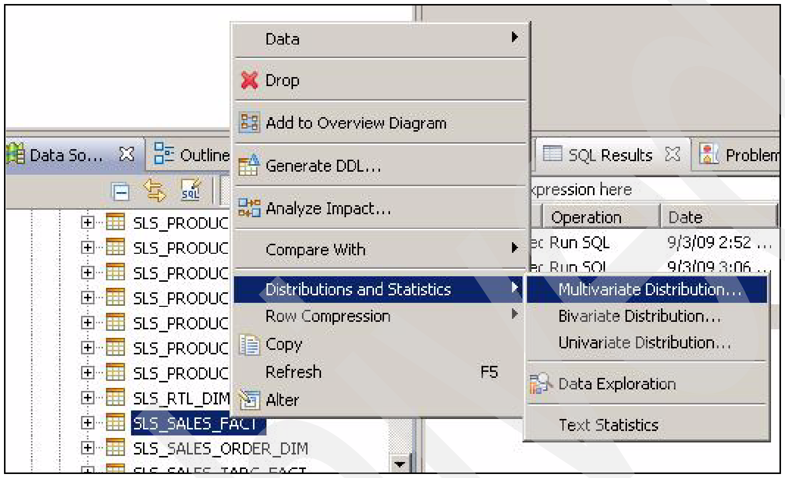


Figure 7-5 SMenu for value distributions

Selecting **Univariate** generates a set of histograms representing the frequency distributions of the individual columns in the table. The distributions are displayed graphically, as shown in Figure 7-6.

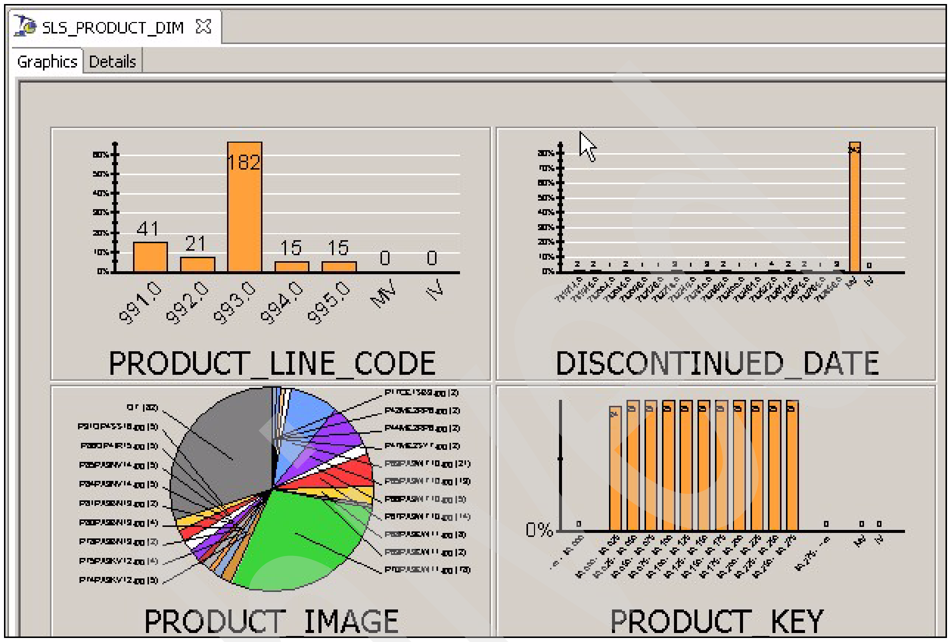
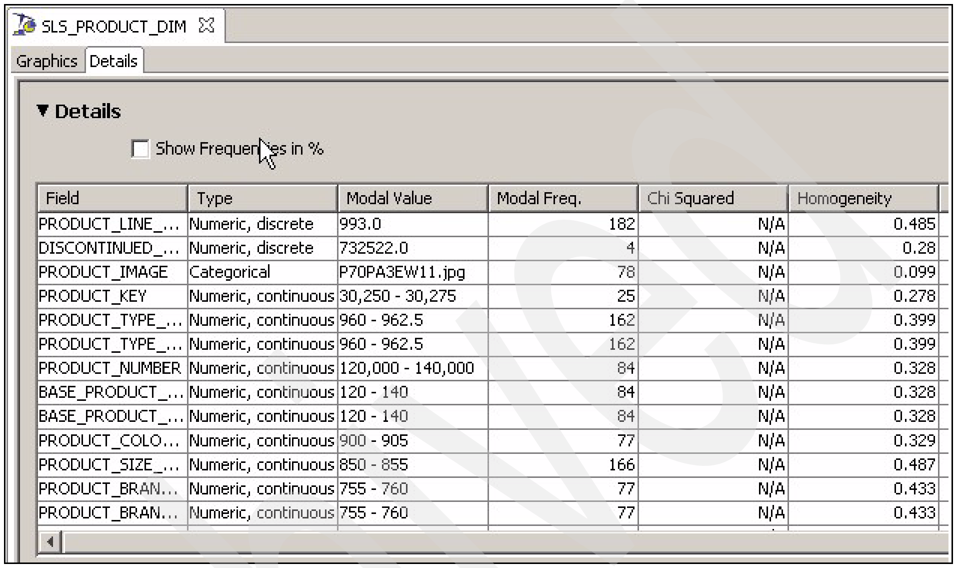


Figure 7-6 Graphical visualization of univariate distributions

They are also displayed in tabular form, as depicted in Figure 7-7. Analyzing the univariate distributions helps the analyst assess data quality and validity for an individual field (column) by revealing potential data problems (such as out-of-range or invalid values) that the analyst should investigate further before proceeding with any analysis.



Field	Type	Modal Value	Modal Freq.	Chi Squared	Homogeneity
PRODUCT_LINE_...	Numeric, discrete	993.0	182	N/A	0.485
DISCONTINUED_...	Numeric, discrete	732522.0	4	N/A	0.28
PRODUCT_IMAGE	Categorical	P70PA3EW11.jpg	78	N/A	0.099
PRODUCT_KEY	Numeric, continuous	30,250 - 30,275	25	N/A	0.278
PRODUCT_TYPE_...	Numeric, continuous	960 - 962.5	162	N/A	0.399
PRODUCT_TYPE_...	Numeric, continuous	960 - 962.5	162	N/A	0.399
PRODUCT_NUMBER	Numeric, continuous	120,000 - 140,000	84	N/A	0.328
BASE_PRODUCT_...	Numeric, continuous	120 - 140	84	N/A	0.328
BASE_PRODUCT_...	Numeric, continuous	120 - 140	84	N/A	0.328
PRODUCT_COLO...	Numeric, continuous	900 - 905	77	N/A	0.329
PRODUCT_SIZE_...	Numeric, continuous	850 - 855	166	N/A	0.487
PRODUCT_BRAN...	Numeric, continuous	755 - 760	77	N/A	0.433
PRODUCT_BRAN...	Numeric, continuous	755 - 760	77	N/A	0.433

Figure 7-7 Tabular display of univariate distributions

Bivariate distribution analysis

Select **Distributions and statistics** → **Bivariate** to generate a set of histograms showing how the data values of each column are distributed with respect to the values of a selected target column. Unlike univariate distributions, bivariate distributions can reveal data inconsistencies between two fields. Figure 7-8 shows the gender distribution for a group of customers.

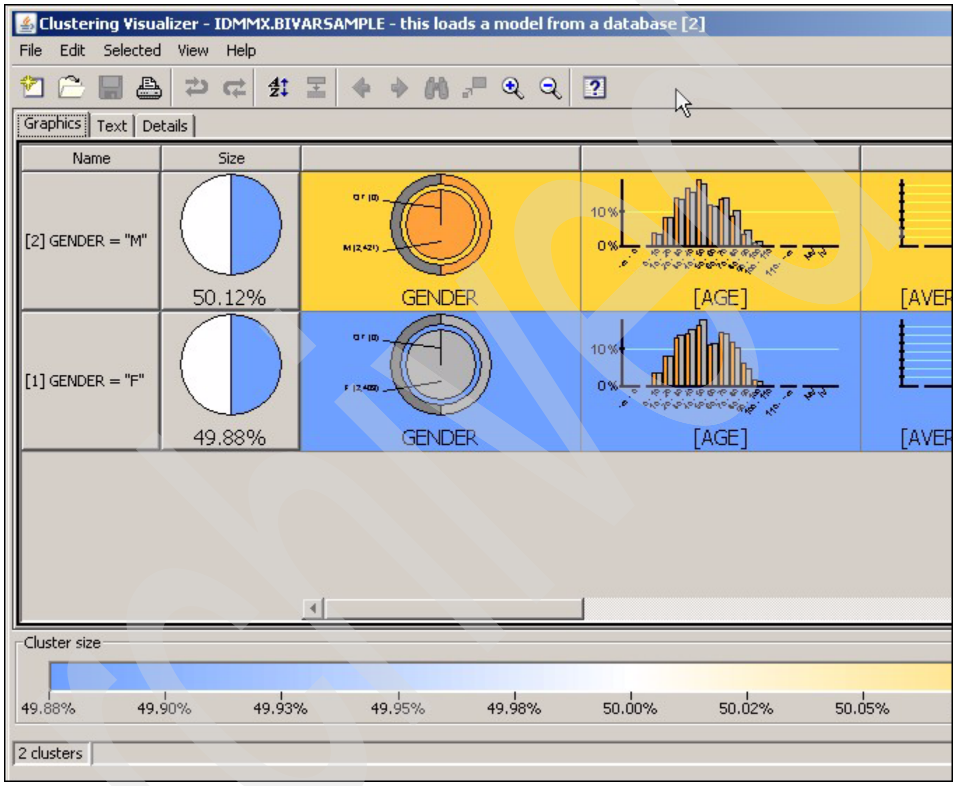
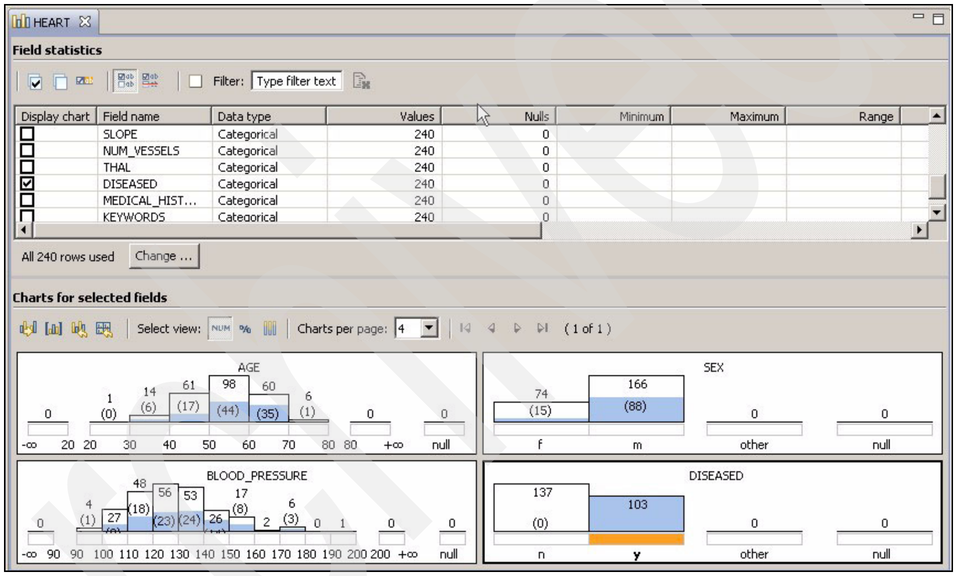


Figure 7-8 Bivariate distributions example

Multivariate distribution analysis

Select **Distributions and statistics** → **Multivariate** to generate a set of histograms showing how data values are distributed across all of the fields in the table. The multivariate view has the capability to highlight how groups of records are distributed across all fields, as illustrated in Figure 7-9. (The distributions are intended only for data exploration and thus, in the interest of execution time to calculate the table sample statistics, are based on a limited number of records.)

The analyst can select a group of records from one field's distribution and see how those records are distributed among the other fields. Multivariate distribution analysis provides insights into the interactions among the fields, which can help the analyst do a better job of data mining modeling and interpretation.



Display chart	Field name	Data type	Values	Nulls	Minimum	Maximum	Range
☐	SLOPE	Categorical	240	0			
☐	NUM_VESSELS	Categorical	240	0			
☐	THAL	Categorical	240	0			
✓	DISEASED	Categorical	240	0			
☐	MEDICAL_HIST...	Categorical	240	0			
☐	KEYWORDS	Categorical	240	0			

Figure 7-9 Multivariate distributions example

In Figure 7-9 we are interested in understanding the characteristics of patients suffering from heart disease. Each record in this data table represents a patient. Selecting the check box next to a field name displays the chart for that field (for example, age and blood pressure). Clicking **Yes** for the Diseased field highlights the histogram bar and displays where those patients (patients with DISEASE = Y) fall in the distributions of the other fields. For example, we see from the DISEASED distribution that 43% of the patients have heart disease (103 patients from a total of 240).

If we then look at the gender of those diseased patients, we can see that 85% of them are male (88 diseased male patients from a total of 103 diseased patients). More detailed exploration can be performed by holding down the Ctrl key and selecting multiple histrogram bars (for example, DISEASED and MALE). This shows how records having both these values are distributed across all fields.

For additional insights, try selecting different charting options by clicking the **Select view** icon. Of particular interest is the third charting option (showing three colored bars) that gives a normalized view. For example, we can see, depicted in Figure 7-10, that 54% of men (81 out of 151) have heart disease versus 19% of women (13 out of 68).

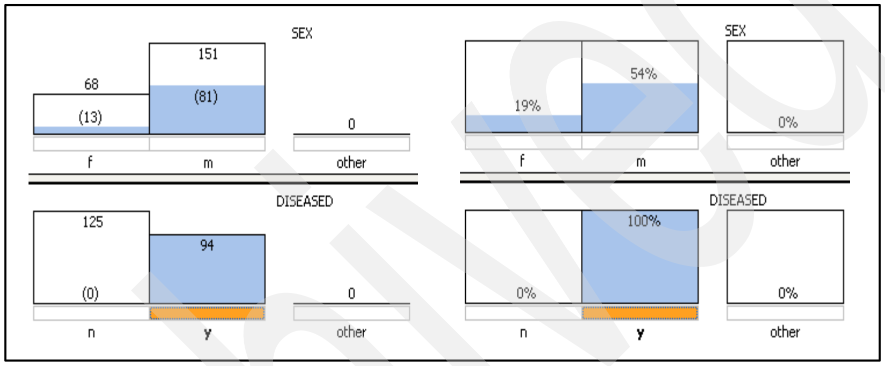


Figure 7-10 Gender and disease

7.5.3 Data preparation

Although modeling is mathematically the most complicated step in the mining process, data preparation usually requires most effort in a data mining project. According to experience, about 40–70% of the time in a data mining project is needed for data preparation.

Data preparation starts at the end of the data understanding phase when the relevant data is understood and its context is known.

This data is not usually ready for immediate analysis for the following reasons:

- Data might not be clean and therefore not suitable for further analysis. In particular, data might be incomplete, wrong, or inconsistent.
- Data might be distributed in many tables, and values might be recorded at an inconvenient granularity for the business purpose at hand.

Data preparation consists of the following major steps:

1. Create the input model.

   The first step is to create the input model. This means to localize and relate the relevant data in the database. This task is usually performed by a DBA or a data warehouse administrator, because it requires knowledge about the database model.

   In this step, the DBA defines semantic concepts such as dimensions and hierarchies. The relevant tables are joined so that the data transformation tasks can be defined by using these semantic concepts.

   If an OLAP model in the form of a Cubing Services model is available, this step can be skipped because the cube model can be imported as the input model.

2. Define a data preparation profile.

   The second step is to define a data preparation profile. This means to determine the focus of analysis and to specify the relevant properties that are to be computed by the data transformation. Because the profile definition can be based on the semantic concepts that are defined in the previous step, it can easily be performed by the mining analyst.

   At the end of this step, a single logical table is defined. This logical table is the starting point for subsequent data mining analysis. It represents the input data in a mining flow. The logical table is used by a data flow or by a generated SQL script. The resulting table of the data flow or the SQL script is used as a table source in a mining flow.

InfoSphere Warehouse Design Studio contains a Data Preparation Wizard to help create input models and data preparation profiles. The wizards helps to prepare the data to ensure that the mining model produces meaningful results.

The Data Preparation Wizard can be accessed from the Data Project Explorer, as shown in Figure 7-11.

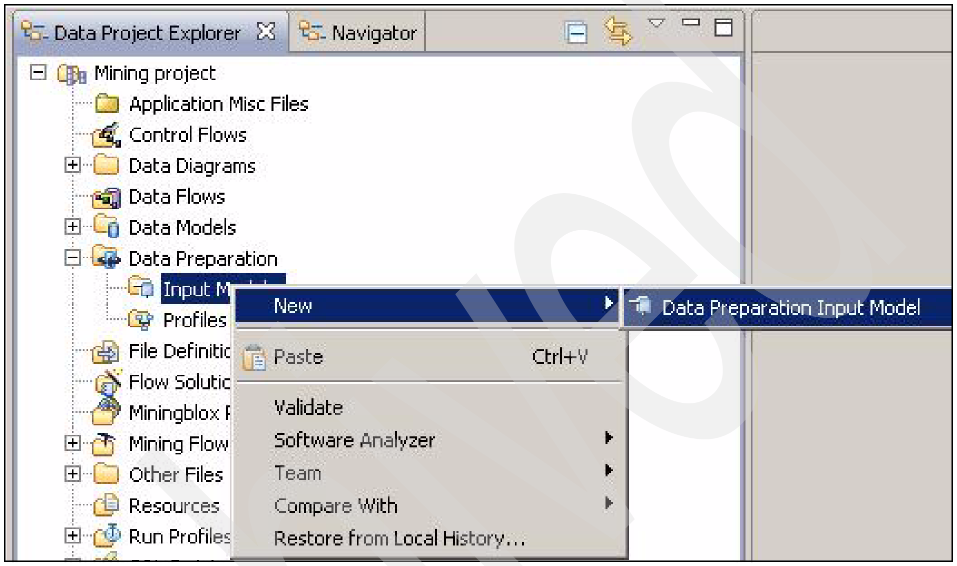
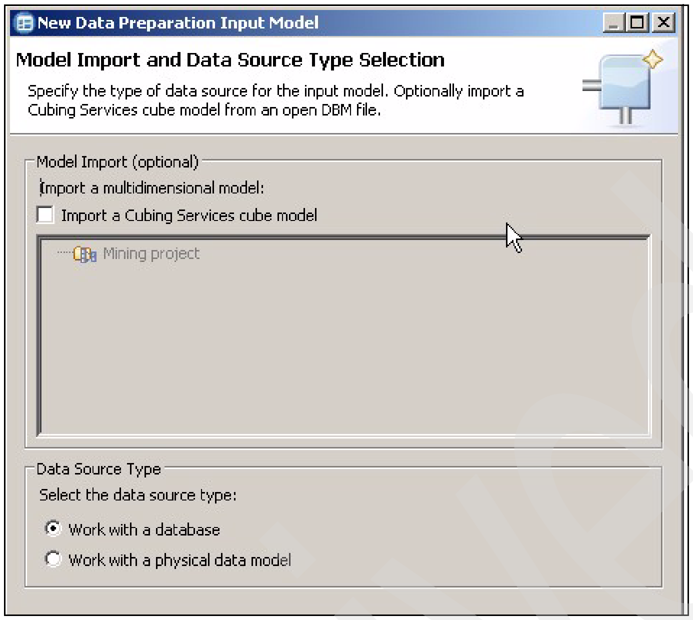


Figure 7-11 Launching the Data Preparation Wizard from the Data Project Explorer

When the Data Preparation Input Model launches, the user can select either of the following options:

- To work with an existing database connection, as illustrated in Figure 7-12 on page 269.
- To import a Cubing Services cube model, as illustrated in Figure 7-13 on page 269.


New Data Preparation Input Model

Model Import and Data Source Type Selection

Specify the type of data source for the input model. Optionally import a Cubing Services cube model from an open DBM file.

Model Import (optional)

Import a multidimensional model:

[ ] Import a Cubing Services cube model

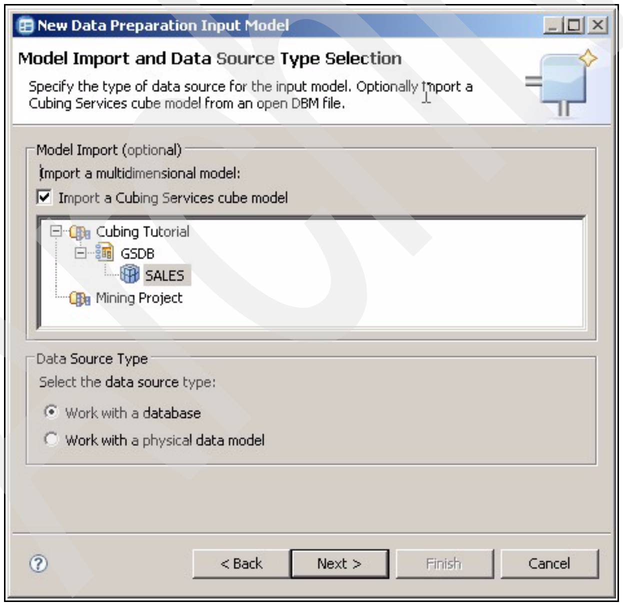
Data Source Type

Select the data source type:

(●) Work with a database

( ) Work with a physical data model

Figure 7-12 Model Import and Data Source Type Selection window


New Data Preparation Input Model

Model Import and Data Source Type Selection

Specify the type of data source for the input model. Optionally import a Cubing Services cube model from an open DBM file.

Model Import (optional)

Import a multidimensional model:

[✓] Import a Cubing Services cube model

Data Source Type

Select the data source type:

(●) Work with a database

( ) Work with a physical data model

Figure 7-13 Input Model Cube Selection

If the **Work with a database** radio button is selected, an input model editor window opens, as shown in Figure 7-14. Tables can be dragged onto the canvas, and joins and hierarchies defined to complete the model.

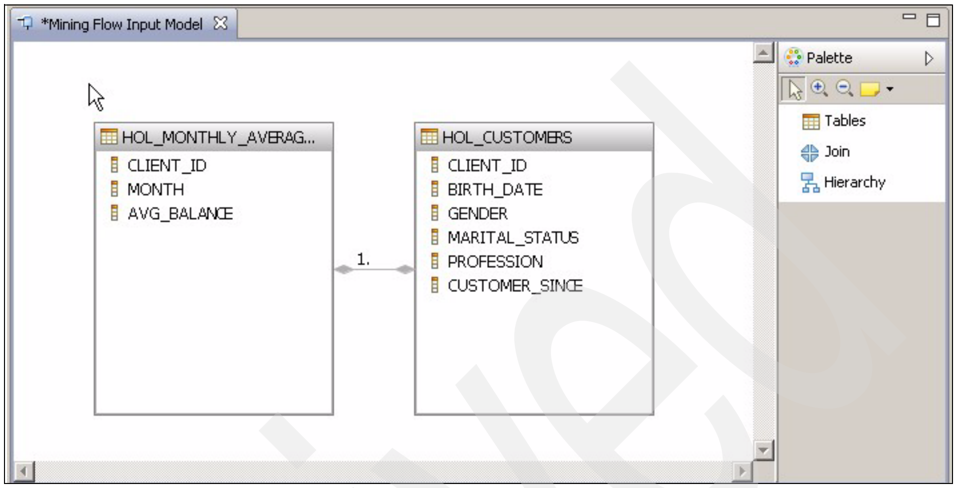


Figure 7-14 Input Model example

If the **Import a Cubing Services cube model** check box is selected, a cube model diagram opens in the editor tab, as shown in Figure 7-15.

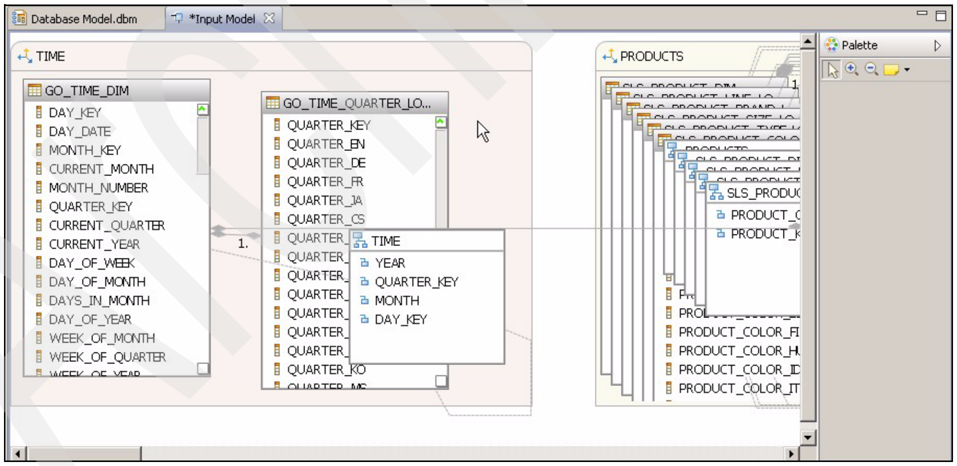
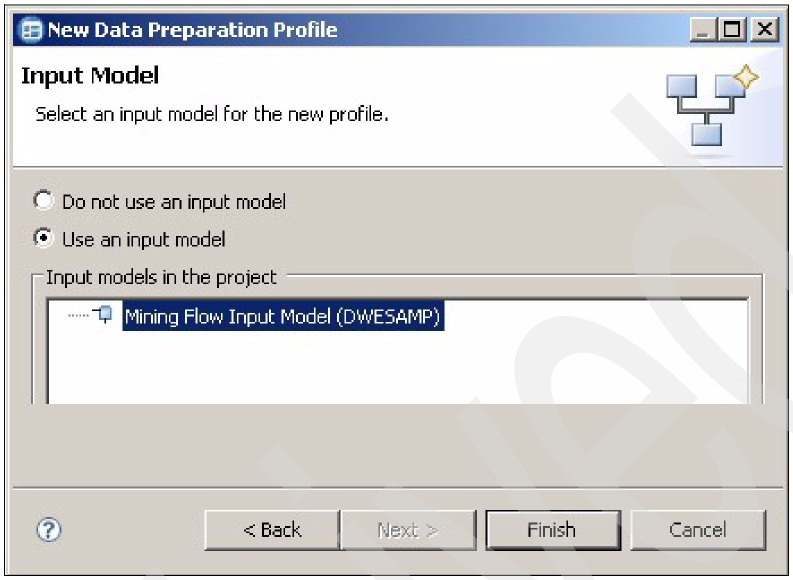


Figure 7-15 Input model example: Cube model

When the Data Preparation Profile launches, the user can select which input model to use, as illustrated in Figure 7-16.


New Data Preparation Profile

Input Model

Select an input model for the new profile.

( ) Do not use an input model

(●) Use an input model

Input models in the project

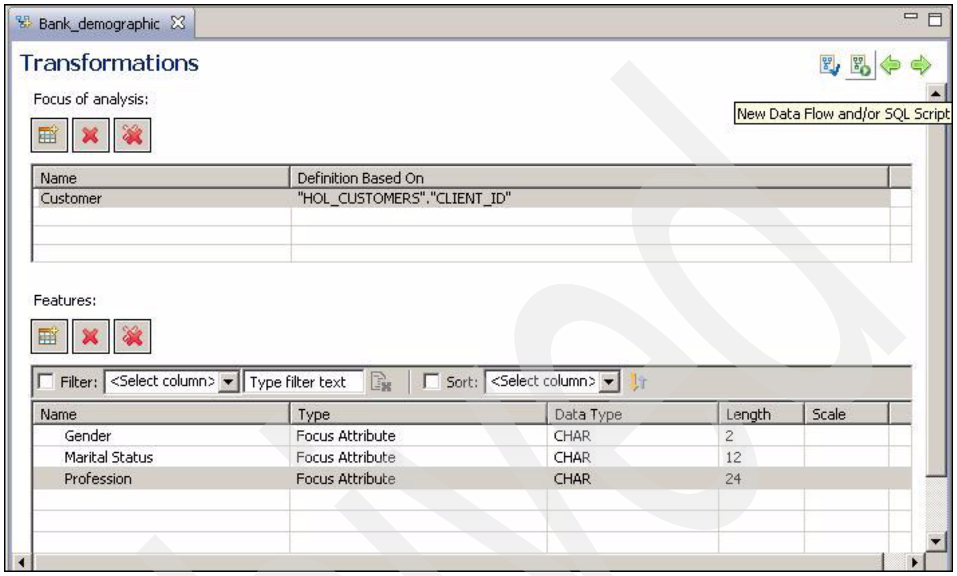
Mining Flow Input Model (DWESAMP)

Figure 7-16 New Data Preparation Profile

The Data Preparation Wizard opens on the Overview tab, which explains input models, transformations, and the results table. The Transformation tab allows the user to specify a focus of analysis, and the features are to be included as part of the analysis.

The Data Preparation Wizard can also be used to calculate proportions, so that the data is represented as a percentage rather than an absolute value. For example, the percentage of bank accounts by type, calculated as a percentage of the total number of bank accounts.

The Data Preparation Wizard takes these inputs, and create a data flow or SQL script that will populate the result table ready for mining. The process described is illustrated in Figure 7-17.



Name	Definition Based On
Customer	"HOL_CUSTOMERS"."CLIENT_ID"

Name	Type	Data Type	Length	Scale
Gender	Focus Attribute	CHAR	2	
Marital Status	Focus Attribute	CHAR	12	
Profession	Focus Attribute	CHAR	24	

Figure 7-17 Data Preparation Wizards - transformations

7.5.4 Mining Flows

A mining flow designs and execute a mining process. Every mining flow includes at least one input table and a mining operator specific to the mining technique being used. SQL functions such as table joins, column selections, sampling, filters, and other data preparation functions are often included in a mining flow. Most mining flows have one or more output targets such as a visualizer operator or an output table containing scored results. Scoring might be done in the same flow as the modeling or in a separate flow using a mining model as an input.

Unlike data flows, mining flows are developed interactively with a live connection to a database. This means that individual branches of the mining flow can be executed and validated separately as the mining flow is being built.

Although Design Studio allows the user to connect to more than one database, a particular mining flow works with only one database. This is because DB2-based data mining uses stored procedures and user-defined functions that operate only in the database where the mined data reside. To allow a table residing in a different database to be used in a mining flow, an alias referring to that table and database would have to be created in the database used by the mining flow.

7.5.5 Data mining operators

Mining flows use many of the same operators used in SQW flows, which are discussed in Chapter 5, “Data movement and transformation” on page 75. Mining flows also use several operators that are specific to data mining. Operators are selected from the palette and dropped onto the mining flow canvas. Additional information about operators is available under the Help tab in the Design Studio menu.

Advanced transformation operators

In addition to other data transformation operators, mining flows use four transformation operators specific to data mining (Figure 7-18):

- Sampler
- Random split
- Discretize
- Item aggregator

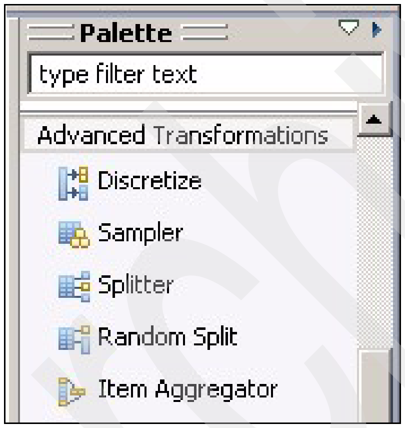


Figure 7-18 Mining specific transformation operators

Sampler operator

The sampler operator is a filter that creates a random sample of an incoming table represented by another operator. For large input tables, processing times to build mining models might be long. One solution to reduce processing time is to create a mining model using a random sample of the input data during the iterative development phase (when different parameters and settings are tested to generate the best model) and run the final model with the entire data set.

When sampling a large table source, however, the random sample can be created more efficiently by specifying a sampling rate on the Table Source operator, which exploits DB2's native sampling with good performance. An example of specifying the sampling rate (entered as a percentage) in the Table Source properties is illustrated in Figure 7-19.

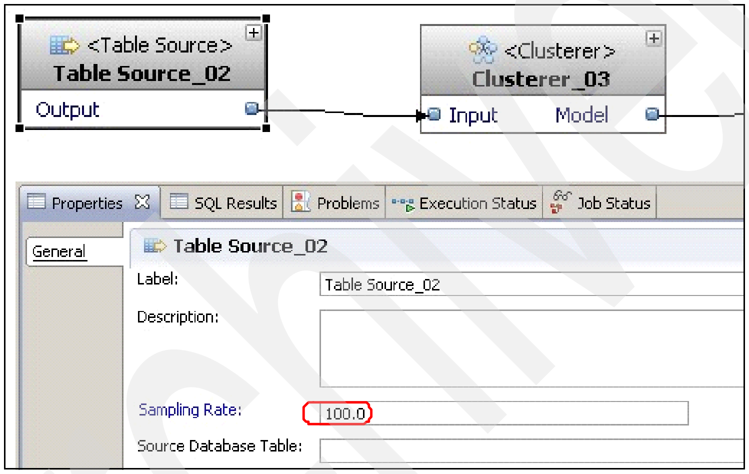


Table Source_02

Label: Table Source_02

Description:

Sampling Rate: 100.0

Source Database Table:

Figure 7-19 Example of specifying a sampling rate in a table source operator

Random split operator

The random split operator is used to split source data into training and testing data sets for building and validating predictive models, respectively. Unlike the splitter operator, which splits the rows of a table into multiple tables based on user-specified conditions, the random split operator splits the input data randomly to ensure that the training and testing sets properly represent the input data.

As shown in Figure 7-20, the random split operator is positioned between the input data and the model building and validation operators. The random split operator randomly selects a testing data set and creates a training data set, which is the complement of the testing data unless stratified sampling is performed. The training data becomes the input data for building the data model, and the testing data becomes the input data for testing or validating the model.

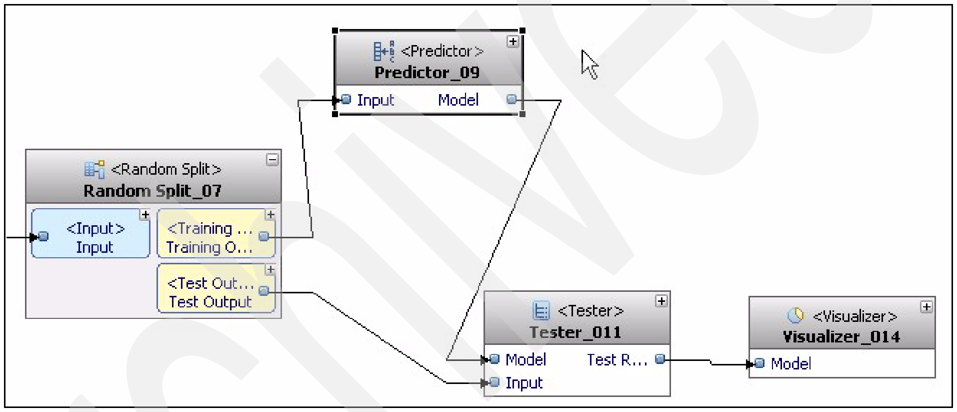
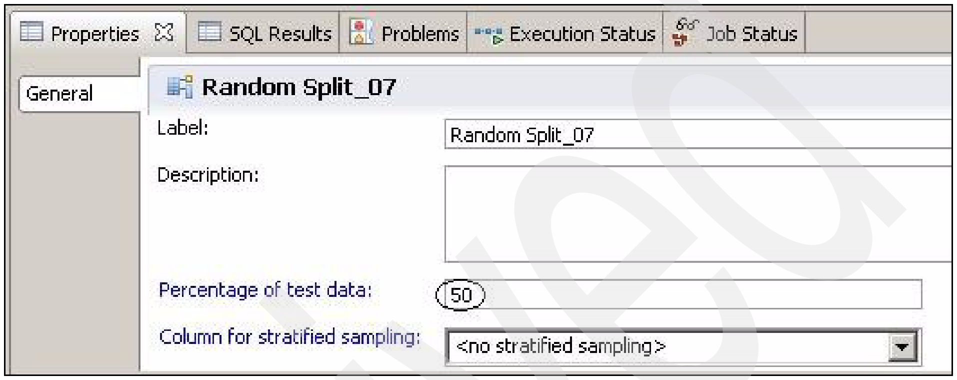


Figure 7-20 Training and testing sets created by the random split operator

The properties of the random split operator are illustrated in Figure 7-21 on page 276. The analyst specifies the percentage of the source data to be randomly selected for the testing data set. Typically, stratified sampling is not done, but it can be useful to an experienced analyst who wants to create an enriched sample for building a classification model when the response or target value of interest occurs for a small percentage of the records.

For example, when building a model to predict adverse patient reaction to a medical treatment protocol, the target value NO might occur with a frequency of 99% and the target value YES with a frequency of only 1%. A highly accurate classification model would be to predict that a patient will never have an adverse reaction (NO).

Although it would be 99% accurate, such a model would be trivial because it would tell us nothing about those who will react adversely (YES). Using stratified sampling to construct an enriched training data set having approximately equal percentages of YES and NO records often yields a better model, which is validated using a testing data set with the actual percentages of YES and NO. If stratified sampling is not specified, the training data set consists of the records of the complement of the testing data set.



Random Split_07

Label: Random Split_07

Description:

Percentage of test data: 50

Column for stratified sampling: <no stratified sampling>

Figure 7-21 Example of random split operator properties

Discretize operator

The discretize operator converts the values of a continuous input field (variable) into a defined set of categorical or numeric buckets. For example, the variable INCOME ranges from $0 to $10 million with records having missing (null) values. The discretize operator can be used to transform the numeric values into categories of LOW, MEDIUM, HIGH, and UNKNOWN. Although discretizing a variable reduces the amount of information contained in the variable, it might be suitable, depending on the objectives of the analysis or issues such as poor data quality (inaccurate or missing values in many of the records). As illustrated in Figure 7-22, the discretize operator is positioned after the input data setup.

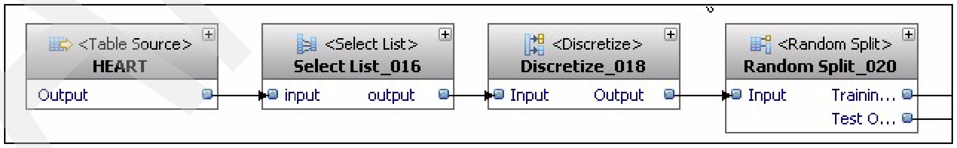
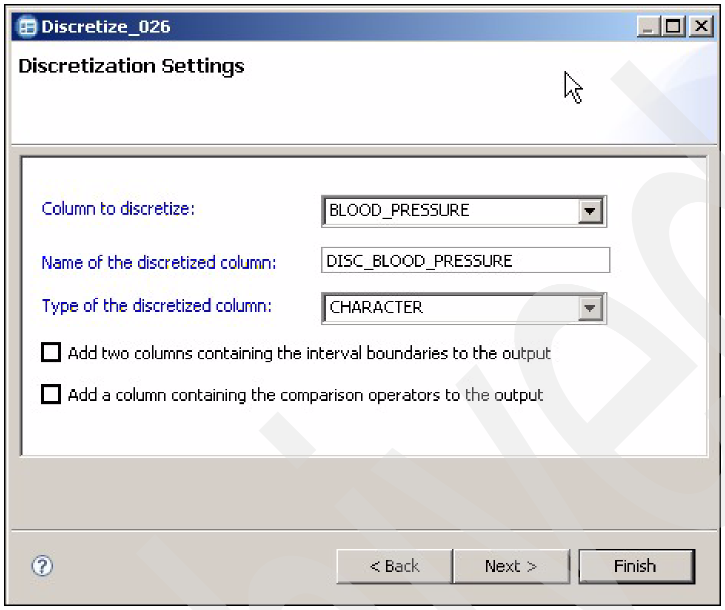


Figure 7-22 Example of the discretize operator in a mining flow

The properties of the discretize operator are illustrated in Figure 7-23. The analyst selects the variable (column) to be discretized and enters the name and type of the new discretized column to be created.



Column to discretize: BLOOD_PRESSURE

Name of the discretized column: DISC_BLOOD_PRESSURE

Type of the discretized column: CHARACTER

☐ Add two columns containing the interval boundaries to the output

☐ Add a column containing the comparison operators to the output

Figure 7-23 Example of discretize operator properties

As illustrated in Figure 7-24 on page 278, any of three methods can be selected to create the discretize interval boundaries for the discretized variable. These methods are:

- Define the intervals manually.

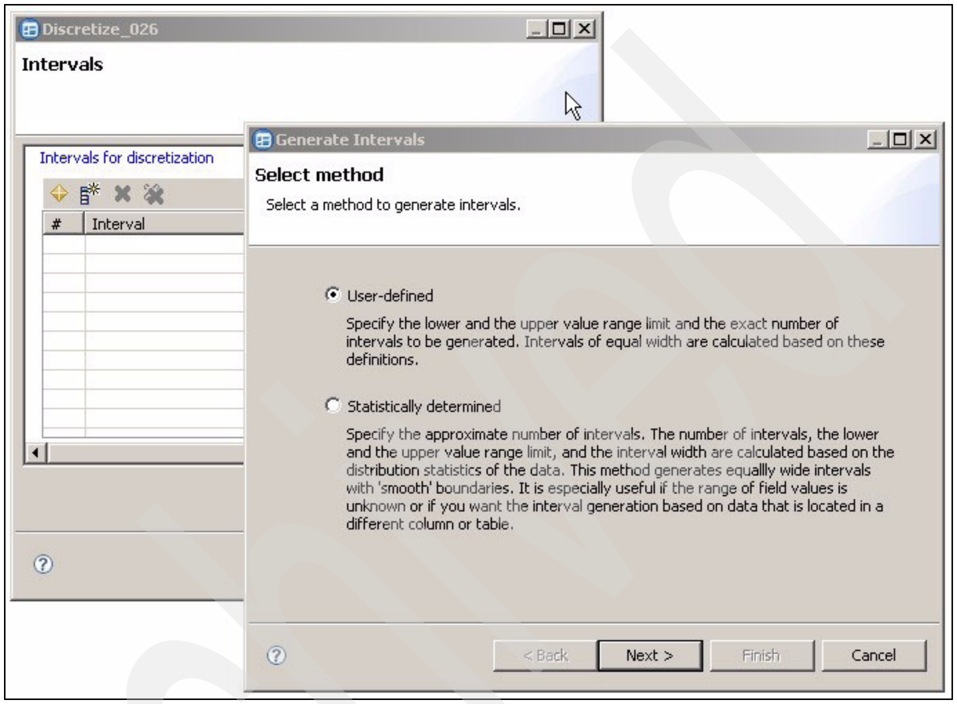
  The analyst creates each interval separately by specifying the lower limit, upper limit, and the new discretized variable for each interval.

- Calculate the intervals from specified values.

  The analyst specifies the desired number of intervals, the lower and upper limits of the overall value range, and a prefix used in generating the new discretized values. The system then calculates intervals of equal width plus one interval for all values below the lower limit and one interval for all values above the upper limit. For each interval, the new discretized value is generated by appending a consecutive number to the prefix.

- Calculate he intervals automatically based on statistical analysis.

  The analyst specifies the approximate number of intervals to be generated and the prefix for the new discretized values. Based on the distribution statistics, the system generates the exact number of intervals, the lower and upper value range limits, and the interval width, creating intervals with smooth boundaries and equal widths. This method is especially useful if the range of field values is unknown or if the interval generation is to be based on data located in a different column or table.



Select method

Select a method to generate intervals.

(●) User-defined

Specify the lower and the upper value range limit and the exact number of intervals to be generated. Intervals of equal width are calculated based on these definitions.

( ) Statistically determined

Specify the approximate number of intervals. The number of intervals, the lower and the upper value range limit, and the interval width are calculated based on the distribution statistics of the data. This method generates equallly wide intervals with 'smooth' boundaries. It is especially useful if the range of field values is unknown or if you want the interval generation based on data that is located in a different column or table.

Figure 7-24 Example of selecting interval method for discretize operator

Item aggregator operator

The item aggregator operator creates set-valued input-columns from multiple rows. For example, a call center might have several notes relating to one particular customer. Therefore, a problem extractor might extract multiple problems per customer. If you built a prediction model that predicts the likelihood of customer regression based on customer demographics and the problems that are extracted from unstructured text analysis, a new input field is added to the prediction model. This new input field contains multiple values. These values are called set-valued fields. The item aggregator operator can create the required XML format for set-valued fields from an input table that contains one row for each value.

The item aggregator operator includes an input port that contains at least two of the following input columns:

- A set of key columns that is used to identify the values that belong to one set
- A value column that contains the values for the set
- (Optional) A confidence column

The operator includes an output port that contains the selected key columns and a column of type VARCHAR or XML. The column of type VARCHAR or XML contains the set-valued XML string.

The example flow using the containing the item aggregator operator is shown in Figure 7-25. The item operator is located between the table source and the table target.

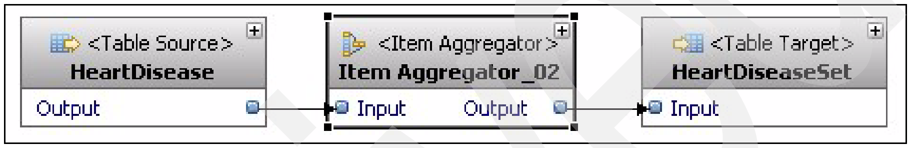
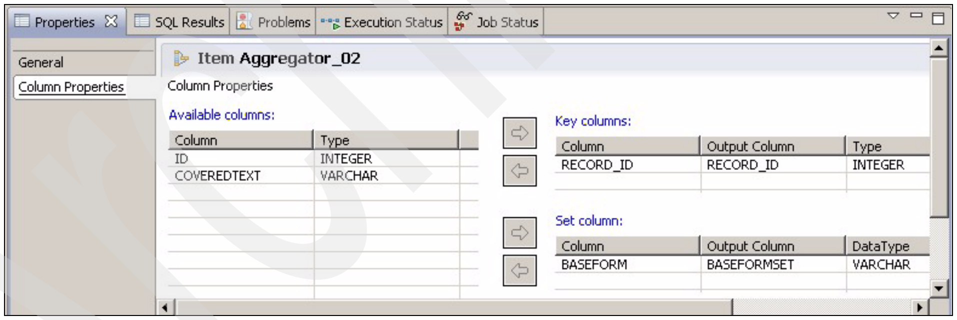


Figure 7-25 Item aggregator flow

The properties view for the item aggregator operator is illustrated in Figure 7-26. The key column and the set column are defined on the Column Properties tab.



Item Aggregator_02

Column Properties

Available columns:

Column	Type
ID	INTEGER
COVEREDTEXT	VARCHAR

Key columns:

Column	Output Column	Type
RECORD_ID	RECORD_ID	INTEGER

Set column:

Column	Output Column	DataType
BASEFORM	BASEFORMSET	VARCHAR

Figure 7-26 Item aggregator properties

Mining operators

In addition to the table source and table target operators, mining flows use several specific mining operators. These operators perform functions for model visualization, creation, validation, scoring, and results extraction.

Visualizer operator

The visualizer operator displays a data mining model using the appropriate visualizer. Each visualizer provides graphical and tabular information about the model results and model quality, depending on the mining method.

Model source operator

The model source operator represents a data mining model stored in the database. With this operator, an existing model can be placed in a mining flow and used for scoring. This operator is useful when incorporating a third-party PMML model, such as one generated by a SAS tool, into a mining flow.

Find deviations operator

The find deviations operator finds records in the input table that deviate from the majority of records as determined by combinations of characteristics that exist only for few records. The procedure is based on clustering and uses data in demographic format. Examples of useful deviations that can be discovered in the data include clients who deserve special attention because they buy expensive products of high quality, possibly fraudulent behavior indicated by clients who consistently have high discounts or high numbers of accident claims, or data quality problems indicated by unusual combinations of characteristics. This is a high-level operator that needs only minimal parameter specifications and does not use a visualizer operator. It creates a clustering model that can be viewed by opening it in the Data Source Explorer, as shown in Figure 7-27.

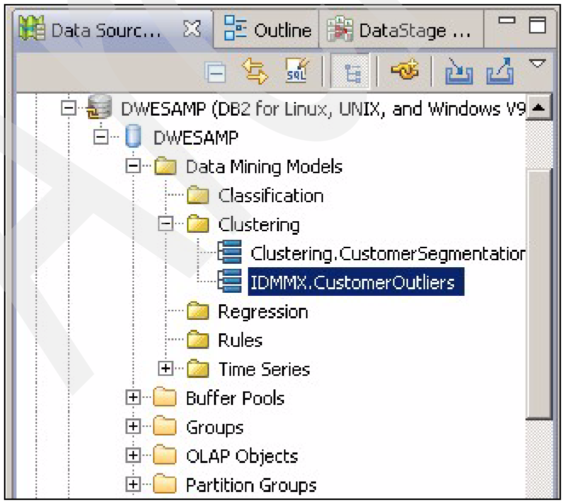


Figure 7-27 Opening a data mining model from the Data Source Explorer

Find rules operator

The find rules operator finds relationships in the input table and creates a set of rules describing those relationships. The operator accepts data in either the transactional or demographic format. The results consist of a model and a set of associations rules describing item affinities or relationships among columns. This is a high-level operator that uses minimal parameter specifications and does not use a visualizer operator. The rule model can be viewed by opening it in the Data Source Explorer.

Cluster table operator

The cluster table operator finds groups with similar characteristics in the input table, using clustering to group similar records. Using data in demographic format, the cluster table operator creates a clustering model and a view of the input records with their respective cluster assignments. This is a high-level operator that uses minimal parameter specifications and does not use a visualizer operator. The clustering model can be viewed by opening it in the Data Source Explorer.

Predict column

The predict column operator predicts the values of a target field by building a model from the columns in the input table. The operator uses data in demographic format. Depending on whether the target field is categorical or numeric, the resulting model type is either classification or regression, respectively. This is a high-level operator that uses minimal parameter specifications and does not use a visualizer operator. The classification or regression model can be viewed by opening it in the Data Source Explorer.

Associations operator

The associations operator builds a rule model containing association rules describing relationships among the items (which might be, for example, products, events, and characteristics) occurring together in a transaction. The operator uses an input table in transactional format with a transaction identifier column and an item column. The analyst has extensive control over parameters. The output port of the associations operator flows to the input port of a visualizer operator. The model visualization is displayed automatically upon completion of the mining run. The rule model also can be viewed by opening it in the Database Explorer or from the visualizer operator in the mining flow.

Sequence operator

The sequences operator builds a rule model containing sequence rules describing relationships among items (such as products and events) occurring together across sequential transactions. The operator uses an input table in transactional format with a sequence identifier column (for example, client) in addition to the transaction identifier and item identifier columns. The analyst has extensive control over parameters. The sequences operator pipes the model to a visualizer operator to display the model visualization automatically when the mining run has finished. The rule model can also be viewed in the Database Explorer or from the visualizer operator in the mining flow.

Clusterer operator

The clusterer operator builds a clustering model that contains groups of similar records. The operator uses an input table in demographic format and allows extensive control over parameters. The clustering model flows to a visualizer operator to display the model visualization automatically when the mining run has finished. The clustering model can also be opened in the Database Explorer or from the visualizer operator in the mining flow.

Predictor operator

The predictor operator creates a mining model that predicts the value of a target field (called the class variable in a classification model or the dependent variable in a regression model). Depending on whether the target field is categorical or numeric, the resulting model type is either classification or regression, respectively. The predictor uses data in demographic format and allows extensive control over parameters. As illustrated in Figure 7-28, a random split operator is used to generate training and testing data sets, passing the training data to the predictor operator to build the model.

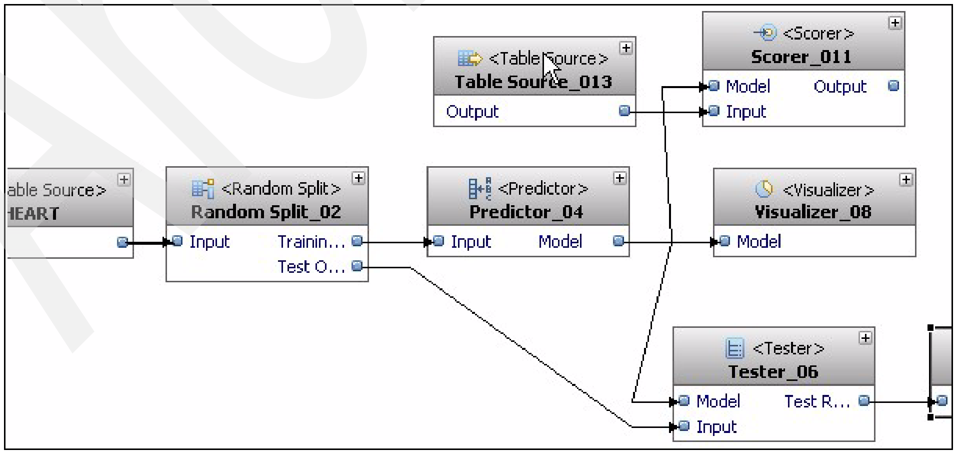


Figure 7-28 Example of predictor operator in mining flow

Time series operator

The time series operator forecasts values of a numerical data field for a future period in time. A data record can be any numerical value that is ordered in time. The order has to be specified with a time/ordering column. The order/time column can be of any numerical or date/time SQL type. The output of the operator is a time series model that contains the original data sets and the forecast predictions for each set. The model can be visualized using the visualizer operator.

Tester operator

The tester operator tests a classification or regression model by applying the model to the testing data set and computing test results containing information about model quality (prediction accuracy, ability to rank records correctly, and reliability on new data). As illustrated in Figure 7-29, the tester operator applies the trained model created by the predictor and applies it to the testing data flowing from the random split operator.

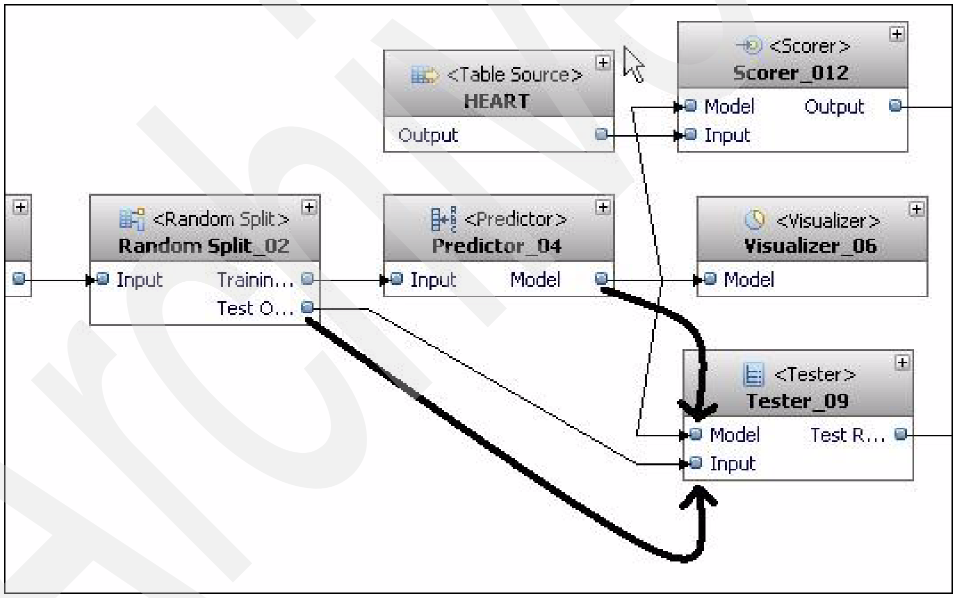


Figure 7-29 Example of tester operator in a mining flow

Two visualizer operators are used to display, respectively, the trained model from the predictor and the testing results from the tester. When the mining flow is executed, visualizations for both the training and the tested are generated.

Scorer operator

The scorer operator applies a mining model to an input table to compute scores for new records. The scorer operator reads a data mining model from a DB2 table and applies it to the new input data. The scorer is not algorithm-specific, but works with many types of data mining models, as described in 7.4.3, “Scoring” on page 256. Depending on the model type, the scorer creates different scoring results in the output table. The scorer operator is positioned in a mining flow, as illustrated in Figure 7-30. The scorer receives the mining model from the predictor and applies it to the table source representing the application data to be scored. The scoring results are then piped to a new table target operator.

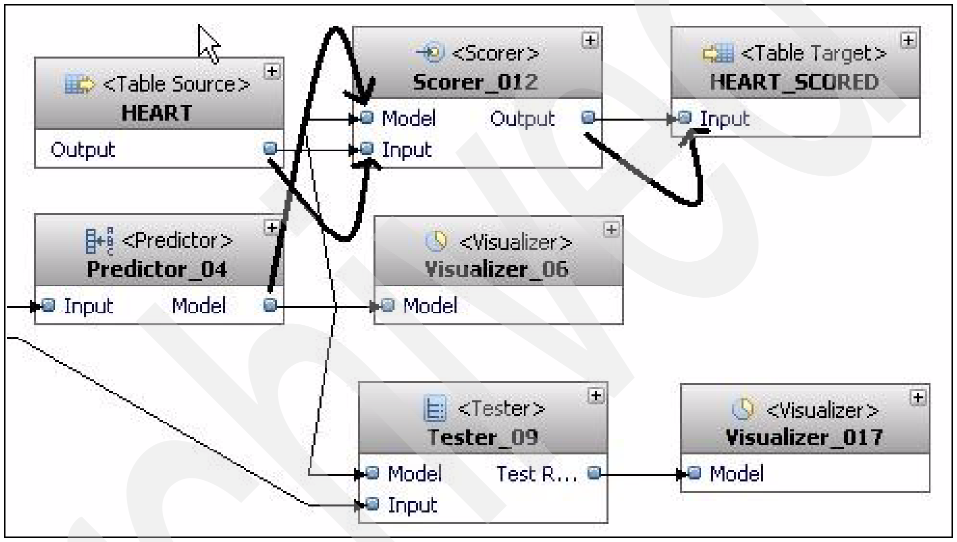


Figure 7-30 Example of scorer operator in a mining flow

Associations extractor operator

The associations extractor operator extracts information from an association rules model and stores the extracted rules and related information in an output table using a table target operator. Associations rules stored in relational tables can, for example, be accessed by a reporting tool to generate reports. The associations extractor operator is illustrated in Figure 7-31.

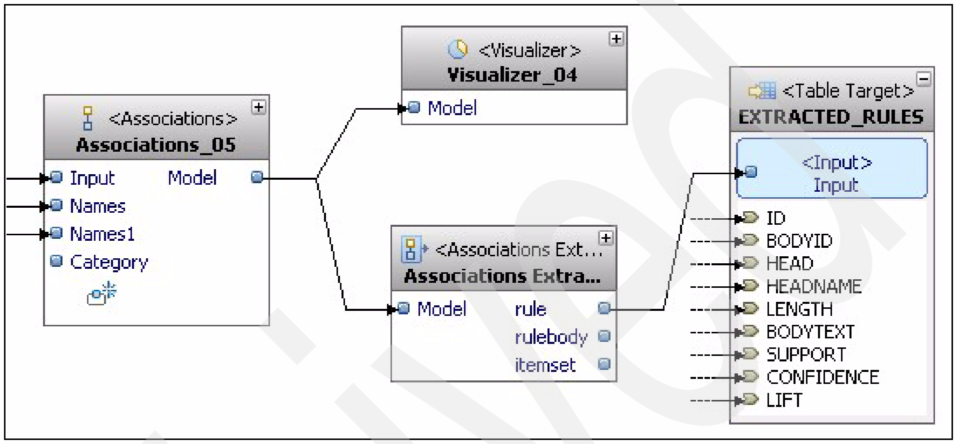


Figure 7-31 Example of association extractor operator in a mining flow

Sequences extractor operator

The sequences extractor operator extracts information from a sequences rules model and stores the extracted rules and related information in an output table. Each row in the output table represents one sequence and contains information about the calculated rules, details about each sequence, and the item sets used in the rules. The sequences extractor is positioned in a mining flow similarly to the extractor illustrated in Figure 7-31, although the information contained in the output table is specific to the mining model type.

Cluster extractor operator

The cluster extractor operator extracts information from a clustering model and stores the extracted information in an output table. Each row in the output table represents one cluster and contains information about each cluster's size, homogeneity of the records in the cluster, and a description of the attributes of the cluster. The cluster extractor is positioned in a mining flow similarly to the extractor illustrated in Figure 7-31.

Time series extractor operator

The time series extractor operator extracts information from a time series model and stores the extracted information in an output table. The output table contains the original data sets, the forecasted results for each set, and the type of algorithm used for each specific forecast. The time series extractor is positioned in a mining flow similar to the extractor illustrated in Figure 7-31 on page 285.

Correlations extractor operator

The correlations extractor operator extracts correlations (Pearson's correlation coefficient, cf. listed in "Related publications" on page 597) from a clustering, classification, or regression model and stores them in an output table. Correlation coefficients are calculated for each pair of columns (variables) in the input data used to train the model.

Fields extractor operator

The fields extractor operator extracts field information from a mining model and stores the information in an output table. The extracted information includes the set of input data fields used in the model's internal equations and the importance (relative contribution) of each field in the model.

Tree rules extractor operator

The tree rules extractor operator extracts rules (decision paths) from a decision tree classification model and stores the information in an output table. The extracted information includes the node identifier, scored value and confidence, size, depth, and decision path for each node.

Gains extractor operator

The gains extractor operator extracts a table containing gains chart information from a classification or regression model. If the model is a decision tree, the desired value of the class variable is specified in the gains extractor operator. A gains chart is useful in assessing the quality of a model or in comparing the relative performance of multiple models in predicting the class variable or dependent variable.

Quality extractor operator

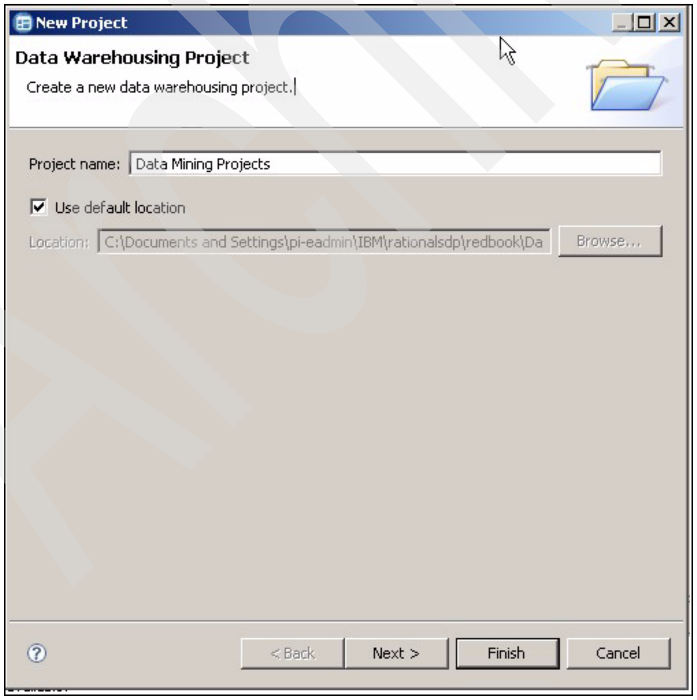
The quality extractor operator extracts model quality information from a data mining model or a test result and stores the result in an output table. The extracted information is specific to the type of data mining model.

7.5.6 Building a mining flow

In this section we demonstrate how a typical mining flow is built. The assumptions are that one or more data tables have already been created and prepared by a data flow, and that these tables are in the correct format (transactional or demographic) required for the mining.

In the example used to illustrate the creation of a mining flow we analyze patient data from a hospital. The hospital's heart department has master-records of their patients together with several measures like heart rate, blood pressure, cholesterol, and so on. Patients are checked for four different heart diseases. The patient records contain a column that indicates whether they have one of the four heart diseases or not. The analytics goal is to predict for new patients the risk that they suffer one of the four heart diseases. If the risk is high, immediate check-ups should be done. The idea is to enable risk management even if no dedicated check for one of those heart diseases was made but the measures are available anyway from earlier check-ups in other areas.

The first step in building a mining flow is to create a new data warehousing project in the Data Project Explorer by selecting **File** → **New** → **Data Warehouse Project** from the menu and stepping through the wizard, as illustrated in Figure 7-32. This new project contains all of the mining flows, data flows, control flows, and other objects created under it.


New Project

Data Warehousing Project

Create a new data warehousing project.

Project name: Data Mining Projects

☑ Use default location

Location: C:\Documents and Settings\pi-eadmin\IBM\rationalsdp\redbook\Da

Figure 7-32 Creating a new data warehouse project for mining flows

The next step is to create a new mining flow in the new project, as shown in Figure 7-33. The new mining flow is given a suitable name, and the appropriate database connection is specified. Unlike data flows, mining flows are created and executed with a live connection to a database. Each mining flow can have only one database connection. If a table in a second database must be used, than an alias pointing to the desired table in the second database can be created in the first database.

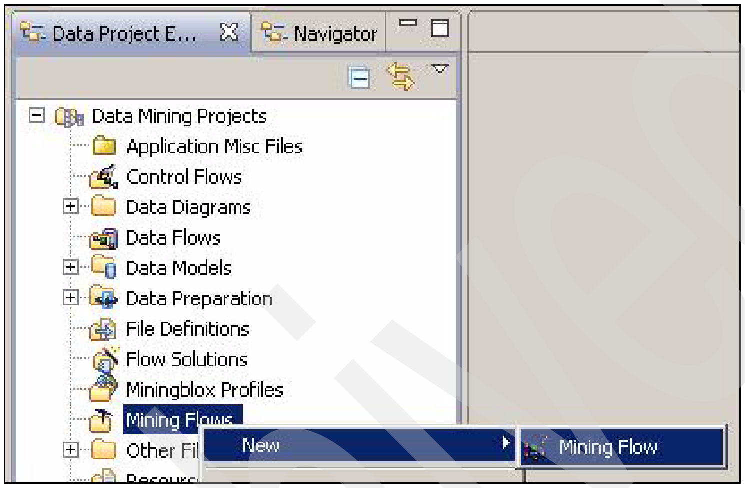


Figure 7-33 Creating a new mining flow

All of the mining flows created under the project are listed under the Mining Flows folder, where each mining flow can be opened by double-clicking it, as shown in Figure 7-34. The mining flow canvas appears in the upper right quadrant of Design Studio. This canvas is initially blank and ready for the mining flow itself to be created.

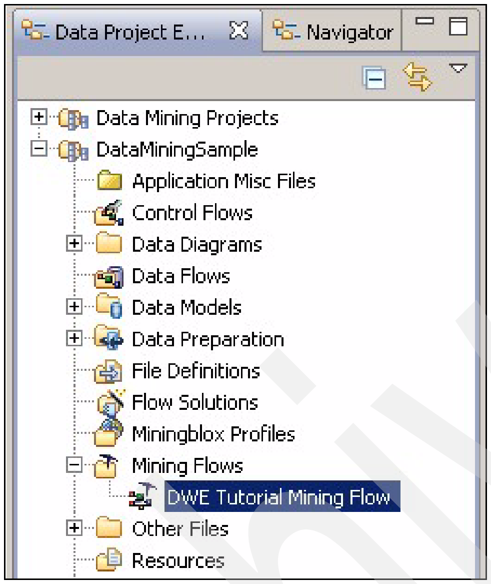


Figure 7-34 Opening a mining flow

Now the construction of the mining flow itself begins. A typical mining flow is depicted in Figure 7-35. Although only a few of the many available operators are used, this example illustrates the basics of input data, data preparation, model creation, visualization, and model application. In the rest of this section we discuss each operator in the example mining flow in Figure 7-35. Although every mining flow varies according to the model type and other requirements driven by the business problem, this example highlights the most commonly used mining operators and shows how they are interrelated in the flow.

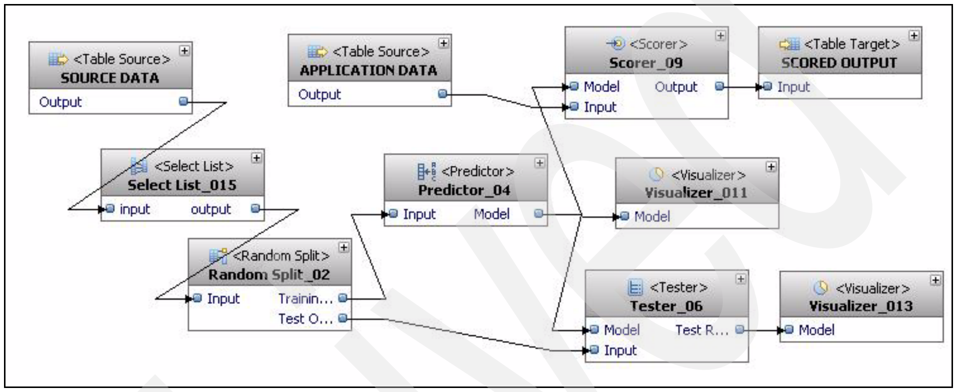
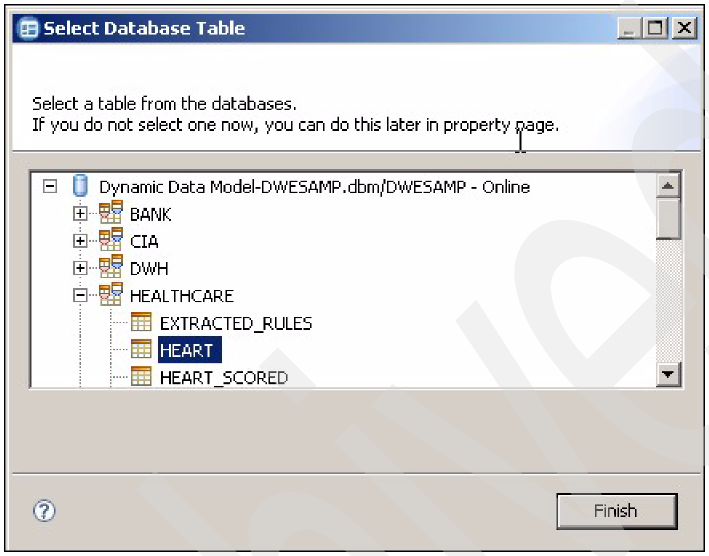


Figure 7-35 Typical mining flow in Design Studio

Referring to Figure 7-35 on page 290, we begin by creating a table source to represent each input data table. In this example, only one table is used as input, but mining flows often use multiple input tables. Here, the table source operator is labeled SOURCE DATA. This operator represents the table selected, as shown in Figure 7-36.

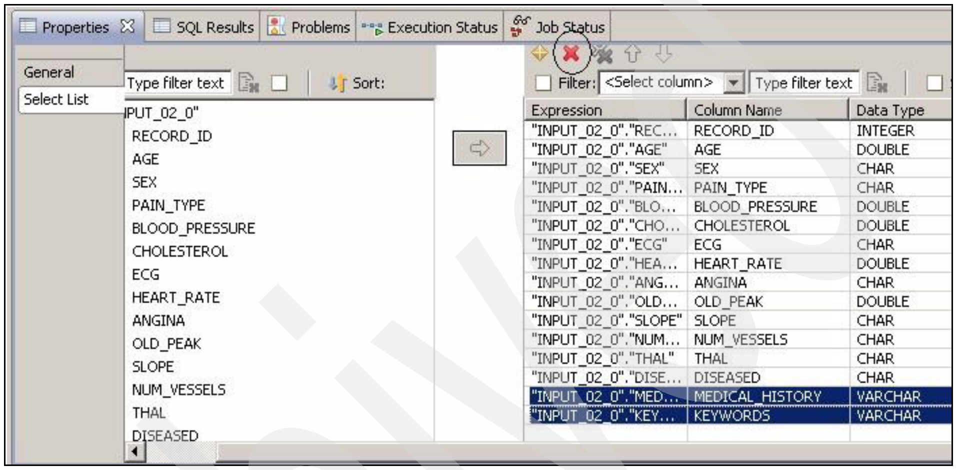


Select a table from the databases.
If you do not select one now, you can do this later in property page.

Dynamic Data Model-DWESAMP.dbm/DWESAMP - Online
- BANK
- CIA
- DWH
- HEALTHCARE
  - EXTRACTED_RULES
  - HEART
  - HEART_SCORED

Figure 7-36 Selecting the input table for a new table source

Because the input data table has already been prepared in demographic format, no further preprocessing is needed. Because the columns in the input table are not included as variables in the data mining model, however, a select list operator is used to select the desired subset of columns. In this case, we remove two columns (medical history and keywords) because they are text fields and not to be used in this model. As shown in Figure 7-37, unwanted columns are selected and removed by clicking the red X (highlighted) in the select list operator.

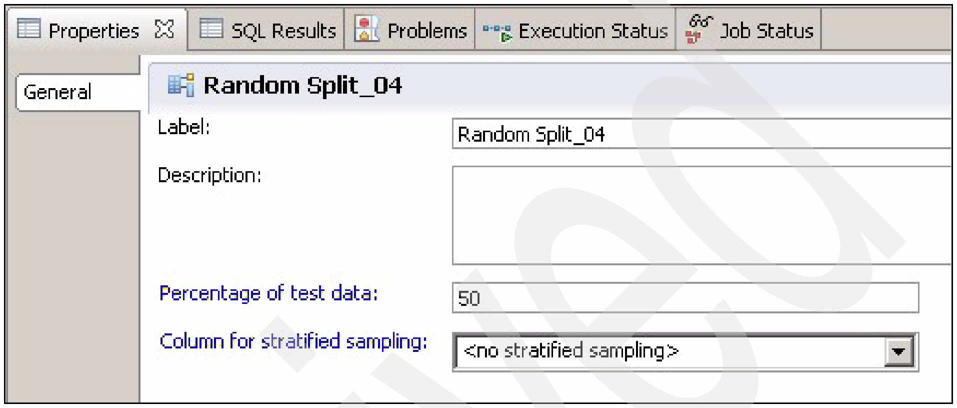


```
IPUT_02_0"
  RECORD_ID
  AGE
  SEX
  PAIN_TYPE
  BLOOD_PRESSURE
  CHOLESTEROL
  ECG
  HEART_RATE
  ANGINA
  OLD_PEAK
  SLOPE
  NUM_VESSELS
  THAL
  DISEASED
```

Expression	Column Name	Data Type
"INPUT_02_0"."REC...	RECORD_ID	INTEGER
"INPUT_02_0"."AGE"	AGE	DOUBLE
"INPUT_02_0"."SEX"	SEX	CHAR
"INPUT_02_0"."PAIN...	PAIN_TYPE	CHAR
"INPUT_02_0"."BLO...	BLOOD_PRESSURE	DOUBLE
"INPUT_02_0"."CHO...	CHOLESTEROL	DOUBLE
"INPUT_02_0"."ECG"	ECG	CHAR
"INPUT_02_0"."HEA...	HEART_RATE	DOUBLE
"INPUT_02_0"."ANG...	ANGINA	CHAR
"INPUT_02_0"."OLD...	OLD_PEAK	DOUBLE
"INPUT_02_0"."SLOPE"	SLOPE	CHAR
"INPUT_02_0"."NUM...	NUM_VESSELS	CHAR
"INPUT_02_0"."THAL"	THAL	CHAR
"INPUT_02_0"."DISE...	DISEASED	CHAR
"INPUT_02_0"."MED...	MEDICAL_HISTORY	VARCHAR
"INPUT_02_0"."KEY...	KEYWORDS	VARCHAR

Figure 7-37 Selecting variables for the model in the Select List operator

Because we are creating a predictive model, the virtual input table resulting from the select list operator must be randomly split into two virtual tables for training and testing the model. In Figure 7-35 on page 290, the select list operator flows to a random split operator. In the random split operator, we accept the default parameters of a 50% testing set size (with the complement becoming the training set) and no stratified sampling, as shown in Figure 7-38.



Random Split_04

Label: Random Split_04

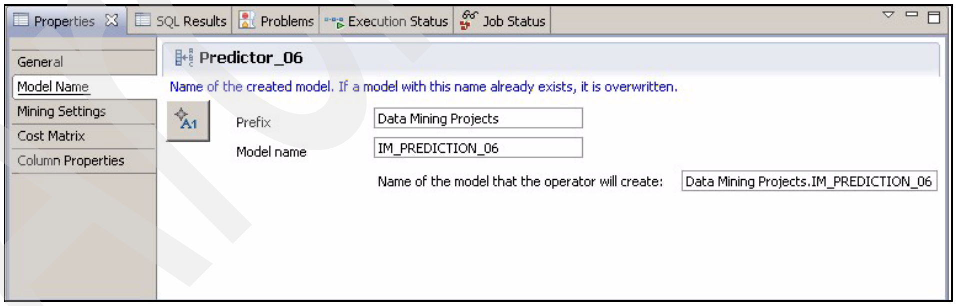
Description:

Percentage of test data: 50

Column for stratified sampling: <no stratified sampling>

Figure 7-38 Parameter specifications for the random split operator

Next we set up the mining operator to train the model. Because the model is to be a classification model, we use the predictor operator labeled Predictor_04 in Figure 7-35 on page 290. The training set from the random split operator goes to the predictor operator. The predictor operator is configured with a two-part name of the mining model to be created, as depicted in Figure 7-39.



Predictor_06

Name of the created model. If a model with this name already exists, it is overwritten.

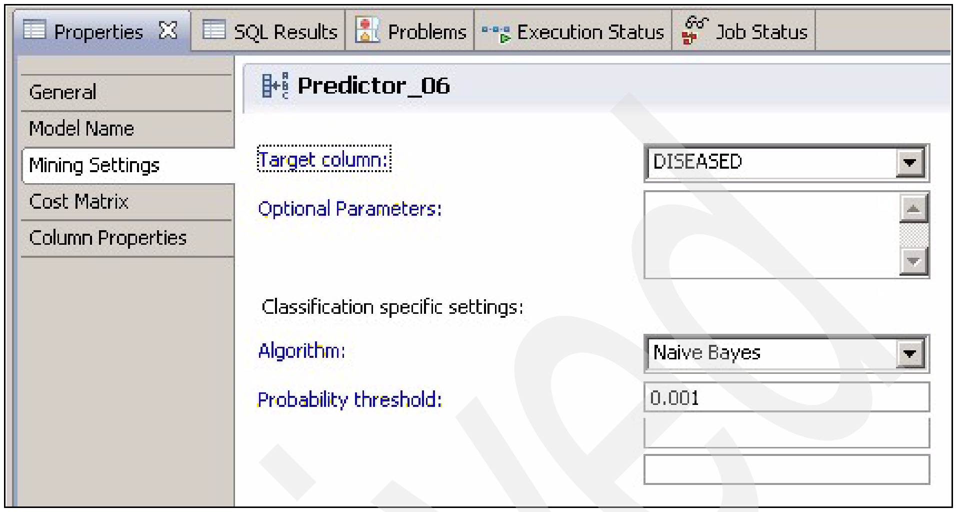
Prefix: Data Mining Projects

Model name: IM_PREDICTION_06

Name of the model that the operator will create: Data Mining Projects.IM_PREDICTION_06

Figure 7-39 Specifying the model name in the predictor operator

The Mining Settings tab of the properties view contains several parameters, including the target field and the algorithm type, as shown in Figure 7-40.



Predictor_06

Target column: DISEASED

Optional Parameters:

Classification specific settings:

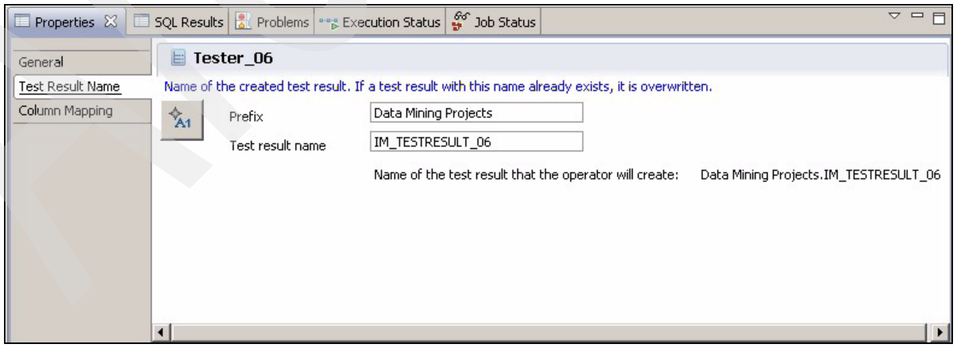
Algorithm: Naive Bayes

Probability threshold: 0.001

Figure 7-40 Specifying the model parameters in the predictor operator

The model output port of the predictor operator flows to a visualizer operator. After execution of the mining flow, the visualizer displays the trained for interpretation and quality assessment.

To generate testing results for model validation and quality assessment for application, the model flows from the predictor to a tester operator. The testing data flows from the random split operator to the tester, which applies the model to the testing data. The tester operator is configured with a two-part name of the testing result, as illustrated in Figure 7-41.



Tester_06

Name of the created test result. If a test result with this name already exists, it is overwritten.

Prefix: Data Mining Projects

Test result name: IM_TESTRESULT_06

Name of the test result that the operator will create: Data Mining Projects.IM_TESTRESULT_06

Figure 7-41 Specifying the test result name in the tester operator

The test result output port of the tester operator flows to another visualizer operator. After the execution of the mining flow, the visualizer displays the testing result for model quality assessment including the model's reliability in predicting the target field for new records.

The next part of the mining flow is to apply the model to score new records. Another table source operator, labeled APPLICATION DATA in Figure 7-35 on page 290, represents the application data. This table must include all of the columns used as variables in the model with the exception of the target field DISEASED. The records in this table represent new patients at the hospital. The objective is to predict each patient's likelihood of suffering from heart disease so that the appropriate steps can be taken.

To score new records, the model flows from the predictor to a scorer operator. The application data flows from the application data table source to the scorer, which applies the model to the application data and scores each record. An output table is needed to receive the scoring results, so a new table is created by right-clicking the output port of the scorer, as illustrated in Figure 7-42.

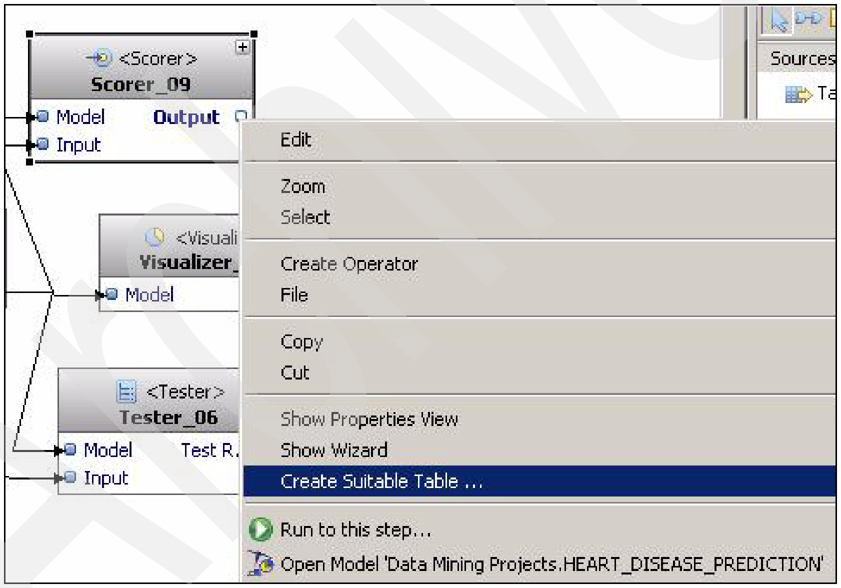


Figure 7-42 Creating a target table to receive the scoring results

The new table contains columns for the scoring results and the input columns used in the model, as shown in Figure 7-43. The scoring results for this model included the predicted value of the target field DISEASED and the confidence of the prediction.

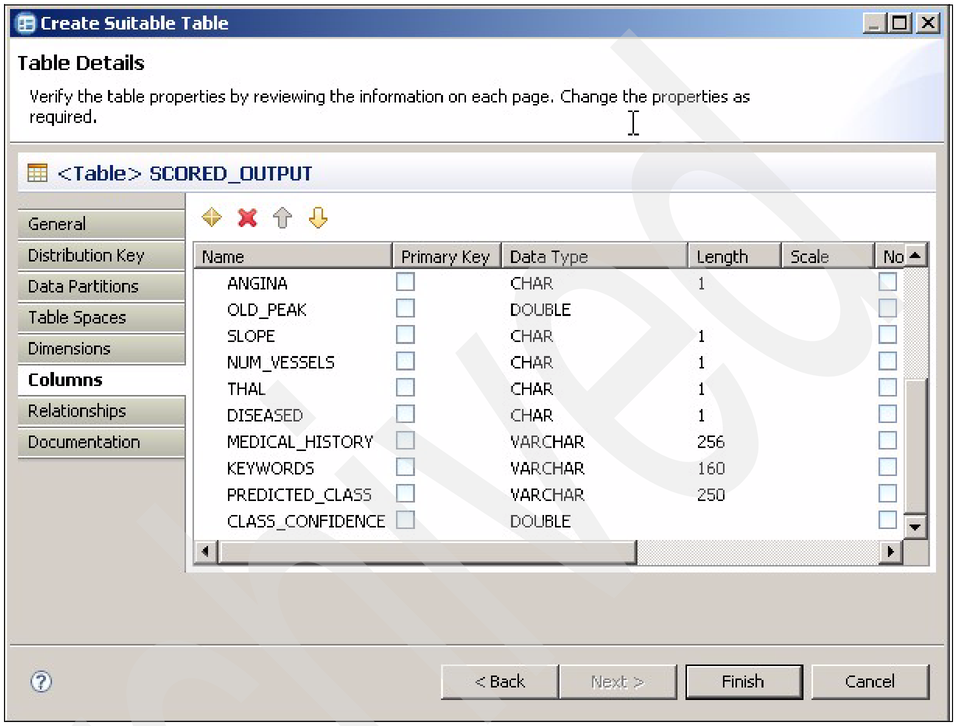


Table Details

Verify the table properties by reviewing the information on each page. Change the properties as required.

<Table> SCORED_OUTPUT

Name	Primary Key	Data Type	Length	Scale	No
ANGINA	☐	CHAR	1		☐
OLD_PEAK	☐	DOUBLE			☐
SLOPE	☐	CHAR	1		☐
NUM_VESSELS	☐	CHAR	1		☐
THAL	☐	CHAR	1		☐
DISEASED	☐	CHAR	1		☐
MEDICAL_HISTORY	☐	VARCHAR	256		☐
KEYWORDS	☐	VARCHAR	160		☐
PREDICTED_CLASS	☐	VARCHAR	250		☐
CLASS_CONFIDENCE	☐	DOUBLE			☐

Figure 7-43 Columns created in the scoring output table

A table target operator, labeled SCORED OUTPUT in Figure 7-35 on page 290, is created to represent the new output table. The table target operator is configured for the new table. The output port of the scorer is then connected to the table target operator to complete the mining flow.

Note: When creating a mining flow in Design Studio, it is possible to reuse operators in a flow by copying them and pasting them onto the canvas. The pasted operator contains the same properties that were set for the original operator.

7.5.7 Using the Solution Plan Wizard

The Solution Plan wizard can be used to create mining flows for a particular business problem or scenario. Using a solution plan can be faster and easier than using the flow editor. The Solution Plan Wizard can be launched from the Data Project Explorer, as shown in Figure 7-44.

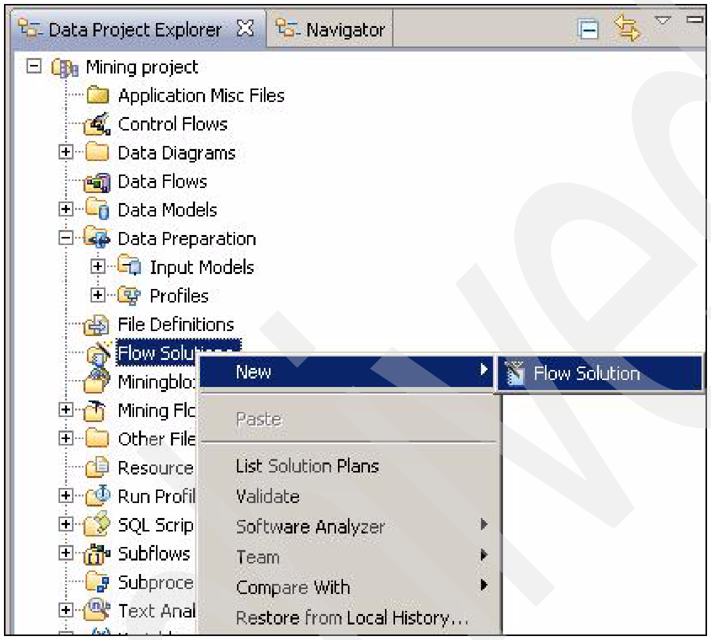
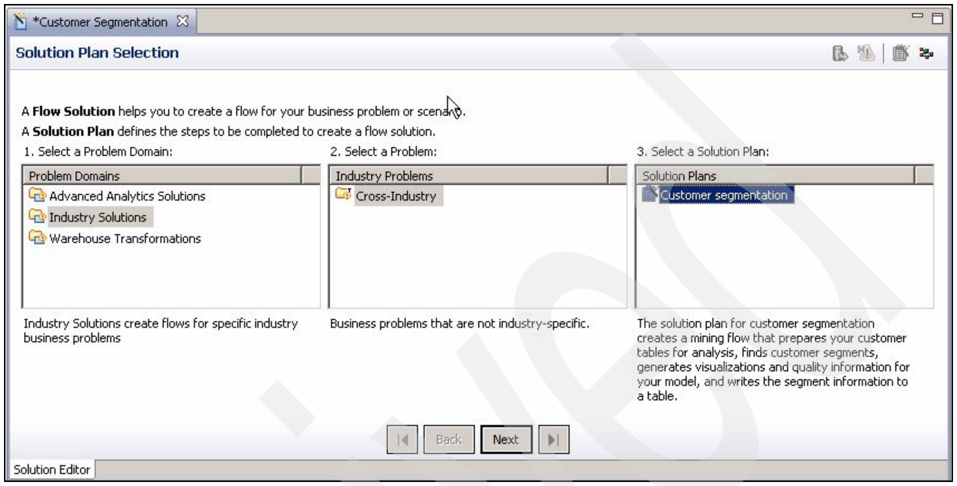


Figure 7-44 Launching the solution plan wizard

When you use a solution plan, the solution editor prompts you to define the parameters that are necessary to create your flow. You can use the pre-defined solution plans to create flows that address many common scenarios. The solution plan selection is illustrated in Figure 7-45.


*Customer Segmentation

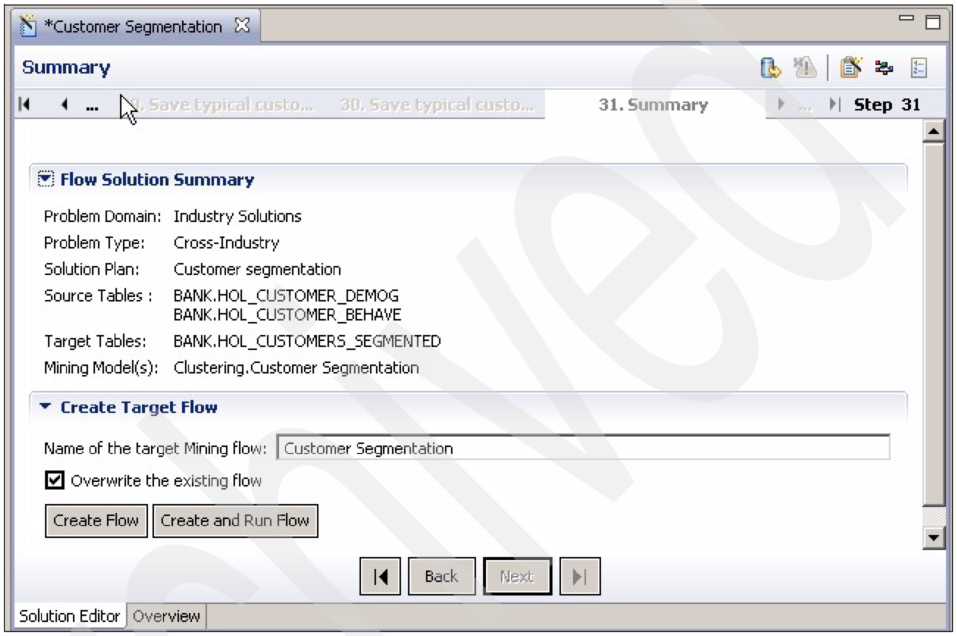
Solution Plan Selection

A **Flow Solution** helps you to create a flow for your business problem or scenario.
A **Solution Plan** defines the steps to be completed to create a flow solution.

1. Select a Problem Domain:	2. Select a Problem:	3. Select a Solution Plan:
Problem Domains	Industry Problems	Solution Plans
Advanced Analytics Solutions Industry Solutions Warehouse Transformations	Cross-Industry	Customer segmentation
Industry Solutions create flows for specific industry business problems	Business problems that are not industry-specific.	The solution plan for customer segmentation creates a mining flow that prepares your customer tables for analysis, finds customer segments, generates visualizations and quality information for your model, and writes the segment information to a table.

Figure 7-45 Solution Plan Selection

Once a solution plan has been selected, several parameters need to be defined, including selecting the relevant database, the type of data to use (demographic, behavioral, or both), and which source tables to use. The user can also specify which algorithm to use for the solution plan, or let the wizard decide based on the input table. The solution plan summary tab lists the source and target tables, and give the user the options of creating and running the flow, or creating the flow to run later, as shown in Figure 7-46.



Problem Domain:	Industry Solutions
Problem Type:	Cross-Industry
Solution Plan:	Customer segmentation
Source Tables :	BANK.HOL_CUSTOMER_DEMOG BANK.HOL_CUSTOMER_BEHAVE
Target Tables:	BANK.HOL_CUSTOMERS_SEGMENTED
Mining Model(s):	Clustering.Customer Segmentation

Name of the target Mining flow: Customer Segmentation

☑ Overwrite the existing flow

Figure 7-46 Solution plan summary

7.5.8 Validating and executing a mining flow

When a mining flow is executed, all of the operations represented by the operators are executed in the order defined by the flow, as described in 7.5.6, “Building a mining flow” on page 287. The models generated in the flow are stored in PMML format in the appropriate DB2 tables under the schema IDMMX, as described in 7.5.1, “Database enablement” on page 257. Models are applied to new data for scoring, and scored results are written out to target tables. Visualizations of trained models and testing results are displayed to enable the analyst to understand, assess, and fine-tune the models.

Validating a mining flow

Prior to execution, the mining flow can be validated to ensure that operators are properly and completely connected. Although validation is not a prerequisite for execution, it can identify problems that would cause the execution to fail. A mining flow is validated either by selecting **Mining Flow** → **Validate Flow** from the menu or by clicking the **Validate this mining flow** icon on the toolbar. Successful validation is indicated by a pop-up message. If errors occur, the problems can be resolved prior to executing the mining flow. An incomplete mining flow might be validated during construction to verify the flow up to that point. In this case, errors can be expected and can usually be ignored until the complete flow is validated.

Executing a mining flow

The completed mining flow is executed either by selecting **Mining Flow** → **Execute** from the menu or by clicking the **Execute this mining flow** icon on the toolbar. An menu offers the choice of executing the complete mining flow or executing the mining flow step by step, as shown in Figure 7-47. The latter choice is particularly useful when debugging an execution error.

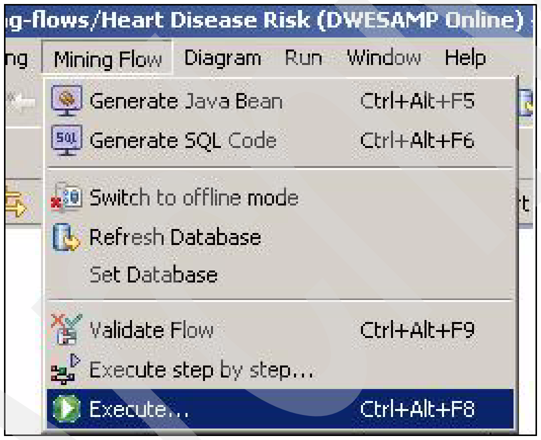
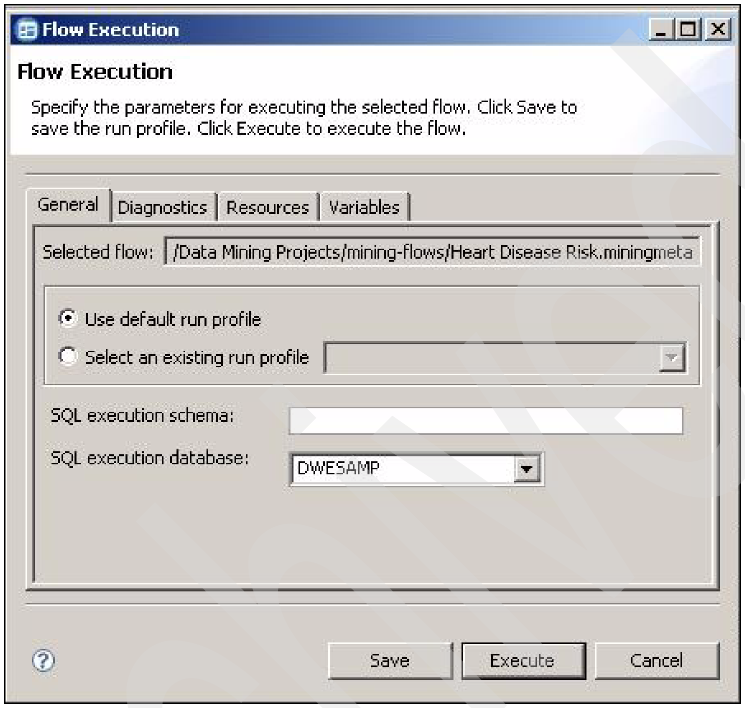


Figure 7-47 Mining flow execution options

An execution selection pop-up opens where you can set variables, change the diagnostics level, and set the SQL execution database. The complete mining flow is executed by accepting the default method and clicking the **Execute** button, as shown in Figure 7-48.


Flow Execution

Flow Execution

Specify the parameters for executing the selected flow. Click Save to save the run profile. Click Execute to execute the flow.

General | Diagnostics | Resources | Variables

Selected flow: /Data Mining Projects/mining-flows/Heart Disease Risk.miningmeta

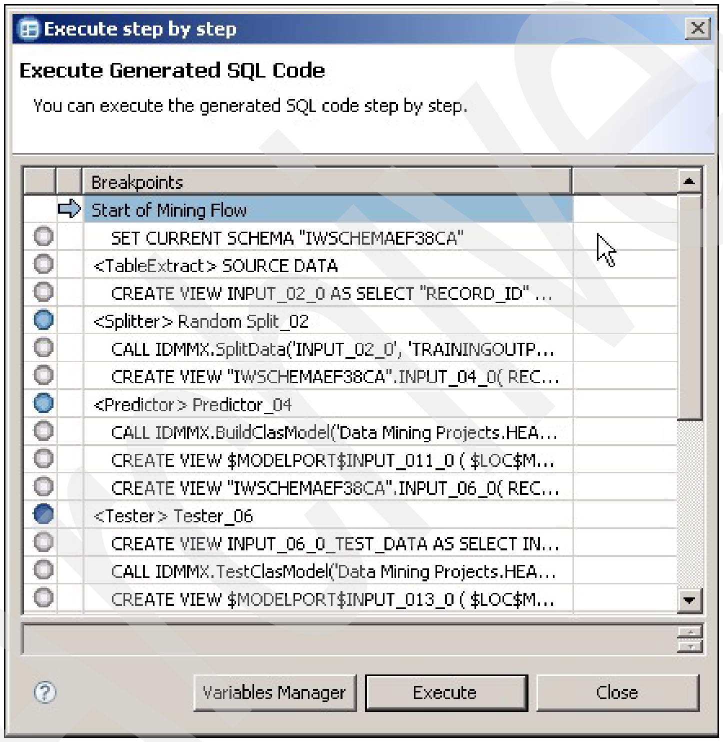
(•) Use default run profile
( ) Select an existing run profile

SQL execution schema:
SQL execution database: DWESAMP

Figure 7-48 Executing a complete mining flow

The mining flow can be executed step-by-step by selecting that execution method (as shown in Figure 7-47 on page 300) and clicking the radio buttons for the desired steps. For example, in Figure 7-49, three steps (random split, predictor, tester) have been selected. The mining flow is initially executed up to the random split operator. If execution to that point is successful, execution is resumed until the predictor is reached.

If successful to that point, execution is resumed up to the tester, and finally to the end of the mining flow. If an execution error occurs, this step-by-step execution process is helpful in isolating the point at which the error occurs. If desired as an additional aid in debugging the mining flow, the SQL for the mining run can be displayed by selecting **Mining Flow** → **Generate SQL Code**.


Execute step by step
Execute Generated SQL Code
You can execute the generated SQL code step by step.
Variables Manager
Execute
Close

		Breakpoints
	⇨	Start of Mining Flow
○		SET CURRENT SCHEMA "IWSCHEMAEF38CA"
○		<TableExtract> SOURCE DATA
○		CREATE VIEW INPUT_02_0 AS SELECT "RECORD_ID" ...
●		<Splitter> Random Split_02
○		CALL IDMMX.SplitData('INPUT_02_0', 'TRAININGOUTP...
○		CREATE VIEW "IWSCHEMAEF38CA".INPUT_04_0(REC...
●		<Predictor> Predictor_04
○		CALL IDMMX.BuildClasModel('Data Mining Projects.HEA...
○		CREATE VIEW $MODELPORT$INPUT_011_0 (LOCM...
○		CREATE VIEW "IWSCHEMAEF38CA".INPUT_06_0(REC...
●		<Tester> Tester_06
○		CREATE VIEW INPUT_06_0_TEST_DATA AS SELECT IN...
○		CALL IDMMX.TestClasModel('Data Mining Projects.HEA...
○		CREATE VIEW $MODELPORT$INPUT_013_0 (LOCM...

Figure 7-49 Executing a mining flow step-by-step

In addition to executing a complete mining flow, a selected portion of a mining flow can be executed without having to execute the entire flow. This feature is useful, for example, when verifying a portion of a mining flow during construction or rerunning a model after changing a parameter. As illustrated in Figure 7-50, a mining flow is executed up to a selected step by right-clicking the desired operator and selecting **Run to this step** from the pop-up menu. If the selected operator links to other operators not directly in the branch, warnings might appear but can usually be ignored unless they indicate an error.

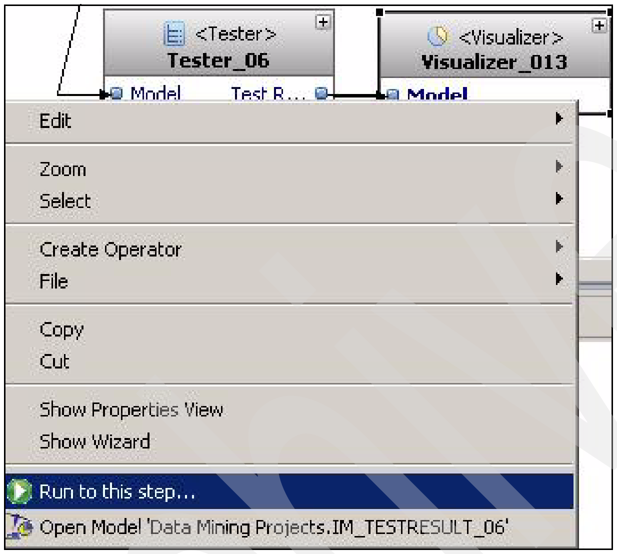


Figure 7-50 Executing a mining flow up to a selected step

Executing mining in other flows

Certain data mining modeling operators (Find Deviations, Find Rules, Cluster Table, Predict Column) are available in data flows. This enables data mining models to be created and applied in an operational data flow (for example, during nightly data loading operations, to refresh a set of models based on newly processed transactions, and to apply those models to update clients' scores for use by client-facing applications).

A mining flow can be deployed by incorporating it into a control flow, thereby sequencing the mining operations with data flows and other data processing activities, as discussed in Chapter 5, "Data movement and transformation" on page 75. A simple example of a control flow with an embedded mining flow is shown in Figure 7-51. The mining flow is represented in the control flow sequence by a mining flow operator selected from the palette and placed onto the canvas. The mining flow is defined as a property of the mining flow operator. Execution of the control flow invokes execution of the mining flow.

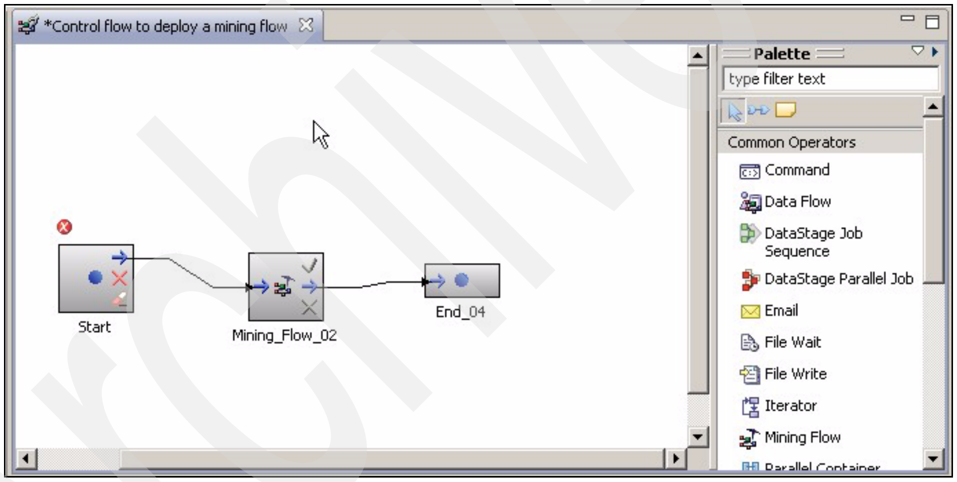


Figure 7-51 Control flow to deploy a mining flow

7.5.9 Visualizing models and test results

As discussed in 7.2.6, “Interpreting and evaluating the data mining results” on page 250, each visualizer presents technique-specific information about the model or testing result to enable the analyst to evaluate the model quality and performance. Visualizations generated by mining flows can be viewed in Design Studio by any of the following three methods:

- When executing a mining flow

  All visualizations generated during a mining flow are automatically displayed. For example, for a mining flow that includes training and testing functions, such as the flow illustrated in Figure 7-35 on page 290, the visualizations for the trained model and the test result are both displayed.

- From a visualizer operator in a previously executed mining flow

  A visualization of a model or test result previously generated in a mining flow can be opened from its visualizer operator, as shown in Figure 7-52. The visualization is opened by right-clicking the visualizer operator and selecting **Open Model <*name*>** from the pop-up menu.

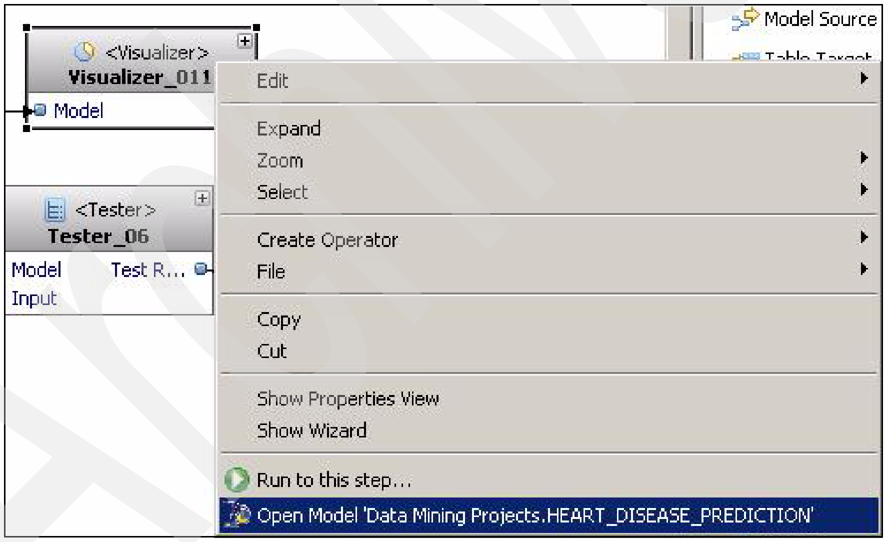


Figure 7-52 Opening a visualization from in a mining flow

- From the Data Source Explorer

  A visualization of a model or test result previously generated in a mining flow can be opened from the Data Source Explorer, as shown in Figure 7-53. The visualization is opened by expanding the database's folder structure to the appropriate folder (classification, clustering, regression, rules, time series) under the Data Mining Models folder and either double-clicking the desired model or right-clicking and selecting **Open** from the pop-up menu.

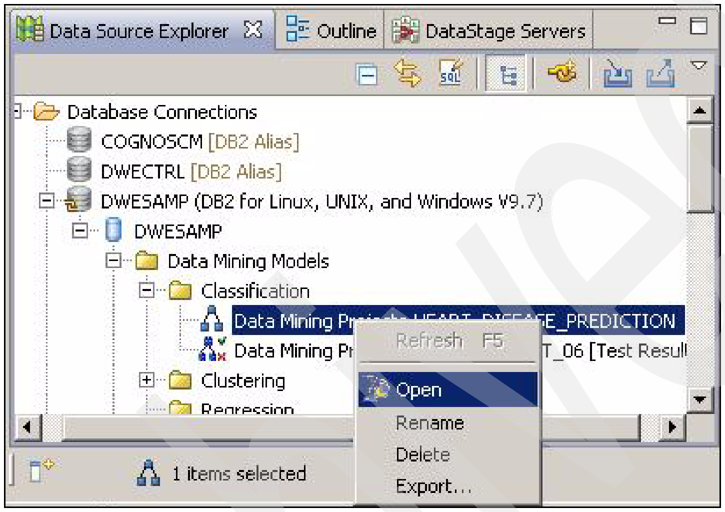


Figure 7-53 Opening a visualization from the Data Source Explorer

Each visualizer presents information specific to the mining method used. Thus, interpretation of the modeling or testing results also depends on the mining method. Explanations of each view in a visualization are available through the help function on the menu bar in Design Studio. Extensive documentation on all of the visualizers as well as modeling and scoring is provided in the Data Warehousing and Analytics section of the DB2 Information Center installed as part of InfoSphere Warehouse.

Clustering Visualization

To illustrate the process of interpreting a visualization we present a clustering example. The business problem for this example is to identify a group of customers that are best suited for a particular marketing campaign. In this example we cluster (or categorize) bank customers according to their demographics and behavioral information, such as their average balance.

The mining flow for the clustering model is shown in Figure 7-54. The mining flow points to the source table containing one record per customer, with columns representing the demographic and behavioral information. These columns are used as variables in the clustering model. The visualizer generates a visualization of the model.

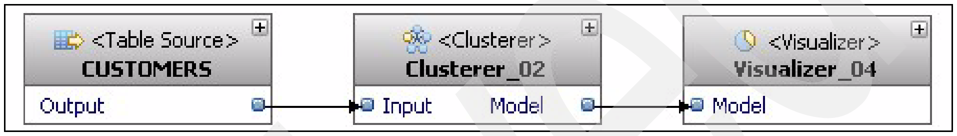


Figure 7-54 Clustering mining flow example

The main view of the visualization generated by the mining flow is shown in Figure 7-55. The clusters are displayed as rows sorted by size (percentage of rows in each cluster). in each cluster, the variables are sorted in descending order of importance (as measured by a chi-squared statistic) in distinguishing the members of that cluster from all other customers. Because the objective of clustering is to segment individuals into homogenous groups that are different from other groups, we interpret clusters by comparing each variable's distribution for a given cluster to its distribution for all customers.

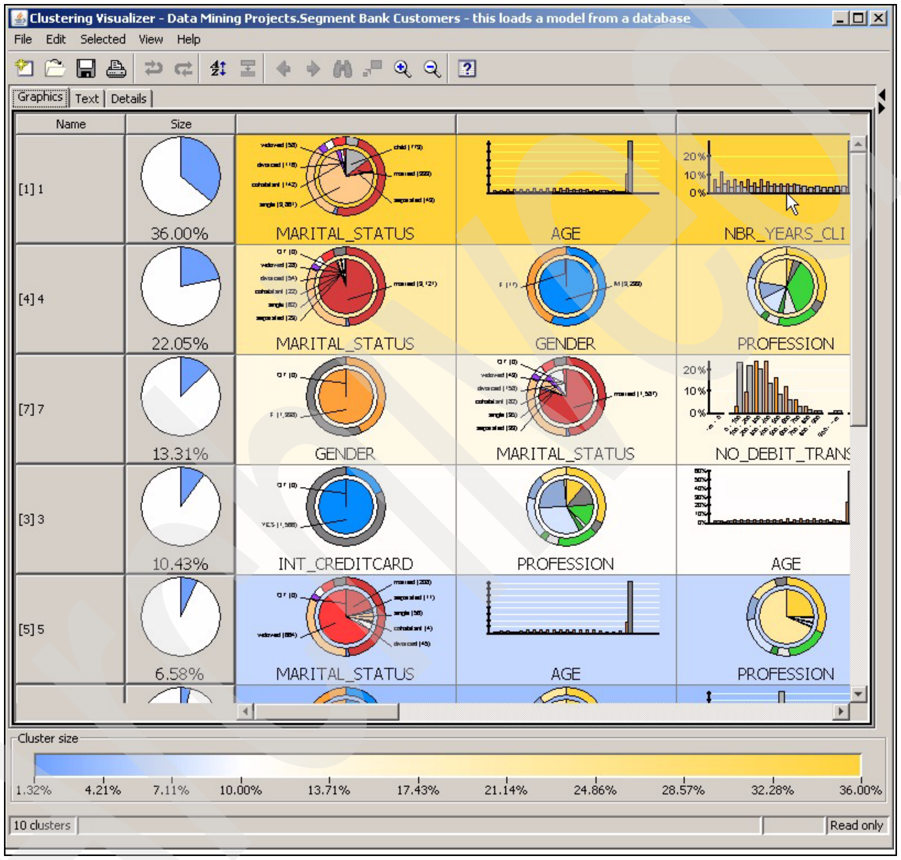

Clustering Visualizer - Data Mining Projects.Segment Bank Customers - this loads a model from a database
File Edit Selected View Help
Graphics Text Details
Name Size
[1] 1 36.00% MARITAL_STATUS AGE NBR_YEARS_CLI
[4] 4 22.05% MARITAL_STATUS GENDER PROFESSION
[7] 7 13.31% GENDER MARITAL_STATUS NO_DEBIT_TRANS
[3] 3 10.43% INT_CREDITCARD PROFESSION AGE
[5] 5 6.58% MARITAL_STATUS AGE PROFESSION
Cluster size
1.32% 4.21% 7.11% 10.00% 13.71% 17.43% 21.14% 24.86% 28.57% 32.28% 36.00%
10 clusters
Read only

Figure 7-55 Clustering visualization example

For a categorical variable, the distributions are represented by concentric pie charts, as illustrated (for a variable called MARITAL_STATUS) in Figure 7-56.

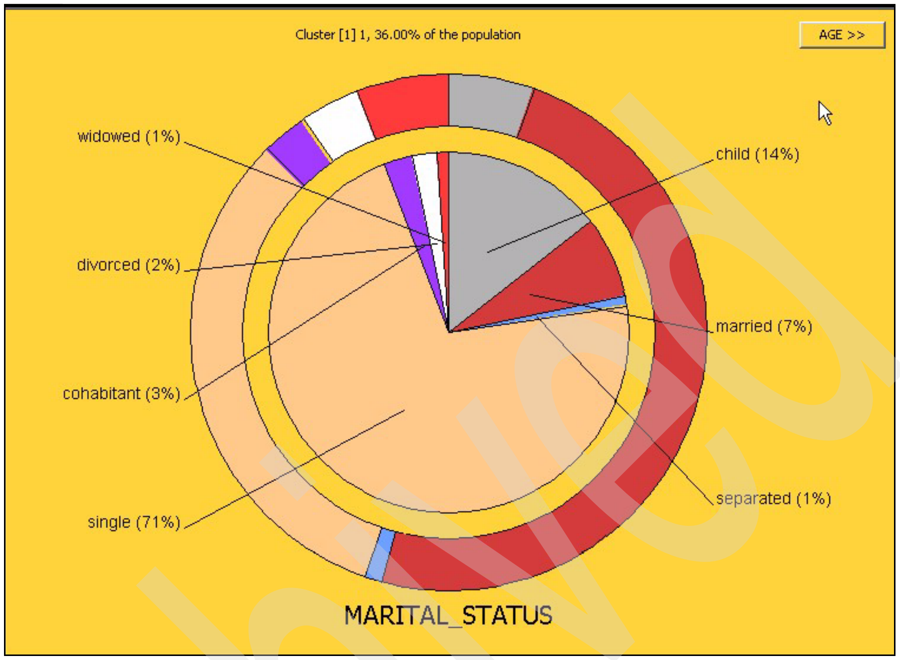


Figure 7-56 Representation of distributions of a categorical variable

The inner circle represents a particular cluster, while the outer ring represents all customers as a whole. In this example, we see that 71% of customers in this cluster are single, compared to 32% of the whole population. 7% of the customers in this cluster are married, compared to 49% of the whole population. Thus, we interpret this cluster as being comprised primarily of customers who are single, in contrast to all of the bank's customers.

For a numeric variable, the distributions are represented by percentage histograms, as illustrated for a variable called AVERAGE_BALANCE in Figure 7-57. The foreground histogram represents a particular cluster, while the background histogram represents all of the customers as a whole. In this cluster we see that 42% of the customers have an average balance of 0–20,000, compared to 73% of the whole. 16% of this cluster have an average balance of 20,000–40,000, compared to 12% of the whole. Thus, we interpret this cluster as being comprised of customers that have a higher-than-average account balance.

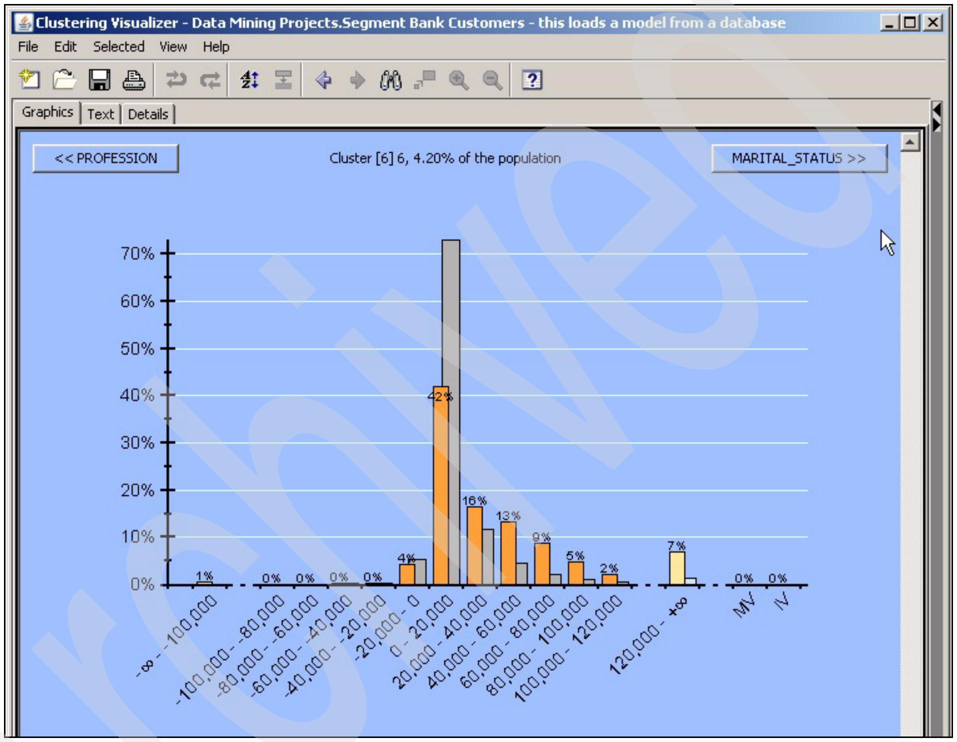


Figure 7-57 Representation of distributions of a numeric variable

When examining a clustering model, the differences among the clusters can be discerned by visually comparing the distributions of the most important (highest-ranked) variables across clusters. The relative sizes of the inner (cluster) and outer (all records) values for categorical variables and the relative positions of the foreground (cluster) and background (all records) histograms for numeric variables across the top several variables provide a readily interpretable overview of the distinct characteristic profiles represented by the clusters. In addition, the variables can be ordered in various ways (for example, alphabetically or by average values of a particular variable) or displayed as pie charts, histograms, or tables to assist with interpreting the model. Many other options such as colors and chart displays are available as well.

In this example we are interested in identifying a group of customers who would be best suited for a particular marketing campaign. We search the clusters to identify one that best represents a distinct group of customers with ideal characteristics. We can drill down on individual clusters to assess them in more detail, as shown in Figure 7-58.

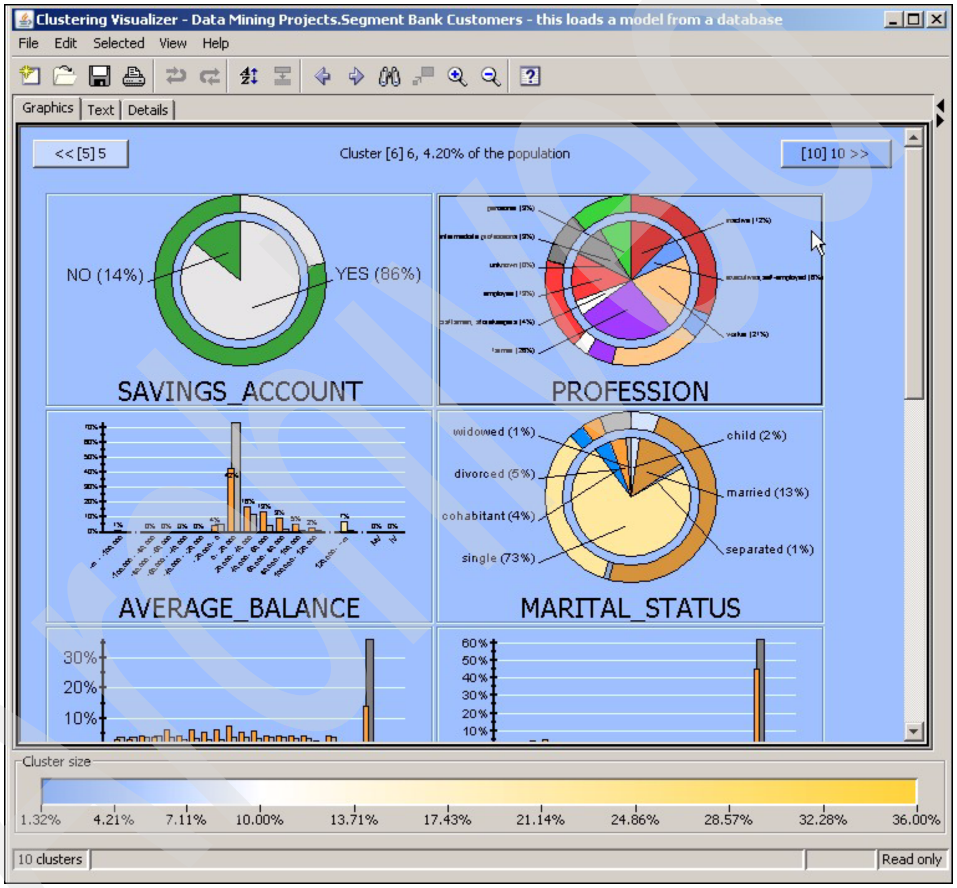


Figure 7-58   Clustering visualization example: Drill down to one cluster

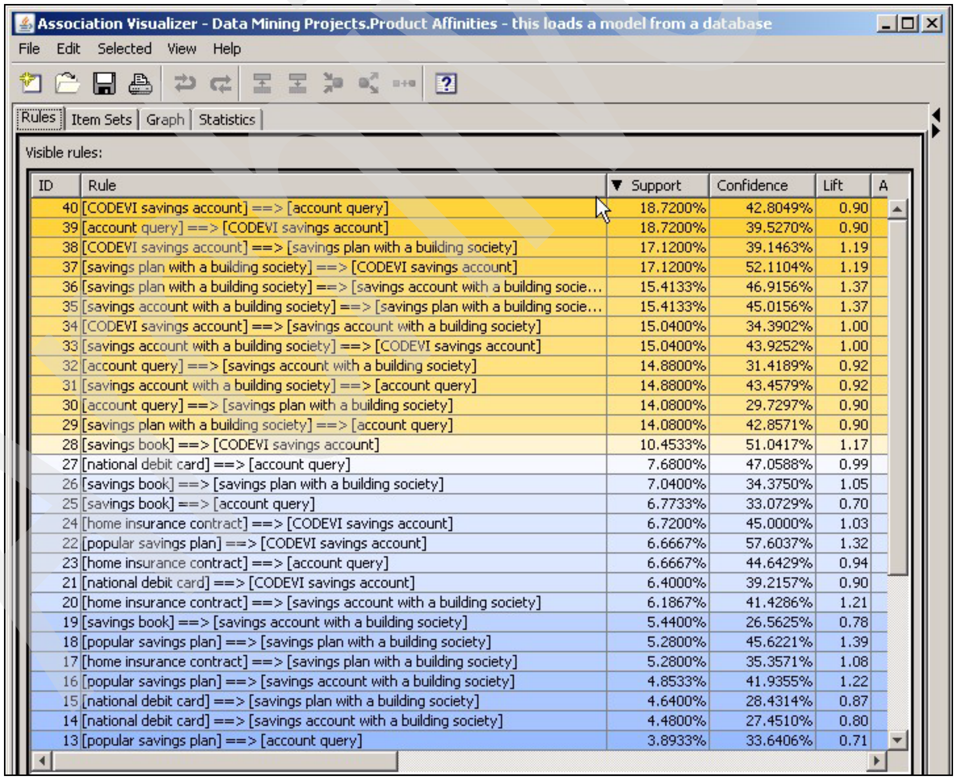
Associations visualization

The associations visualizer presents a set of rules describing affinities among items or higher levels of taxonomy that occur together in a transaction, as discussed in the following sections of this IBM Redbooks publication:

- 7.1.1, “Discovery data mining” on page 244
- 7.2.3, “Understanding the data” on page 247
- 7.2.4, “Preparing the data” on page 249

The associations visualization includes tables of affinity rules, the items sets on which the rules are based, a graph of the rules, and a table of statistics about the transactions data, mining parameters, and rules. The graph and tables are interactive to aid the analyst in focusing on rules of interest for a particular business objective.

In this example, the rules table is shown in Figure 7-59. The rules can be sorted by any column in this table, and various filtering capabilities are available to enable the analyst to focus on a subset of item relationships of interest.

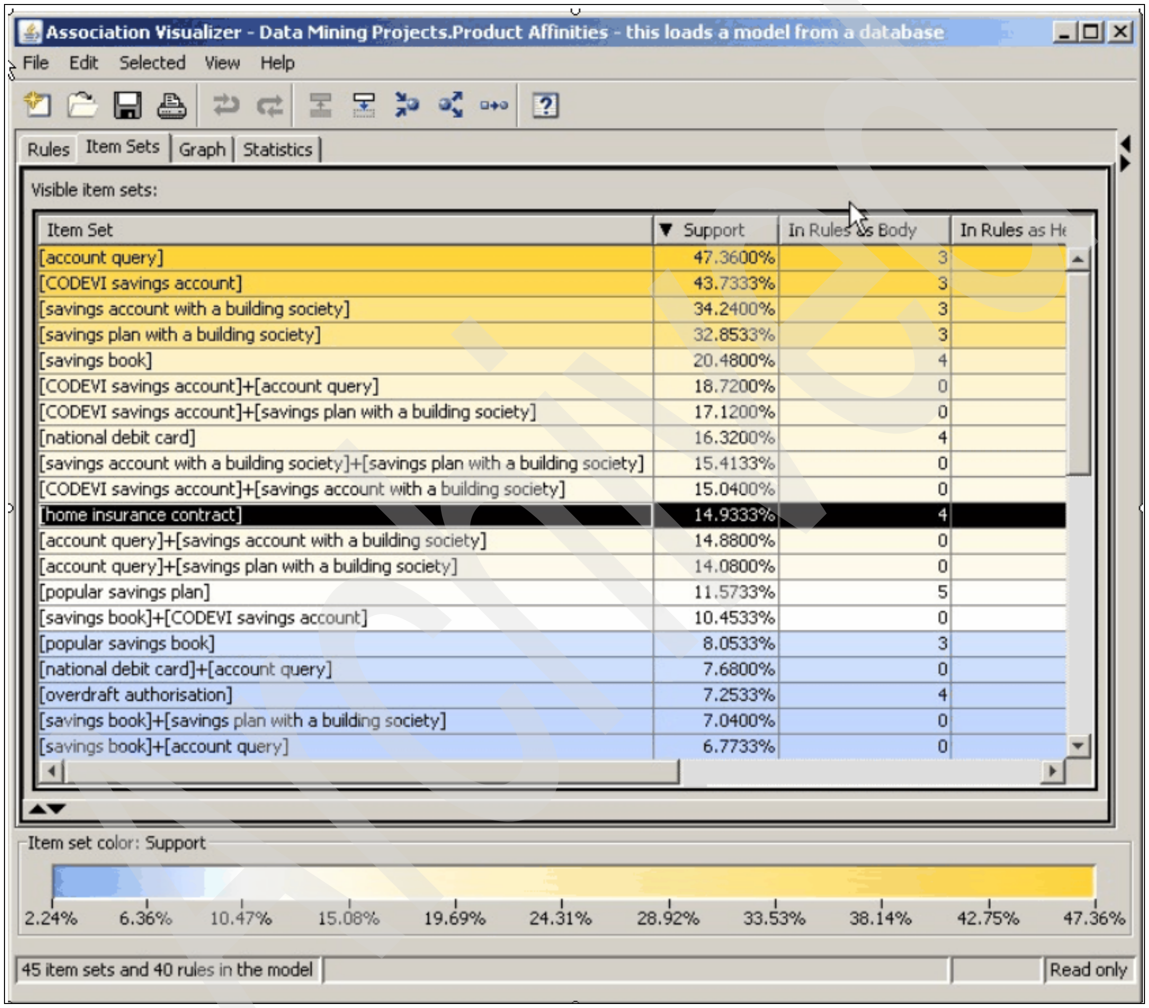

Association Visualizer - Data Mining Projects.Product Affinities - this loads a model from a database
File
Edit
Selected
View
Help
Rules
Item Sets
Graph
Statistics

Visible rules:

ID	Rule	▼ Support	Confidence	Lift	A
40	[CODEVI savings account] ==> [account query]	18.7200%	42.8049%	0.90	
39	[account query] ==> [CODEVI savings account]	18.7200%	39.5270%	0.90	
38	[CODEVI savings account] ==> [savings plan with a building society]	17.1200%	39.1463%	1.19	
37	[savings plan with a building society] ==> [CODEVI savings account]	17.1200%	52.1104%	1.19	
36	[savings plan with a building society] ==> [savings account with a building socie...	15.4133%	46.9156%	1.37	
35	[savings account with a building society] ==> [savings plan with a building socie...	15.4133%	45.0156%	1.37	
34	[CODEVI savings account] ==> [savings account with a building society]	15.0400%	34.3902%	1.00	
33	[savings account with a building society] ==> [CODEVI savings account]	15.0400%	43.9252%	1.00	
32	[account query] ==> [savings account with a building society]	14.8800%	31.4189%	0.92	
31	[savings account with a building society] ==> [account query]	14.8800%	43.4579%	0.92	
30	[account query] ==> [savings plan with a building society]	14.0800%	29.7297%	0.90	
29	[savings plan with a building society] ==> [account query]	14.0800%	42.8571%	0.90	
28	[savings book] ==> [CODEVI savings account]	10.4533%	51.0417%	1.17	
27	[national debit card] ==> [account query]	7.6800%	47.0588%	0.99	
26	[savings book] ==> [savings plan with a building society]	7.0400%	34.3750%	1.05	
25	[savings book] ==> [account query]	6.7733%	33.0729%	0.70	
24	[home insurance contract] ==> [CODEVI savings account]	6.7200%	45.0000%	1.03	
22	[popular savings plan] ==> [CODEVI savings account]	6.6667%	57.6037%	1.32	
23	[home insurance contract] ==> [account query]	6.6667%	44.6429%	0.94	
21	[national debit card] ==> [CODEVI savings account]	6.4000%	39.2157%	0.90	
20	[home insurance contract] ==> [savings account with a building society]	6.1867%	41.4286%	1.21	
19	[savings book] ==> [savings account with a building society]	5.4400%	26.5625%	0.78	
18	[popular savings plan] ==> [savings plan with a building society]	5.2800%	45.6221%	1.39	
17	[home insurance contract] ==> [savings plan with a building society]	5.2800%	35.3571%	1.08	
16	[popular savings plan] ==> [savings account with a building society]	4.8533%	41.9355%	1.22	
15	[national debit card] ==> [savings plan with a building society]	4.6400%	28.4314%	0.87	
14	[national debit card] ==> [savings account with a building society]	4.4800%	27.4510%	0.80	
13	[popular savings plan] ==> [account query]	3.8933%	33.6406%	0.71	

Figure 7-59 Associations visualization example: Rules

The table of item sets used in constructing the rules is shown in Figure 7-60. This table is particularly useful for selecting a particular item set (containing one or more items) and displaying the rules and graph just for that item set. For example, if the analyst wants to focus on household insurance (the highlighted item set in the table in Figure 7-60) for a promotion, then he can select that item set and display only the rules containing household insurance.


Association Visualizer - Data Mining Projects.Product Affinities - this loads a model from a database
File Edit Selected View Help
Rules
Item Sets
Graph
Statistics
Visible item sets:
Item set color: Support
2.24% 6.36% 10.47% 15.08% 19.69% 24.31% 28.92% 33.53% 38.14% 42.75% 47.36%
45 item sets and 40 rules in the model
Read only

Item Set	▼ Support	In Rules as Body	In Rules as He
[account query]	47.3600%	3	
[CODEVI savings account]	43.7333%	3	
[savings account with a building society]	34.2400%	3	
[savings plan with a building society]	32.8533%	3	
[savings book]	20.4800%	4	
[CODEVI savings account]+[account query]	18.7200%	0	
[CODEVI savings account]+[savings plan with a building society]	17.1200%	0	
[national debit card]	16.3200%	4	
[savings account with a building society]+[savings plan with a building society]	15.4133%	0	
[CODEVI savings account]+[savings account with a building society]	15.0400%	0	
[home insurance contract]	14.9333%	4	
[account query]+[savings account with a building society]	14.8800%	0	
[account query]+[savings plan with a building society]	14.0800%	0	
[popular savings plan]	11.5733%	5	
[savings book]+[CODEVI savings account]	10.4533%	0	
[popular savings book]	8.0533%	3	
[national debit card]+[account query]	7.6800%	0	
[overdraft authorisation]	7.2533%	4	
[savings book]+[savings plan with a building society]	7.0400%	0	
[savings book]+[account query]	6.7733%	0	

Figure 7-60 Associations visualization example: Item sets

A graphical representation of the rules table is shown in Figure 7-61. As the figure clearly illustrates, displaying all the rules at once can result in a complex and messy graph, making it unreadable. Much of the data overlap is because of the density of data in a particular area or regarding a particular set of attributes and rules. You likely need to focus on a subset of the rules.

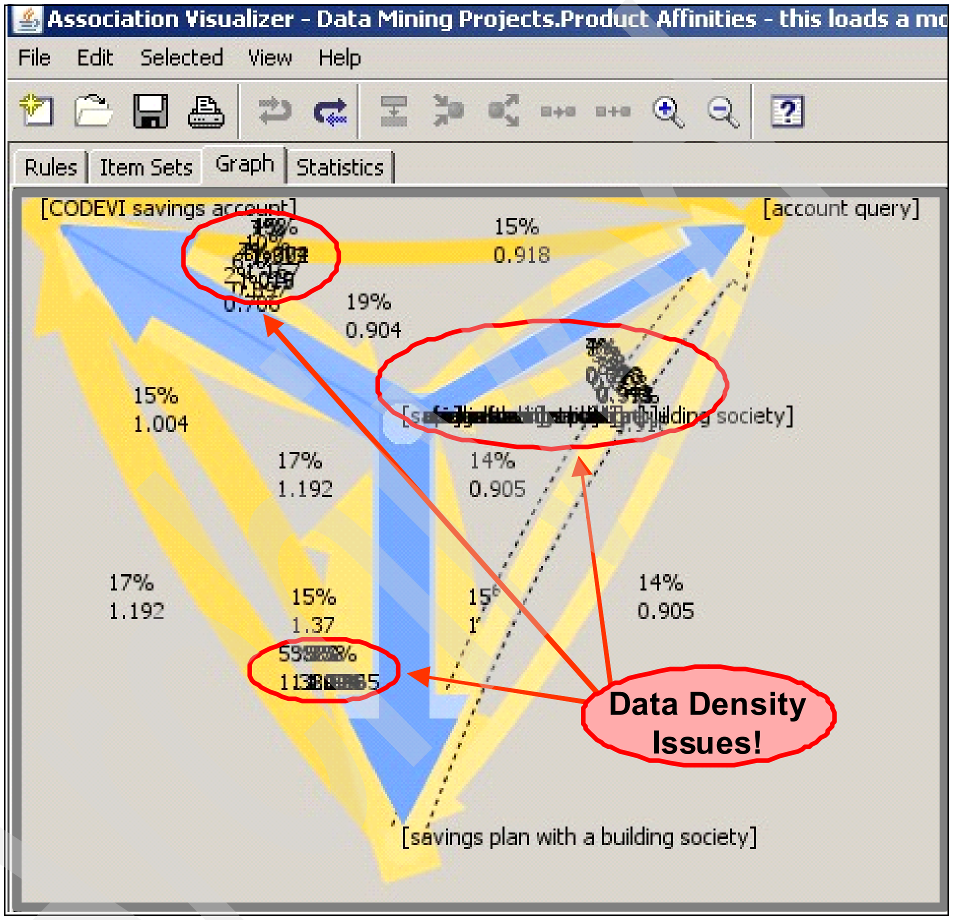


Figure 7-61   Graph of all rules

When a subset of rules of interest for a particular business objective is selected, the combination of the graph and rules table becomes much more useful. For example, if the objective is to identify a set of items associated with household insurance for an upcoming promotion, then the household insurance item set can be selected. Then only the rules containing household insurance are displayed in the rules table and in the graph, as shown in Figure 7-62.

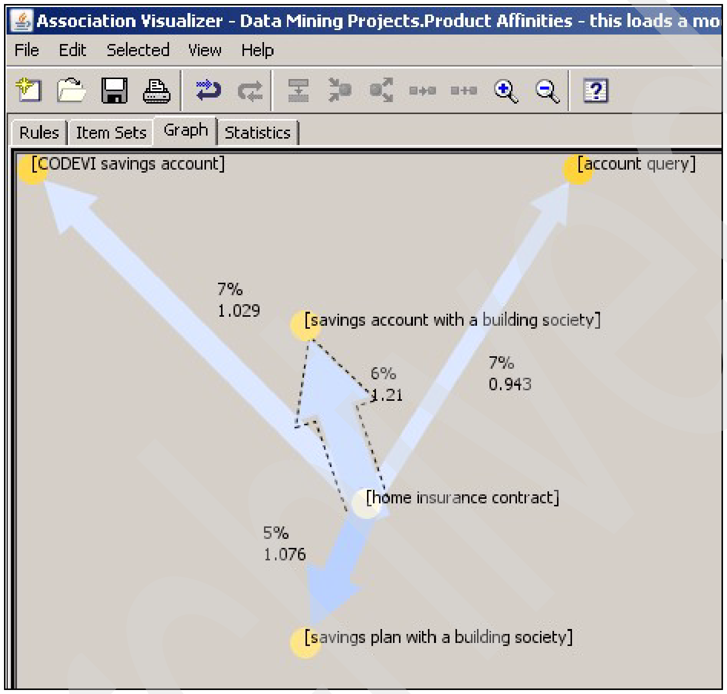

Association Visualizer - Data Mining Projects.Product Affinities - this loads a mo
File
Edit
Selected
View
Help
Rules
Item Sets
Graph
Statistics
[CODEVI savings account]
[account query]
7%
1.029
[savings account with a building society]
6%
1.21
7%
0.943
[home insurance contract]
5%
1.076
[savings plan with a building society]

Figure 7-62 Graph of rules for a selected item set

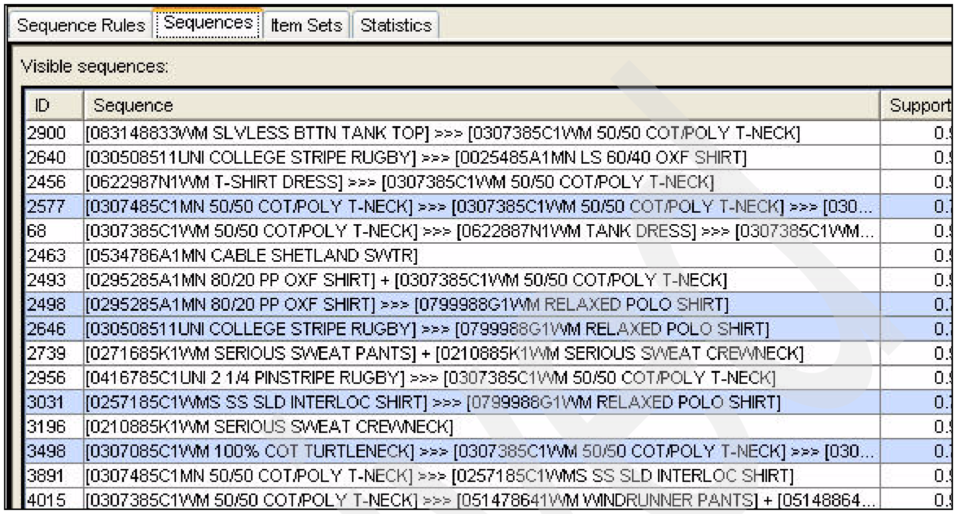
Once the relevant subset of rules has been found, the analyst can evaluate those rules in terms of the business objective.

Sequences visualization

The sequences visualizer presents a set of rules describing a series of transactions that occur sequentially over time, as discussed in the following sections of this book:

- 7.1.1, “Discovery data mining” on page 244
- 7.5.5, “Data mining operators” on page 273
- 7.5.5, “Data mining operators” on page 273

The sequences visualization includes tables of sequence rules, sequences and item sets on which the rules are based, a graph of the rules, and a table of statistics about the rules. In this example, views of the sequence rules are shown in Figure 7-63.

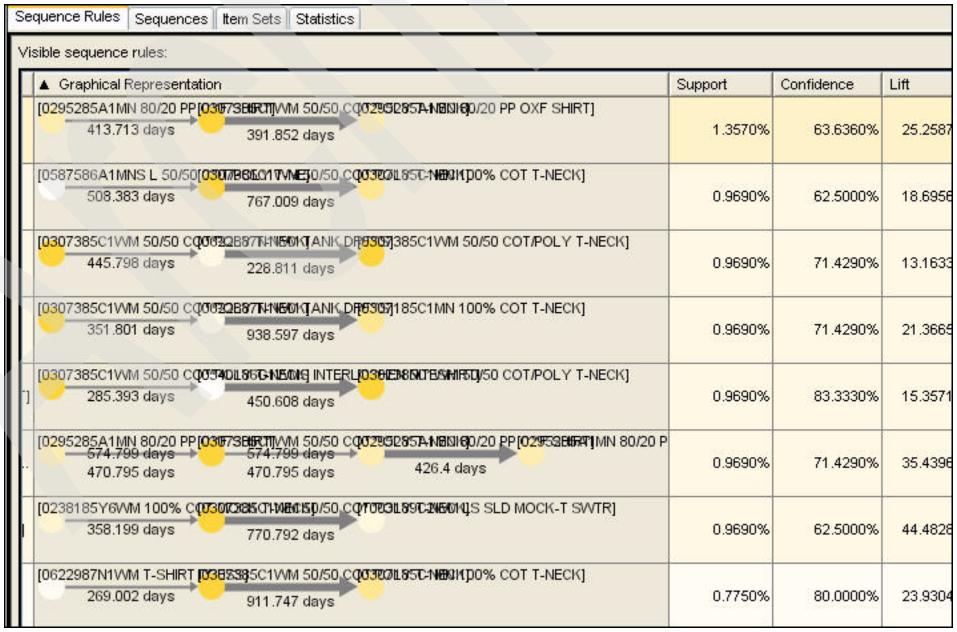


Visible sequences:

ID	Sequence	Support
2900	[083148833WM SLVLESS BTTN TANK TOP] >>> [0307385C1WM 50/50 COT/POLY T-NECK]	0.9
2640	[030508511UNI COLLEGE STRIPE RUGBY] >>> [0025485A1MN LS 60/40 OXF SHIRT]	0.9
2456	[0622987N1WM T-SHIRT DRESS] >>> [0307385C1WM 50/50 COT/POLY T-NECK]	0.9
2577	[0307485C1MN 50/50 COT/POLY T-NECK] >>> [0307385C1WM 50/50 COT/POLY T-NECK] >>> [030...	0.7
68	[0307385C1WM 50/50 COT/POLY T-NECK] >>> [0622887N1WM TANK DRESS] >>> [0307385C1WM...	0.9
2463	[0534786A1MN CABLE SHETLAND SWTR]	0.9
2493	[0295285A1MN 80/20 PP OXF SHIRT] + [0307385C1WM 50/50 COT/POLY T-NECK]	0.9
2498	[0295285A1MN 80/20 PP OXF SHIRT] >>> [0799988G1WM RELAXED POLO SHIRT]	0.7
2646	[030508511UNI COLLEGE STRIPE RUGBY] >>> [0799988G1WM RELAXED POLO SHIRT]	0.7
2739	[0271685K1WM SERIOUS SWEAT PANTS] + [0210885K1WM SERIOUS SWEAT CREWNECK]	0.9
2956	[0416785C1UNI 2 1/4 PINSTRIPE RUGBY] >>> [0307385C1WM 50/50 COT/POLY T-NECK]	0.9
3031	[0257185C1WMS SS SLD INTERLOC SHIRT] >>> [0799988G1WM RELAXED POLO SHIRT]	0.7
3196	[0210885K1WM SERIOUS SWEAT CREWNECK]	0.9
3498	[0307085C1WM 100% COT TURTLENECK] >>> [0307385C1WM 50/50 COT/POLY T-NECK] >>> [030...	0.7
3891	[0307485C1MN 50/50 COT/POLY T-NECK] >>> [0257185C1WMS SS SLD INTERLOC SHIRT]	0.9
4015	[0307385C1WM 50/50 COT/POLY T-NECK] >>> [051478641WM WINDRUNNER PANTS] + [05148864...	0.9

Figure 7-63   Sequences visualization example: Sequences table

The graphical representation of those sequence rules is illustrated in Figure 7-64.



Visible sequence rules:

▲ Graphical Representation	Support	Confidence	Lift
[0295285A1MN 80/20 PP [illegible] 80/20 PP OXF SHIRT] 413.713 days 391.852 days	1.3570%	63.6360%	25.2587
[0587586A1MNS L 50/50 [illegible] 100% COT T-NECK] 508.383 days 767.009 days	0.9690%	62.5000%	18.6958
[0307385C1WM 50/50 C[illegible] TANK DR[illegible]385C1WM 50/50 COT/POLY T-NECK] 445.798 days 228.811 days	0.9690%	71.4290%	13.1633
[0307385C1WM 50/50 C[illegible] TANK DR[illegible]185C1MN 100% COT T-NECK] 351.801 days 938.597 days	0.9690%	71.4290%	21.3665
[0307385C1WM 50/50 C[illegible] INTERL[illegible]/50 COT/POLY T-NECK] 285.393 days 450.608 days	0.9690%	83.3330%	15.3571
[0295285A1MN 80/20 PP [illegible] 80/20 PP [illegible]MN 80/20 P 574.799 days 470.795 days 574.799 days 470.795 days 426.4 days	0.9690%	71.4290%	35.4398
[0238185Y6WM 100% C[illegible]LS SLD MOCK-T SWTR] 358.199 days 770.792 days	0.9690%	62.5000%	44.4828
[0622987N1WM T-SHIRT [illegible]5C1WM 50/50 C[illegible]100% COT T-NECK] 269.002 days 911.747 days	0.7750%	80.0000%	23.9304

Figure 7-64   Sequence visualization example: Sequence rules

Classification visualization

The classification visualizer presents information to evaluate a classification model and its testing result, as discussed in the following sections of this book:

- 7.1.2, “Predictive data mining” on page 245
- 7.5.6, “Building a mining flow” on page 287
- Figure 7-35 on page 290.

In this section, we consider an example of predicting the risk that a patient will have heart disease. The response variable DISEASED has possible values of Y (yes) and N (no).

A useful graph from the visualization of the trained model is the field importance chart illustrated in Figure 7-65. This graph shows the relative contribution of each of the explanatory variables used in the model to predict the DISEASED response. In this case, the variable THAL is the most important predictor of heart disease, followed by PAIN_TYPE, ANGINA, and so on. The variable CHOLESTEROL contributes the least to explaining heart disease in this model.

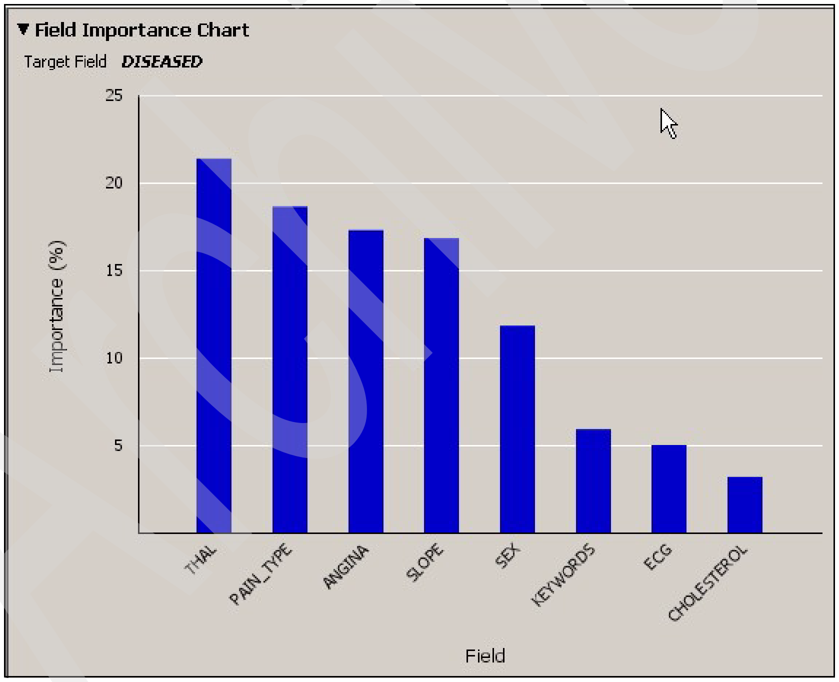
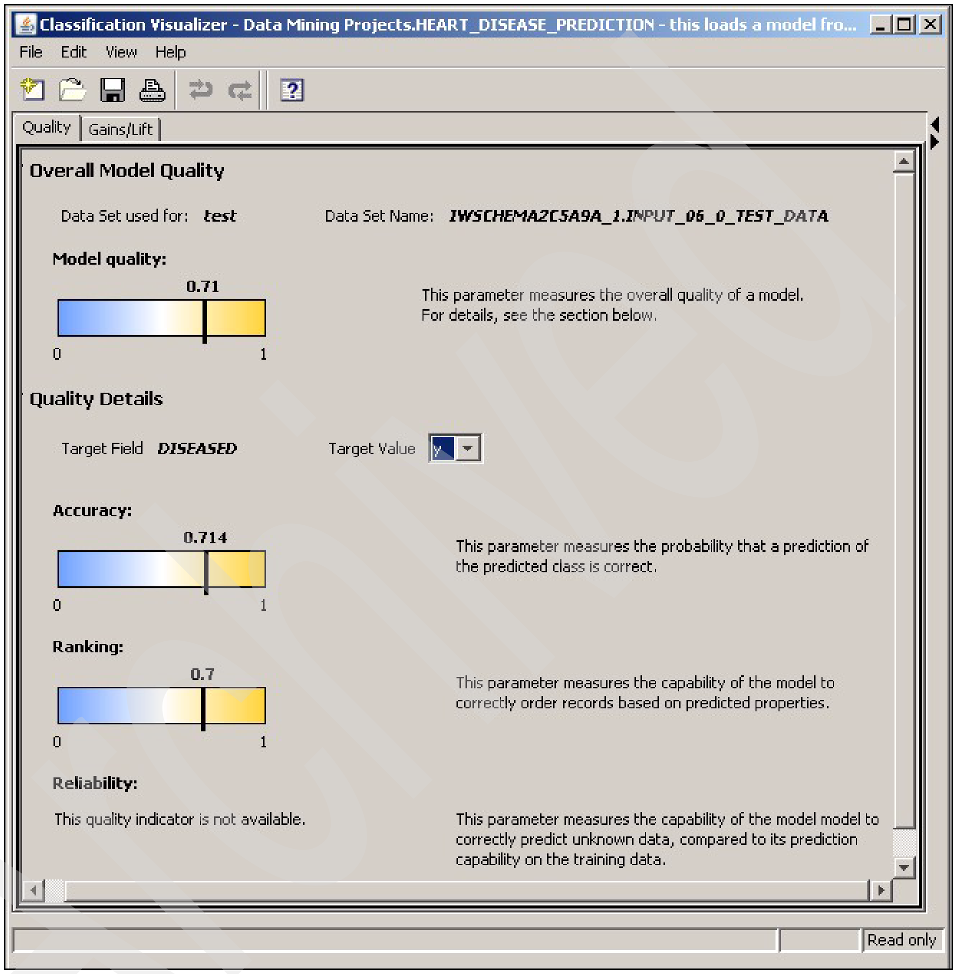


Figure 7-65 Classification visualization example: Field importance chart

Model quality measures are provided in the visualizations of both the trained model and the testing results. The quality based on the testing results, shown in Figure 7-66, is more relevant for how the model can be expected to perform with new data.


Classification Visualizer - Data Mining Projects.HEART_DISEASE_PREDICTION - this loads a model fro...
File
Edit
View
Help
Quality
Gains/Lift
Read only

Overall Model Quality

Data Set used for: *test* Data Set Name: *IWSCHEMA2C5A9A_1.INPUT_06_0_TEST_DATA*

Model quality: 0.71 (0 – 1)

This parameter measures the overall quality of a model. For details, see the section below.

Quality Details

Target Field *DISEASED* Target Value y

Accuracy: 0.714 (0 – 1)

This parameter measures the probability that a prediction of the predicted class is correct.

Ranking: 0.7 (0 – 1)

This parameter measures the capability of the model to correctly order records based on predicted properties.

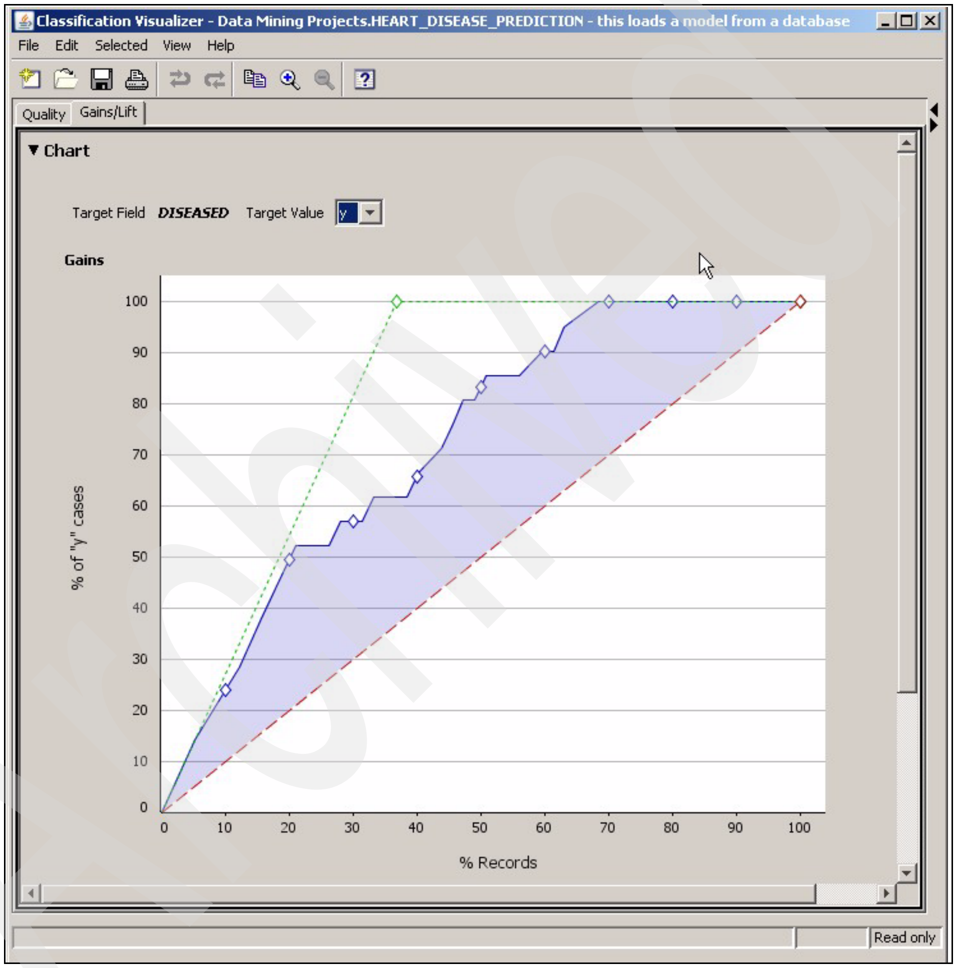
Reliability:

This quality indicator is not available.

This parameter measures the capability of the model model to correctly predict unknown data, compared to its prediction capability on the training data.

Figure 7-66 Classification visualization example: Model quality

The gains curve is a graphical representation of the model's performance. The gains curve based on the testing data is represented by the solid curve in Figure 7-67. The greater the degree of curvature of the gains curve relative to the straight, dashed line below the gains curve, the better the model's performance in correctly identifying patients with heart disease. (The light dashed line above the gains curve represents the upper boundary theoretically achievable by a model with perfect accuracy.)


Classification Visualizer - Data Mining Projects.HEART_DISEASE_PREDICTION - this loads a model from a database
File
Edit
Selected
View
Help
Quality
Gains/Lift
Chart
Gains
% of "y" cases
% Records
Read only

Target Field DISEASED Target Value y

Figure 7-67 Classification visualization example: Gains chart

Regression visualization

The regression visualizer presents information to evaluate a regression model and its testing result. Similar to the classification visualizer, the regression visualizer reports model quality, field importance, and a gains curve.

Time series visualizer

The time series visualizer displays a time series along with a forecast for a specified number of future time periods. A time series consists of numeric (continuous) data points for equally-spaced time periods. Up to three different algorithms can be used to generate the forecast for a time series. Multiple time series can be displayed together.

The forecast also comes with a confidence interval that is, by default, the 68% confidence interval (that is, the probability that the real value lies in this interval is 68%). The forecast overlaps with the unspecified part of the series that allows us to validate the quality of the forecast.

The Time Series mining function enables forecasting of time series values. However, which algorithm performs best in a given situation depends on the data. That is, how well the assumptions underlying each algorithm fit the actual situation. Similar to common regression methods, time series algorithms predict a numerical value. In contrast to common regression methods, time series predictions are focused on future values of an ordered series. These predictions are commonly called forecasts.

The time series algorithms are univariate algorithms. This means that the independent variable is a time column or an order column. The forecasts are based on past values, not on other independent columns. Time series algorithms are different from common regression algorithms because they not only predict future values but also incorporate seasonal cycles into the forecast. An example application is forecasting warehouse stock levels to optimize purchasing management.

The Time Series mining function provides algorithms that are based on different underlying model assumptions with several parameters. The learning algorithms try to find the best model and the best parameter values for the given data. If you do not specify a seasonal cycle, it is automatically determined. Also, missing values and non-equidistant time series are automatically interpolated.

The Time Series mining function in ISW provides the following algorithms to predict future trends:

- Autoregressive Integrated Moving Average (ARIMA)
- Exponential Smoothing
- Seasonal Trend Decomposition

Which of the algorithms creates the best forecast of your data depends on different model assumptions. You can calculate all forecasts at the same time. The algorithms calculate a detailed forecast including seasonal behavior of the original time series. With the Time Series Visualizer, you can evaluate and compare the resulting curves.

Autoregressive Integrated Moving Average (ARIMA)

The ARIMA algorithm also incorporates seasonal components. Therefore this algorithm is also referred to as Seasonal ARIMA (SARIMA).

The autoregressive part of the algorithm uses weighted previous values while the moving average part weighs the previously assumed errors of the time series. The ARIMA algorithm assumes the error to be independent and identically distributed from a normal distribution with zero mean. The basic ARIMA model works for stationary time series only. Stationary time series contain equal mean and equal variance for the whole time series. Therefore, the integrated part creates stationary series by differentiation.

Exponential smoothing

Exponential smoothing can consist of the following components:

- Basic level at a certain point in time
- Trend

  The trend can have additive or multiplicative characteristics. Also, it can be damped or non-damped.
- The seasonal component

Dependant on the data, trend and the seasonal component are optional.

There are ARIMA models that correspond to exponential smoothing models and vice versa.

Seasonal trend decomposition

Seasonal trend decomposition fits different seasonal trend functions to the given data and selects the best seasonal trend function according to an error measure. The following trends are used during the training run:

- Linear trend
- Quadratic trend
- Cubic trend
- Logarithmic trend
- Exponential trend
- Hyperbolic trend

The seasonality is incorporated in an additive or multiplicative way. For more information, see the following web pages:

- http://en.wikipedia.org/wiki/Forecasting#Time_series_methods
- http://en.wikipedia.org/wiki/Decomposition_of_time_series

This is an important technique for all types of time series analysis, especially for seasonal adjustment. It seeks to construct, from an observed time series, a number of component series (that could be used to reconstruct the original by additions or multiplications) where each of these has a certain characteristic or type of behavior. For example, monthly or quarterly economic time series are usually decomposed into the following components:

- Trend Component (Tt) that reflects the long term progression of the series (secular variation)
- Cyclical Component (Ct) that describes repeated but non-periodic fluctuations, possibly caused by the economic cycle
- Seasonal Component (St) reflecting seasonality (Seasonal variation)
- Irregular Component (It) that describes random, irregular influences (noise). Compared to the other components it represents the residuals of the time series.

The forecast also comes with a confidence interval that is, by default, the 68% confidence interval (that is, the probability that the real value lies in this interval is 68%). The forecast overlaps with the unspecified part of the series that allows us to validate the quality of the forecast, as shown in Figure 7-68.

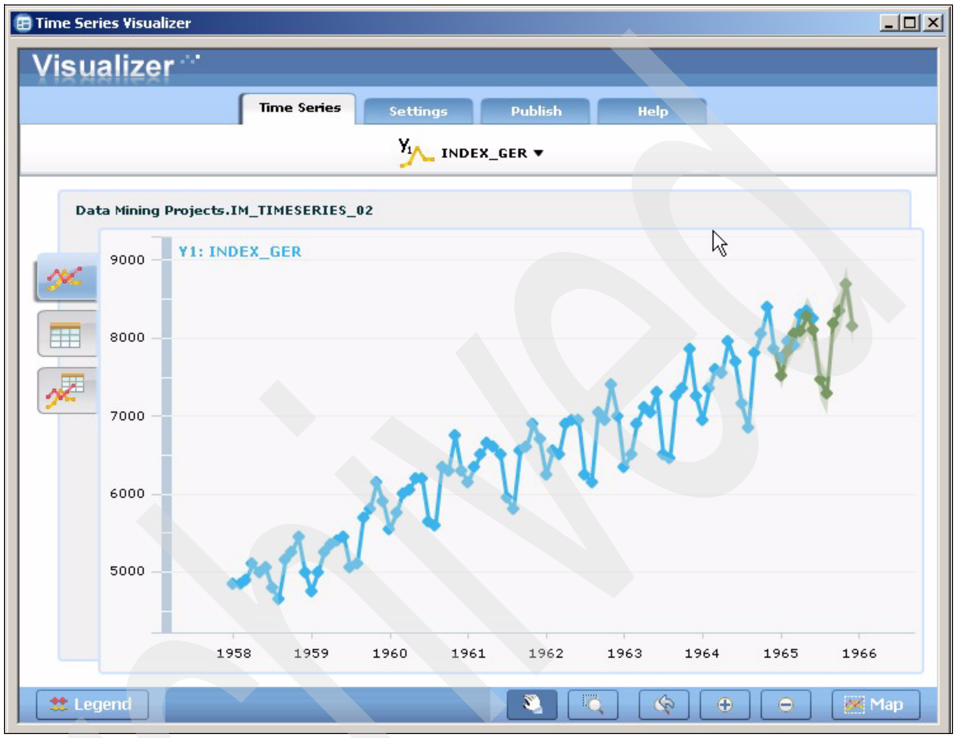


Figure 7-68 Time series visualizer

7.5.10 Scoring with other PMML models

Data mining models created by tools other than InfoSphere Warehouse Modeling can be used in mining flows to score new data. For example, an analyst might create a data mining model using a statistical tool capable of exporting data mining models in PMML format. This model can be exported from the statistical tool, imported into a DB2 table in Design Studio, and incorporated into a mining flow for scoring. This capability is useful for making use of high-performance InfoSphere Warehouse Scoring with data mining models not generated by InfoSphere Warehouse Modeling.

Once a data mining model has been created and exported in PMML format to a file system, it is imported into a DB2 database using the Data Source Explorer. The folder for the appropriate model type is selected under the Data Mining Models folder. For example, a logistical regression model is imported into the Classification folder, as shown in Figure 7-69.

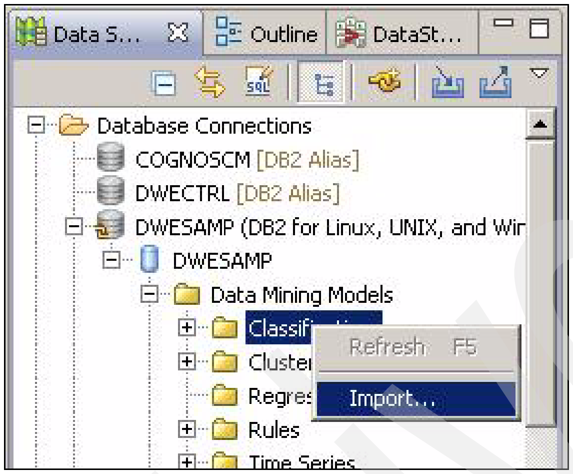
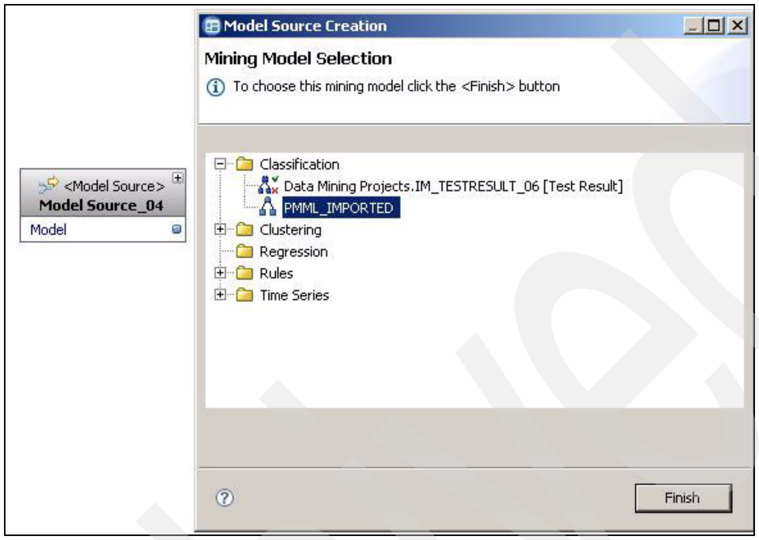


Figure 7-69 Importing a PMML model into the database

The PMML model is specified in a model source operator by creating a model source operator from the palette, as shown in Figure 7-70. The operator can also be created by dragging the PMML model onto the canvas from the Data Source Explorer.


Model Source Creation

Mining Model Selection

To choose this mining model click the <Finish> button

Figure 7-70 Selecting a PMML model in a model source operator

The mining flow containing the PMML model as the model source is illustrated in Figure 7-71. When the mining flow is executed, the scoring results are stored in the output table specified in the table target operator.

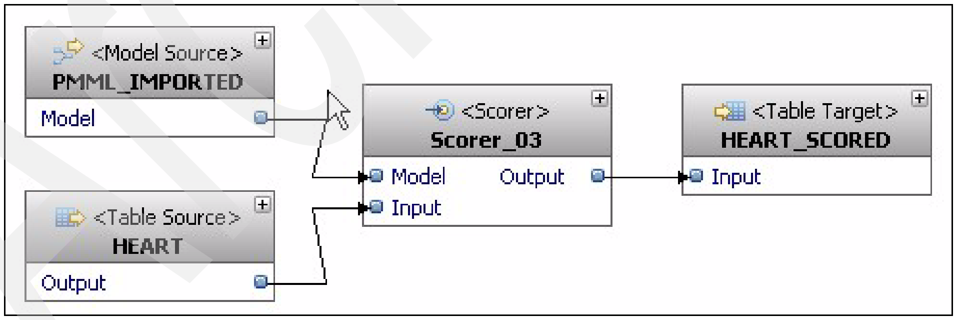


Figure 7-71 Mining flow with a model source operator for scoring

7.5.11 Managing data mining models

Data mining models residing in DB2 tables under the schema IDMMX can be opened for viewing, renamed, deleted, or exported in PMML format through the Data Source Explorer, as shown in Figure 7-72.

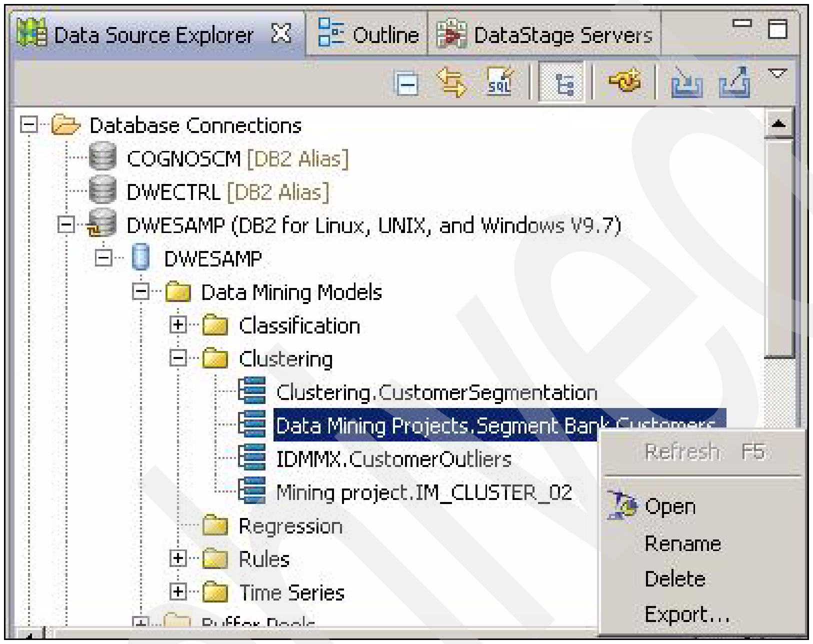
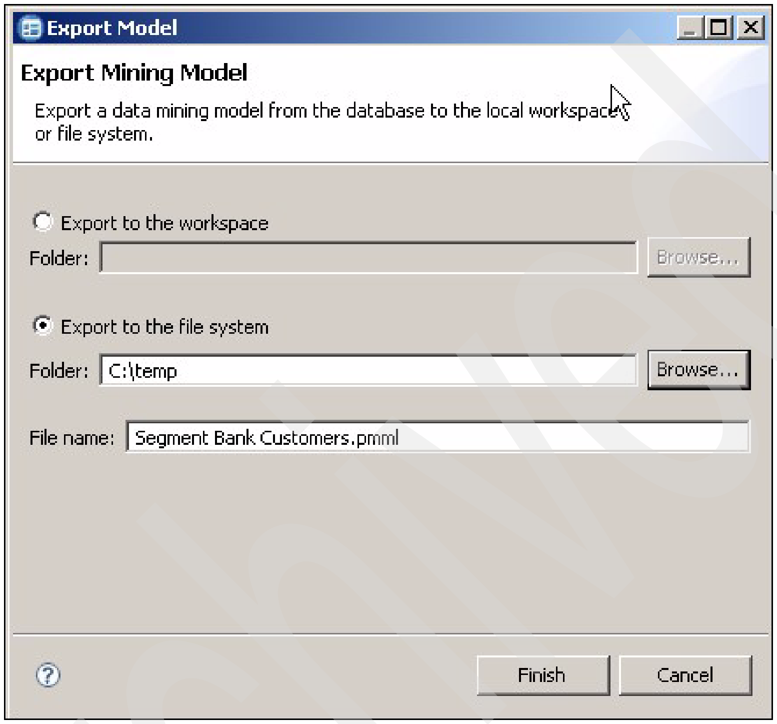


Figure 7-72 Managing data mining models

As illustrated in Figure 7-73, a model can be exported to another workspace or to the file system in PMML format. In addition, a model can be dragged from the Data Source Explorer and placed onto a mining flow canvas as a Model Source operator.


Export Model

Export Mining Model

Export a data mining model from the database to the local workspace or file system.

Export to the workspace [ ]
Folder:

Export to the file system [✓]
Folder: C:\temp

File name: Segment Bank Customers.pmml

Figure 7-73 Exporting a data mining model

8

Text analytics

Unstructured information represents the largest, most current, and fastest growing source of information that is available today. This information exists in many different sources such as call center records, repair reports, product reviews, emails, and more. The text analysis features of InfoSphere Warehouse can help to uncover the hidden value in this unstructured data. In this chapter we discuss the general architecture and business opportunities of analyzing unstructured data with the text analysis capabilities of InfoSphere Warehouse.

8.1 Text analytics overview

With InfoSphere Warehouse, users can create business insight from unstructured information. Information can be extracted from text columns in the data warehouse (for example call center notes, free-text survey-fields, repair reports, patient records). This extracted information can be used in reports, multidimensional analysis, or as input for data mining.

Unstructured data must be analyzed to interpret, detect, and locate concepts of interest that are not explicitly tagged or annotated in the original document. For example, documents might include the following domain-specific information:

- Named entities, such as persons, organizations, facilities, and products
- Opinions, complaints, threats, or facts
- Relations that are located in finances, purchases, or repairs

The text analysis functions of InfoSphere Warehouse focus on information extraction to create structured data that can be analyzed with InfoSphere Warehouse tools.

8.2 Business goals and business examples

Text data is usually grouped in a category, such as unstructured data, together with video tapes, images, audio tapes, and other data that is stored with little or no subdivisions at all into different fields. Though you can also apply the concepts of the Unstructured Information Management Architecture (UIMA) to audio tapes or video tapes, unstructured information in InfoSphere Warehouse typically refers to text strings such as call-center notes, customer-satisfaction surveys, or problem reports.

With the existing data warehouse tools, you cannot efficiently use this data to create insight. While there is huge insight in structured data, that insight is typically lost because it is difficult to identify in the vast amount of unstructured data that is used by organizations to run their business.

The goal of text analysis is to transform unstructured data into a structure that can be analyzed in the InfoSphere Warehouse, together with existing structured data by using data warehousing tools, such as reporting tools, tools for multidimensional analysis or data mining tools. You can create this structure by determining concepts that are included in the text, and by extracting these concepts into relational tables.

8.2.1 Scenarios for text analysis

The primary scenarios for text analysis are similar to the following examples:

- Creating simple reports

  You can want to identify the top 10 customer satisfaction problems that are recorded in a text field of a customer survey.

- Creating multidimensional reports

  In retail, you can want to derive the new Return reasons OLAP dimension from text analysis and combine it with existing dimensions such as time, geography, or product.

- Generating additional input fields for data mining to improve the predictive power of data mining models

  You might want to use symptoms that are included in patient records to improve the predictive power of a data mining model. That model can predict the patients who need treatment based only on available structured data, such as patient age or blood pressure.

8.2.2 Business examples

You can apply text analysis or a combined analysis of structured and unstructured data in all industries and across many business functions. The following examples illustrate the concepts.

Cross-business analysis

In this example we look at analyzing customer satisfaction surveys. Typically, customer satisfaction surveys include structured fields and text fields. As examples, structured fields might include simple Yes or No questions, multiple-choice questions, or questions requiring a rating - such as one that uses a scale that ranges from 1 to 5. Above the structured fields, surveys usually include text fields that allow customers to express their views in their own words. These comments are of high value. However, processing these comments can be tedious, time-consuming, and costly to categorize or identify. You can streamline this process by using text analysis. The process would follow these steps:

1. Find the most frequently occurring terms or concepts in free-text fields.
2. Categorize these terms into broader categories.
3. Create simple reports, such as determining the top 10 problems, or multidimensional reports that identify trends of specific topics over time.

Customer retention analysis

Customer retention is important in any business, because it is expensive to gain new customers. For example, in the telecommunication industry, a cell-phone provider might want to combine text analysis with the data mining technique of predictive modeling to reinforce customer retention. Therefore, customers must not be attracted by competitors. With the data mining technique of predictive modeling, you can predict a customer's propensity to cancel their contract. Predictive modeling is based on available data about each customer and on historical data of customers who have left your company.

In a traditional data mining model, only structured data is used. This includes the following data as examples:

- Demographic data, including age, gender, income, and number of children.
- Transactional data, including payment type, number of overseas calls, and number of long-distance calls.

With text analysis, you can extract the most important concepts that are mentioned by customers during a call to the call center, such as a new phone or a bad phone rate). By using the data that can be extracted from call-center notes as additional input into the data mining prediction model, you can improve the predictive power of the data mining model.

Causal analysis of repair reports

A new car manufacturer might want to analyze repair reports from repair shops to understand the root or the repair sequences that cause frequent failures. The business goals are to improve failing parts and to receive early warning indicators to avoid costly product recalls.

Repair shops that are accredited dealers for a specific brand might provide most of the repair reports in a structured form, such as part types or standard codes for standard services. This data is already used today to find sequential patterns of part-failure sequences. However, non-standard cases are reported in free-text fields. Extracting part types and problem types from these free-text fields and including them in the causal and sequence analysis improves the manufacturers ability to react to detect quality problems earlier and avoid costly product recalls.

Causal analysis for coronary heart disease

Medical scientists might want to evaluate a study on the risk of patients who suffer from heart diseases. For each patient, structured data such as blood pressure, cholesterol, or age, are collected. Additionally, unstructured textual data for each patient was collected. The unstructured data includes the medical history with data about the lifestyle and medical symptoms of the patient. This data might include relevant risk factors for heart diseases.

With text analysis, the following keywords might be detected in the records of the patients:

- Smoker
- Physical inactivity
- Alcoholism
- Obesity

You can improve association models and classification models by including the keywords that are retrieved by using text analysis on these models.

8.3 Text analysis in InfoSphere Warehouse

In this section we focus on the text analytics capabilities of InfoSphere Warehouse.

8.3.1 Unstructured analytics in InfoSphere Warehouse

InfoSphere Warehouse uses UIMA for the analysis of unstructured data. UIMA is an open, scalable, and extensible platform for creating, integrating, and deploying text-analysis solutions. UIMA-based components that are used to extract entities such as names, sentiments, or relationships are called UIMA Annotators or Analysis Engines.

InfoSphere Warehouse provides operators and tooling for dictionary-based and regular expression-based named entity recognition. For other text analysis tasks, a generic text analysis operator is available that can be used to run Apache UIMA-compatible annotators in analytic flows:

- Data understanding

  Data understanding is important for successful information extraction from text data, so InfoSphere Warehouse provides the Data Exploration feature to find columns with relevant text information (Text Statistics view) and to browse through the text (Sample Contents view). For a deeper analysis, you can use the Frequent Terms Extraction feature to extract the most frequent terms that occur in the text column together with advanced visualization, like a cloud view. Frequent Terms Extraction is an important feature for the efficient creation of dictionaries that can be used in dictionary-based analysis.

- Dictionary-based analysis

  Dictionary-based analysis is the extraction of keywords from text. Examples of entities that you can extract are names, companies, and products. You can also extract all entities that are contained in a list. InfoSphere Warehouse supports dictionary-based analysis of text columns through the Dictionary Lookup operator. The Dictionary Lookup operator is based on technology from IBM LanguageWare®. It supports natural language processing such as word stem reduction and tokenization in multiple languages. Dictionaries can be created and maintained with the Dictionary Editor in InfoSphere Warehouse. InfoSphere Warehouse also includes a taxonomy editor that categorizes dictionary entries in a taxonomy tree for use in data mining and OLAP.

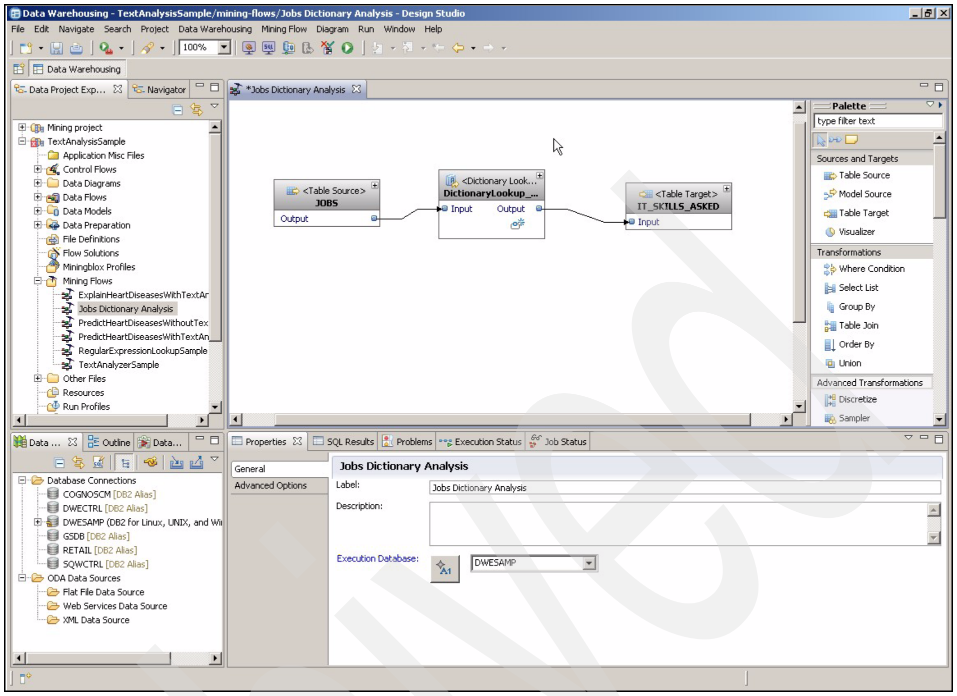
- Rule-based analysis

  Rule-based analysis is the extraction of information from text through regular expression rules. Regular expressions are ideal to extract concepts such as phone or credit card numbers, addresses, and dates. InfoSphere Warehouse supports rule-based analysis through the Regular Expression Lookup operator. The operator uses rule files containing regular expression rules to extract concepts from text columns. You can create and modify these rule files with the Regular Expressions editor.

In addition to the common text analysis methods above, InfoSphere Warehouse allows the use of Apache UIMA compatible annotators. You can import these into a Data Warehousing project and use them in the Text Analyzer operator.

8.3.2 Creating text analysis flows in Design Studio

Design Studio (Figure 8-1 on page 335) is the integrated tooling platform of InfoSphere Warehouse. The default project for all data warehouse work is the Data Warehousing Project. This project contains a text analysis folder that contains the text analytic resources (such as dictionaries, rule files, and taxonomies).



Label: Jobs Dictionary Analysis

Execution Database: DWESAMP

Figure 8-1 InfoSphere Warehouse Design Studio

The information extraction is done by the text operators in the data transformation flows (data flows and mining flows). With these flows, you can sample, join, and modify tables. Text operators can extract structured information from text columns and add them to the output as new columns containing found concepts (such as names, skills, and dates).

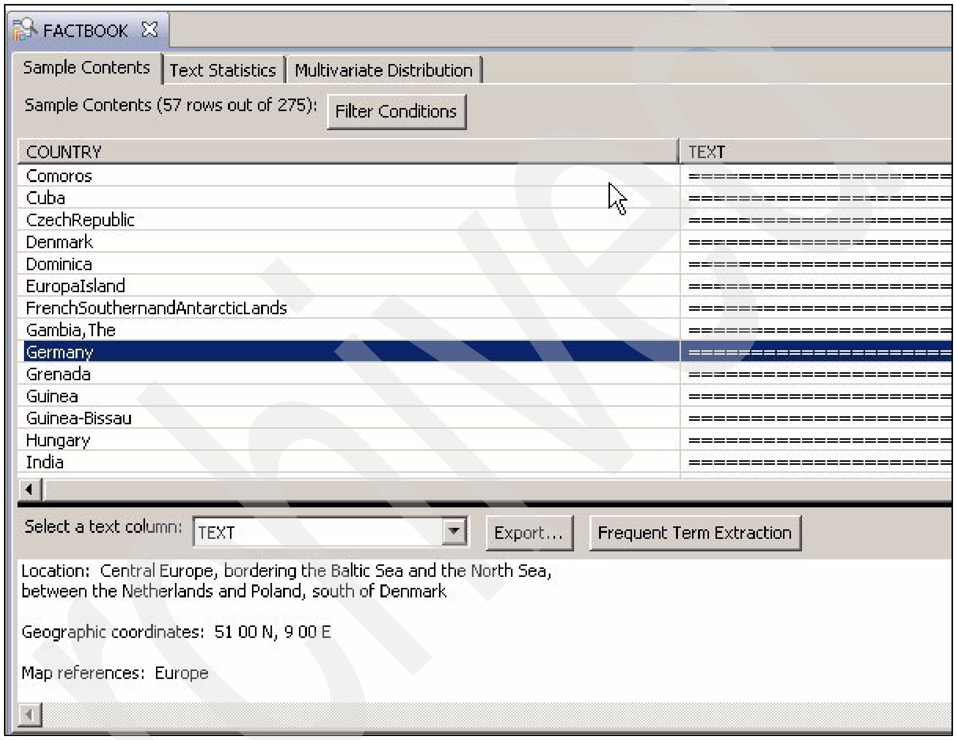
The remainder of this section focuses on examples of the text mining capabilities available in InfoSphere Warehouse.

Exploring the unstructured data

To perform an analysis of text in database tables, first explore the available data to select the relevant text columns for the analysis. With the InfoSphere Warehouse Data Exploration view you can browse samples of database tables, browse the contents of large text columns, and determine the average length of text strings.

To open the Data Exploration view, navigate to the FACTBOOK table in the DWESAMP database and CIA schema. Right-click the table and select **Distribution and Statistics** → **Data Exploration** from the context menu.

The Data Exploration view shown in Figure 8-2 shows a sample of fifty random rows out of the 275 rows in the table. Each row contains the country name and a column named TEXT with information about the country. Below the table, you can select one of the text columns in the table to display the complete content.



Sample Contents (57 rows out of 275):

COUNTRY	TEXT
Comoros	====================
Cuba	====================
CzechRepublic	====================
Denmark	====================
Dominica	====================
EuropaIsland	====================
FrenchSouthernandAntarcticLands	====================
Gambia,The	====================
Germany	====================
Grenada	====================
Guinea	====================
Guinea-Bissau	====================
Hungary	====================
India	====================

Location: Central Europe, bordering the Baltic Sea and the North Sea, between the Netherlands and Poland, south of Denmark

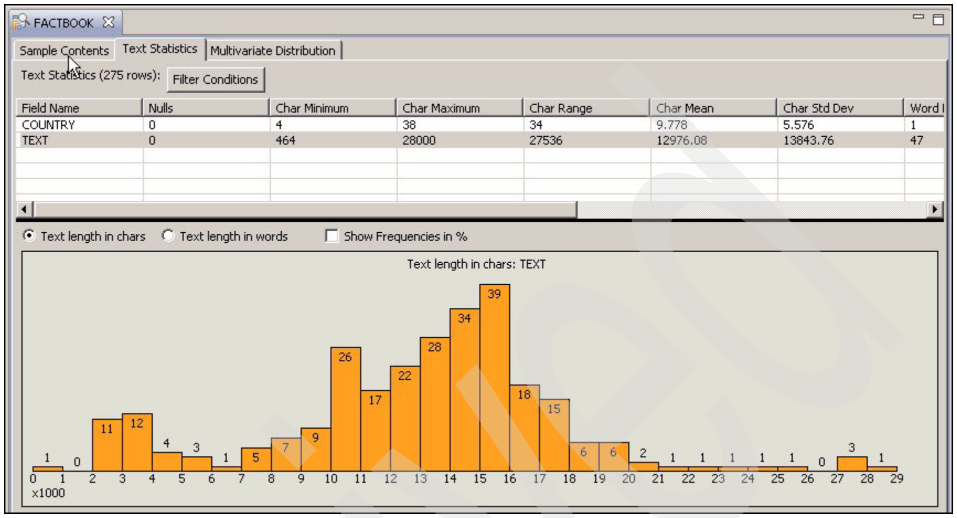
Geographic coordinates: 51 00 N, 9 00 E

Map references: Europe

Figure 8-2 Data Exploration view - sample contents

To view information about a country, select the appropriate row in the sample contents table. The text description of the country contains information about its geographic location and its area in square kilometers. A country's geographic coordinates and area are always given in the same format. The task of extracting concepts in this format can be easily handled with regular expression rules.

The text statistics tab shows information about the size of the text fields, including maximum and minimum number of characters, and character range. This view is illustrated in Figure 8-3.



Field Name	Nulls	Char Minimum	Char Maximum	Char Range	Char Mean	Char Std Dev	Word I
COUNTRY	0	4	38	34	9.778	5.576	1
TEXT	0	464	28000	27536	12976.08	13843.76	47

Figure 8-3 Data Exploration view: Text statistics

Creating a rule file with regular expressions

With rule files, you can define concepts such as phone numbers or uniform resource locators. These concepts are called *types*. To find these concepts in text, you can specify rules that define a pattern to match these concepts.

A concept type can have *features*. For example, given a phone number that consists of a country code, an area code, and the extension number, you can create the type *phone number* and specify the features *country code*, *area code*, and *extension number*.

With the RegEx rules editor, you define concept types and their features and assign rules with regular expression patterns to those types. When a pattern matches a part of the text, an annotation is created for the associated type. You can set the feature values of an annotation by assigning a sub-pattern of the regular expression rule - a match group - to the feature.

When the rule file is used in the Regular Expression Lookup operator in a mining flow or a data flow, the features can be mapped to columns of a relational table, which denotes the extracted concepts.

To create a rule file, perform the following steps:

1. Select **File** → **New** → **Data Warehousing Project**. Give the project a name and click **Finish**.
2. In the Text Analysis folder, right-click the Rules folder and select **New** → **Rule File** from the context menu, as shown in Figure 8-4. The New Rules dialog displays.

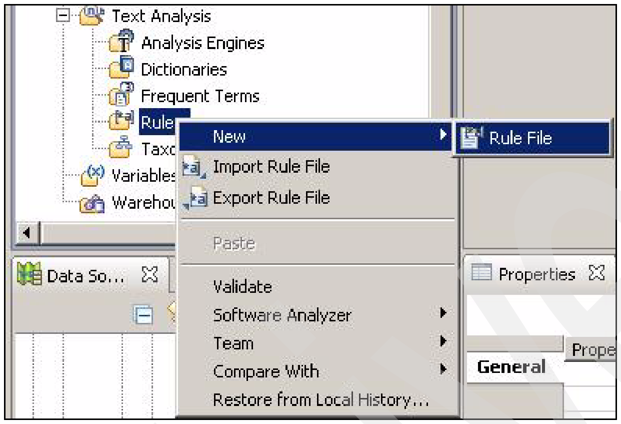
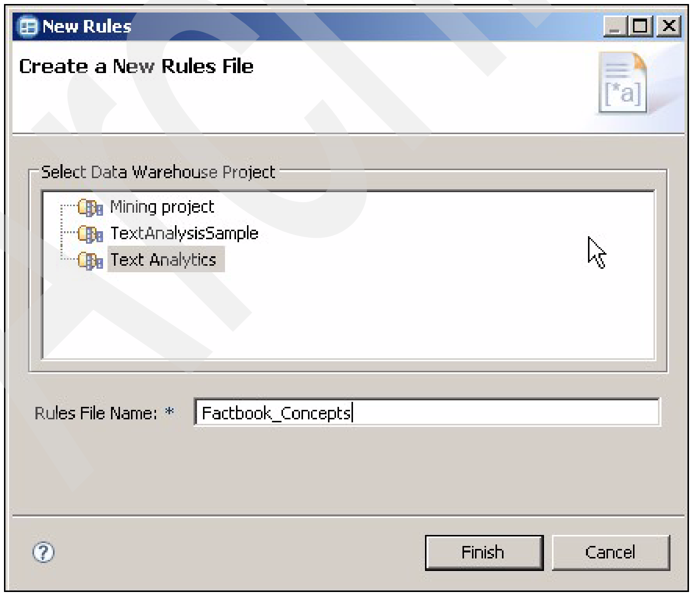


Figure 8-4 Creating a new rule file

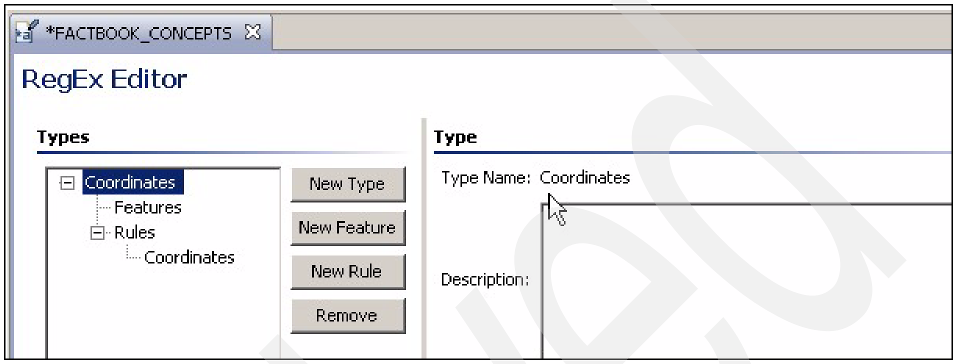
3. Select the Data Warehousing Project you created in step 1. Specify `Factbook_Concepts` as the rule file name, as in Figure 8-5.



Rules File Name: * Factbook_Concepts

Figure 8-5 Rules file name

4. Click **Finish**. This displays the RegEx editor.
5. In the Types section, "Factbook_Concepts" is displayed as the initial type. Delete this type and create a new type. Name the new type `Coordinates`. This automatically creates a new rule with the same name, as shown in Figure 8-6. No regular expression is defined yet.



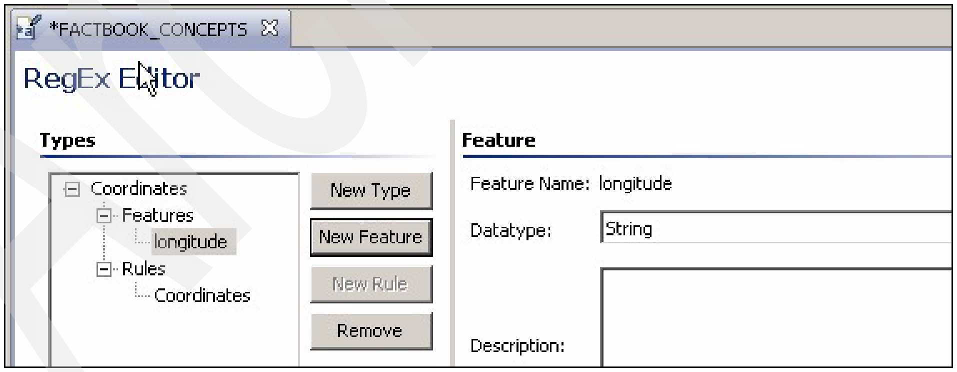
Type Name: Coordinates

Description:

Figure 8-6 Regular Expression editor

The first rule to be created should extract the concept type Coordinates with features longitude and latitude from the CIA.FACTBOOK table.

6. In the RegEx editor, for the Coordinates type, select the Features folder and click **New Feature**.
7. In the New Feature dialog, type `longitude` in the entry field, and click **OK**. Accept the default type of **String**, as shown in Figure 8-7.



Feature Name: longitude

Datatype: String

Description:

Figure 8-7 New feature

8. Repeat the step 7 to add another feature named `latitude` with the data type **String**.

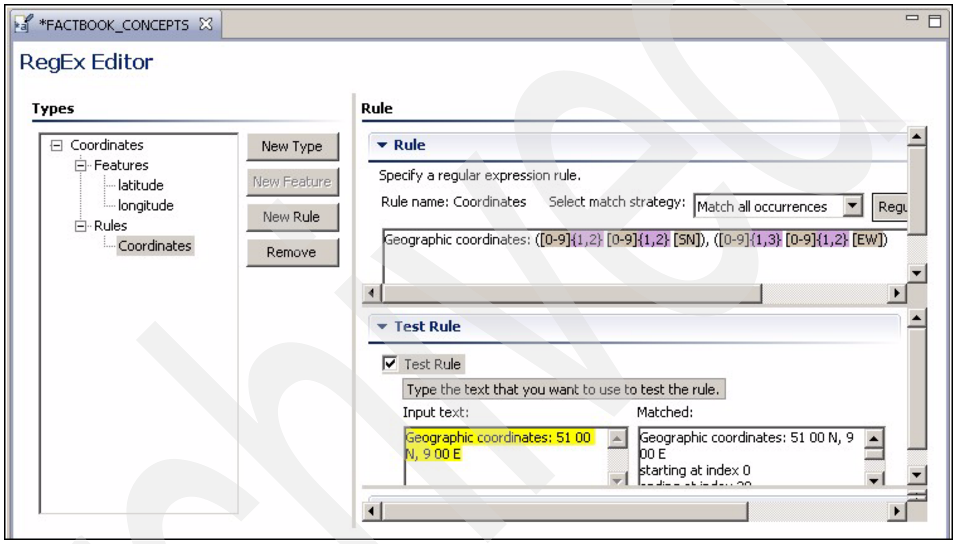
9. Specify the regular expression pattern for the rule:

   a. Expand the Rules folder in the tree and select the Coordinates rule.

      The Rule section of the RegEx editor, shown in Figure 8-8, is where the regular expression is entered.

   b. Enter the following regular expression pattern into the input field of the Rule section:

      Geographic coordinates: ([0-9]{1,2} [0-9]{1,2} [SN]), ([0-9]{1,3} [0-9]{1,2} [EW])



Specify a regular expression rule.

Rule name: Coordinates Select match strategy: Match all occurrences

```
Geographic coordinates: ([0-9]{1,2} [0-9]{1,2} [SN]), ([0-9]{1,3} [0-9]{1,2} [EW])
```

Type the text that you want to use to test the rule.

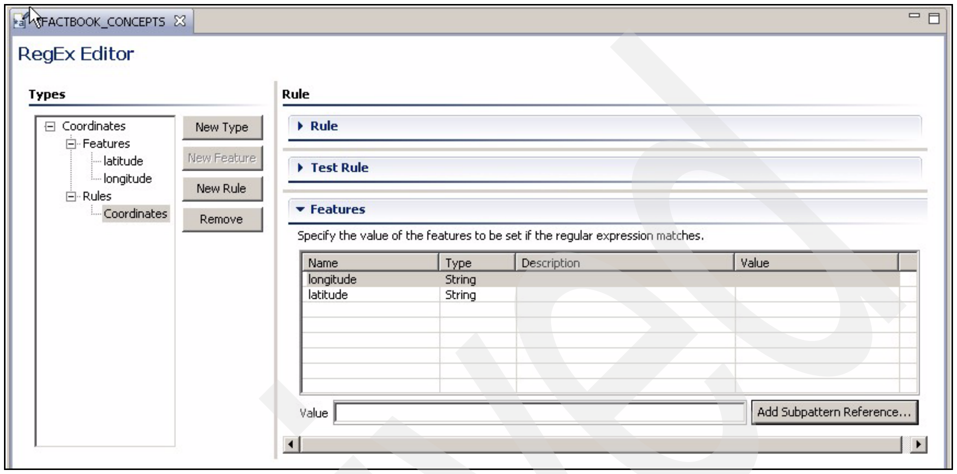
Input text: Geographic coordinates: 51 00 N, 9 00 E

Matched: Geographic coordinates: 51 00 N, 9 00 E starting at index 0

Figure 8-8 New regular expression

The above rule specifies 1–2 digits, followed by a blank, followed by 1–2 digits, followed by S or N (for south or north), followed by a comma, followed by 1–3 digits, followed by a blank, followed by 1–2 digits, followed by E or W (for east or west). Alternatively, you can build the regular expression pattern yourself using the Regular Expression Builder. The builder provides syntax assistance for the regular expression notation.

c. Assign sub-patterns to the features longitude and latitude so that these concepts can be extracted separately. In the Features section, select the entry for latitude and click **Add subpattern reference**, as shown in Figure 8-9.

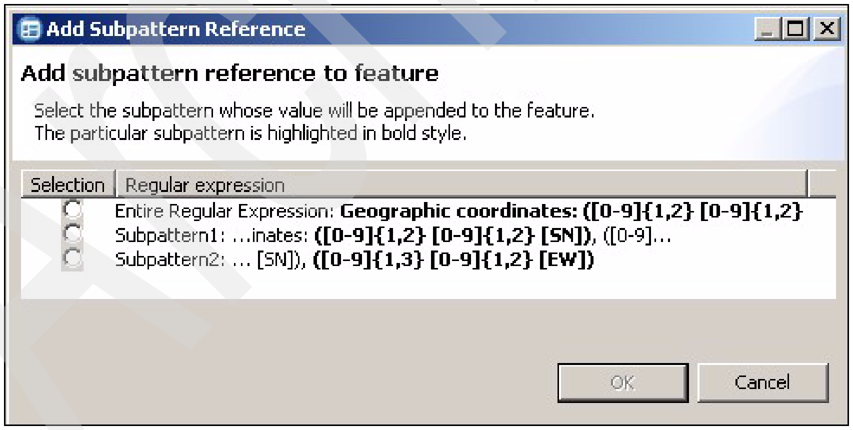


Specify the value of the features to be set if the regular expression matches.

Name	Type	Description	Value
longitude	String		
latitude	String		

Figure 8-9 Add subpattern reference

d. The “Add Subpattern Reference” dialog box opens as illustrated in Figure 8-10. For each choice listed on the dialog, the part of the regular expression that defines the sub-pattern is displayed in bold.



Add subpattern reference to feature

Select the subpattern whose value will be appended to the feature.
The particular subpattern is highlighted in bold style.

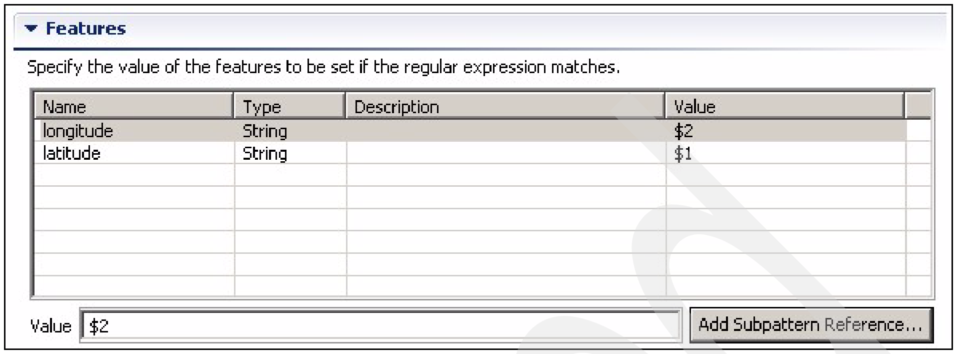
Selection	Regular expression
○	Entire Regular Expression: **Geographic coordinates: ([0-9]{1,2} [0-9]{1,2}**
○	Subpattern1: ...inates: **([0-9]{1,2} [0-9]{1,2} [SN])**, ([0-9]...
○	Subpattern2: ... [SN]), **([0-9]{1,3} [0-9]{1,2} [EW])**

Figure 8-10 Add subpattern reference dialog

e. Select **Subpattern1** and click **OK**.

f. Go back to the features section, select the entry for longitude and click **Add Subpattern reference**. From the dialog box, select **Subpattern2**.

As shown in Figure 8-11, the Features section should now show the assigned sub-pattern for each feature.



Specify the value of the features to be set if the regular expression matches.

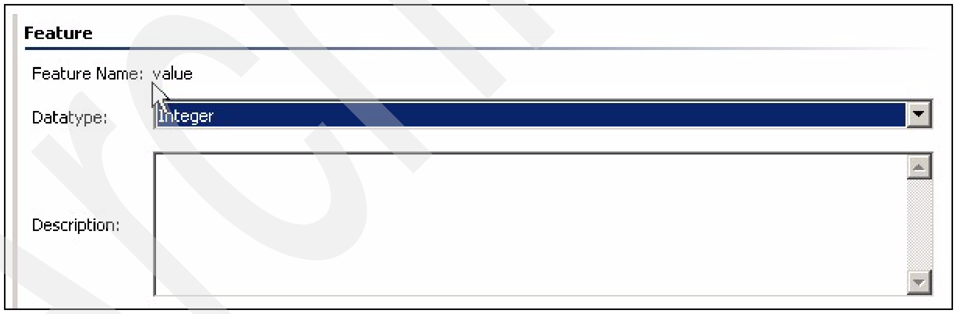
Name	Type	Description	Value
longitude	String		$2
latitude	String		$1

Value: $2

Figure 8-11 Assigned sub-pattern references

The second rule to create will extract the concept type Area.

1. Click **New Type** and enter `Area` as the type name. This creates a rule with the name Area.
2. Click **New Feature** and enter `value` as the feature name.
3. Change the data type of the new feature named value to `Integer`, as shown in Figure 8-12.



Feature Name: value

Datatype: Integer

Description:

Figure 8-12 Data type for feature

4. Select **Area** under the Rules folder to create a new regular expression. Enter the following expression:

   Area: total: ([0-9]{1,3}(,[0-9]{1,3})*) sq km
5. Assign Subpattern1 to the feature named value.
6. Save the rule file by clicking in the editor area and pressing Ctrl+S.

Creating a flow with a regular expression lookup operator

The Regular Expression Lookup operator is based on rule files. The rule files contain regular expression patterns that let you extract concepts such as phone numbers or email addresses from database tables. With the regular expression lookup operator you can find the text sections that match the expressions contained in the selected rule files.

To create a mining flow, perform the following steps:

1. Right-click the Mining Flows folder in your Data Warehousing project and select **New** → **Mining Flow** from the context menu.
2. In the wizard, type a name for the mining flow. Make the selection to work against a database and click **Next**.
3. From the Select Connection page, select the **DWESAMP** database and click **Finish**. This opens the mining flow editor.
4. Drag a table source operator to the canvas, and select the **CIA.FACTBOOK** table. Click **Finish**.
5. From the Text Operators section, drag a regular expression lookup operator onto the canvas. Connect the output port of the table source operator with the input port of the regular expression operator, shown in Figure 8-13.

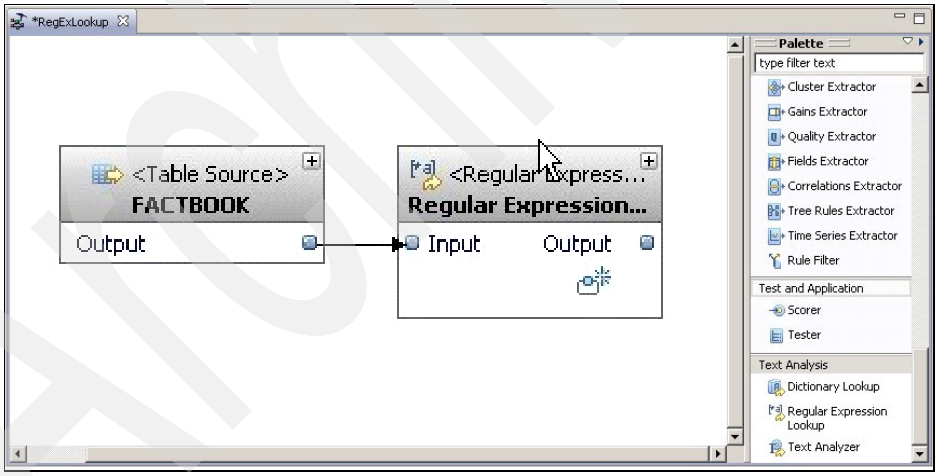


Figure 8-13 Mining flow with a regular expression lookup operator

6. On the settings tab of the Properties view, select the input text column **TEXT** from the list of available columns.

7. On the Analysis Results tab, click the **Add new port** icon (shown in Figure 8-14) to add two new output ports.

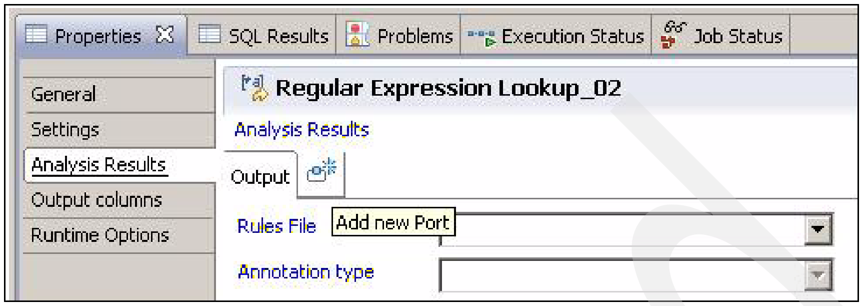
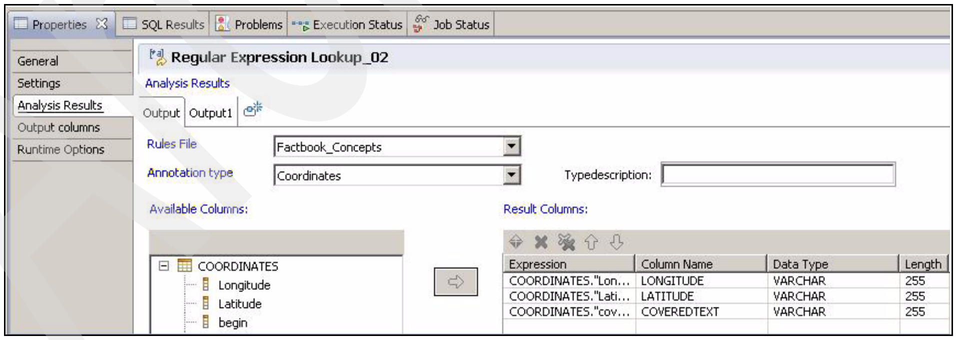


Figure 8-14 Adding new output ports

8. For the 1st output port (output), perform the following steps:
   a. Select the Output tab on the Analysis Results tab.
   b. Select **Factbook_Concepts** as the rule file to be used in the text analysis.
   c. Select **Coordinates** from the drop down list of annotation types.
   d. Delete the columns named being and end from the Result Columns table.
9. For the 2nd output port (output1), perform the following steps:
   a. Select the Output1 tab on the Analysis Results tab.
   b. Select **Factbook_Concepts** as the rule file to be used in the text analysis.
   c. Select **Area** from the drop down list of annotation types.
   d. Delete the columns named begin and end from the Result Columns table.

Figure 8-15 shows the Analysis Results tab.



Rules File: Factbook_Concepts

Annotation type: Coordinates

Typedescription:

Expression	Column Name	Data Type	Length
COORDINATES."Lon...	LONGITUDE	VARCHAR	255
COORDINATES."Lati...	LATITUDE	VARCHAR	255
COORDINATES."cov...	COVEREDTEXT	VARCHAR	255

Figure 8-15 Analysis Results tab

10. On the Output Columns Tab of the Properties view, select the COUNTRY column from the list of available columns and move it to the list of output columns on the right. Now you can relate the extracted concepts with the country key, because this column is also contained in the operator output.

11. Create the tables that receive the analysis results by performing the following steps:

    a. Right-click the first output port of the regular expression lookup operator and select **Create a suitable table** from the context menu. Enter `COUNTRY_COORDINATES` for the table name, `CIA` for the schema, and click **Finish**.

    b. Right click the second output port of the operator and select **Create a suitable table** from the context menu. Enter `COUNTRY_NAME` for the table, `CIA` for the schema, and click **Finish**.

12. Click in the editor area and save the mining flow.

Figure 8-16 shows the complete mining flow.

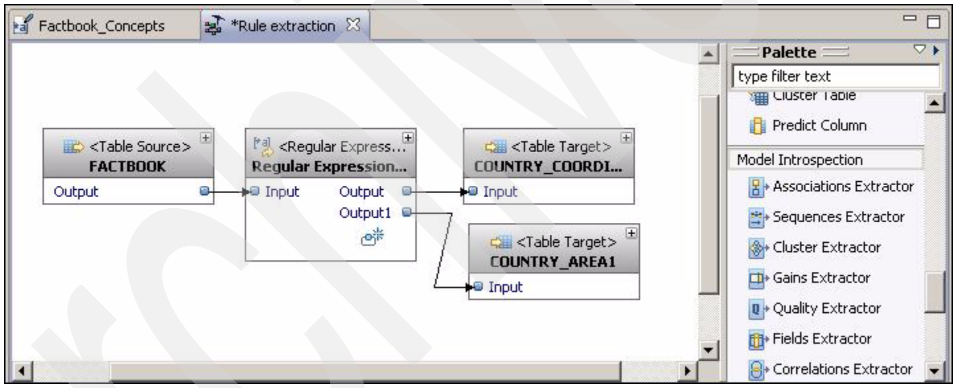
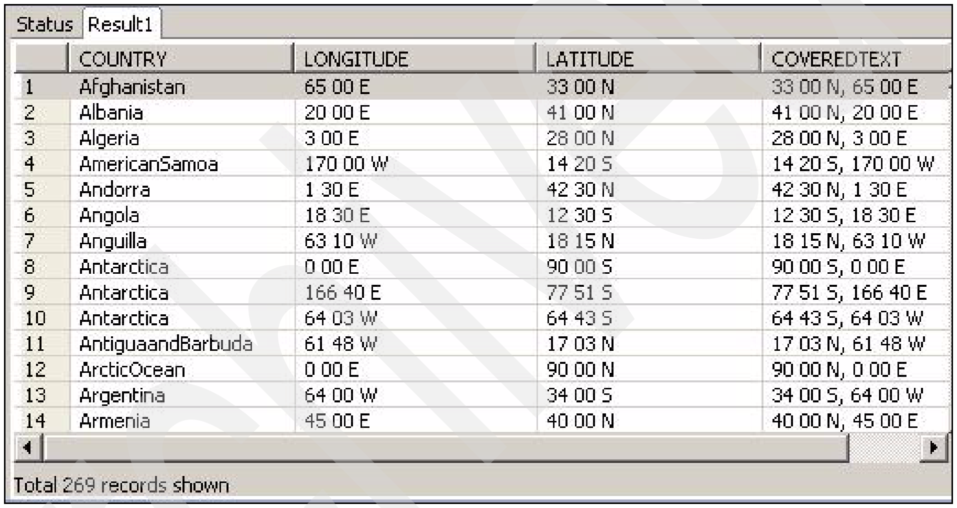


Figure 8-16 Complete mining flow

Executing the mining flow

To execute the mining flow, perform the following steps:

1. To execute the mining flow, select **Mining Flow** → **Execute** from the menu and click **Execute** on the wizard. The execution process analyzes the source columns in table CIA.FACTBOOK using your rules file and writes the result into the COUNTRY_COORDINATES and COUNTRY_AREA tables you just created.
2. After executing the mining flow, you can explore the content of the target tables. Right-click the COUNTRY_COORDINATES table and select **Sample contents of database table** from the context menu. This causes the sample contents to be shown (as shown in Figure 8-17).



	COUNTRY	LONGITUDE	LATITUDE	COVEREDTEXT
1	Afghanistan	65 00 E	33 00 N	33 00 N, 65 00 E
2	Albania	20 00 E	41 00 N	41 00 N, 20 00 E
3	Algeria	3 00 E	28 00 N	28 00 N, 3 00 E
4	AmericanSamoa	170 00 W	14 20 S	14 20 S, 170 00 W
5	Andorra	1 30 E	42 30 N	42 30 N, 1 30 E
6	Angola	18 30 E	12 30 S	12 30 S, 18 30 E
7	Anguilla	63 10 W	18 15 N	18 15 N, 63 10 W
8	Antarctica	0 00 E	90 00 S	90 00 S, 0 00 E
9	Antarctica	166 40 E	77 51 S	77 51 S, 166 40 E
10	Antarctica	64 03 W	64 43 S	64 43 S, 64 03 W
11	AntiguaandBarbuda	61 48 W	17 03 N	17 03 N, 61 48 W
12	ArcticOcean	0 00 E	90 00 N	90 00 N, 0 00 E
13	Argentina	64 00 W	34 00 S	34 00 S, 64 00 W
14	Armenia	45 00 E	40 00 N	40 00 N, 45 00 E

Total 269 records shown

Figure 8-17 Sample contents of target table

Dictionary-based concept extraction

InfoSphere Warehouse contains rich tooling to extract concepts from text based on dictionaries. With the help of a dictionary editor, you can build a dictionary and use the dictionary lookup operator to embed the concept extraction in a flow. InfoSphere Warehouse dictionaries are much more than simple word lists. A dictionary entry consists of a base form and multiple optional variants. The variants can include acronyms, abbreviations, synonyms, or any other variation or association of the base form.

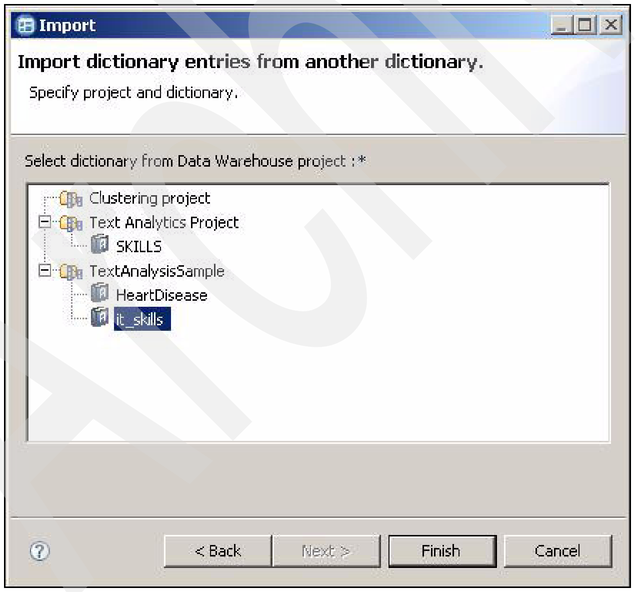
Dictionaries can be created manually, or the entries can be imported from flat files, database tables, or other dictionaries. They can also be created from the results of a frequent term extraction. This extraction shows the most frequent nouns, verbs, or any specific part-of-speech patterns. In the results view, you can mark all interesting words and export them to a dictionary.

Creating a dictionary

To create a dictionary, perform the following steps:

1. in your Data Warehousing project, right-click the Text Analysis folder and select **New** → **Dictionary**. Give the new dictionary a name and click **Finish**.

   The dictionary editor appears in the editor view. On the left side of the editor you can see a view of the dictionary, including each term and a sample of its variants. On the right side of the editor is the editing zone.
2. To add a new entry to the dictionary, on the right side type `Windows` as the base form, and add the variants `Microsoft Windows`, `XP`, and `Windows XP Professional`.
3. To import entries into the dictionary, click the **Import** button in the dictionary editor to import entries. Select dictionary as the import source, and select a dictionary to import (as shown in Figure 8-18).



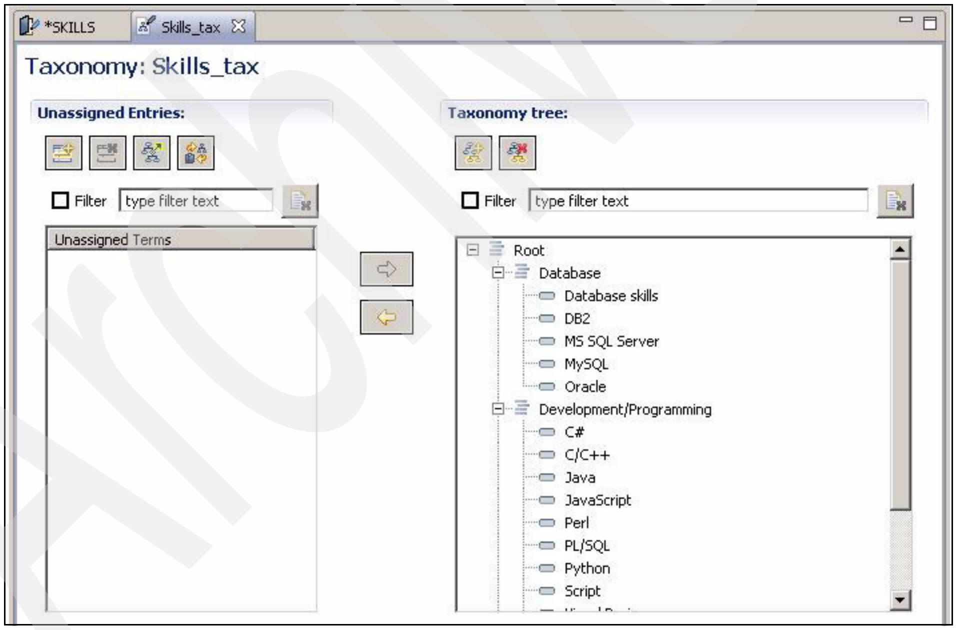
Import
Import dictionary entries from another dictionary.
Specify project and dictionary.
Select dictionary from Data Warehouse project :*
Clustering project
Text Analytics Project
SKILLS
TextAnalysisSample
HeartDisease
it_skills
< Back
Next >
Finish
Cancel

Figure 8-18 Dictionary import

Creating a taxonomy

To create a taxonomy, perform the following steps:

1. Right click the Text Analysis folder, and select **New** → **Taxonomy**.
2. Give the taxonomy a name and click **Next**.
3. Select **Create taxonomy from import source**. Select the Dictionary check box, and click **Next**.
4. Select the dictionary created in "Creating a dictionary" on page 347, and click **Finish**. The taxonomy editor opens.
5. To define the taxonomy, create the first level. On the taxonomy tree right-click **Root**, and select **Add Level**.
6. Add five levels: `Database`, `Development/Programming`, `Operating Systems`, web `Oriented`, and `Others`.
7. Assign the unassigned terms to the levels created in step 6 by placing your cursor on the first category and using the arrow buttons to assign terms to a category. The final taxonomy is shown in Figure 8-19.



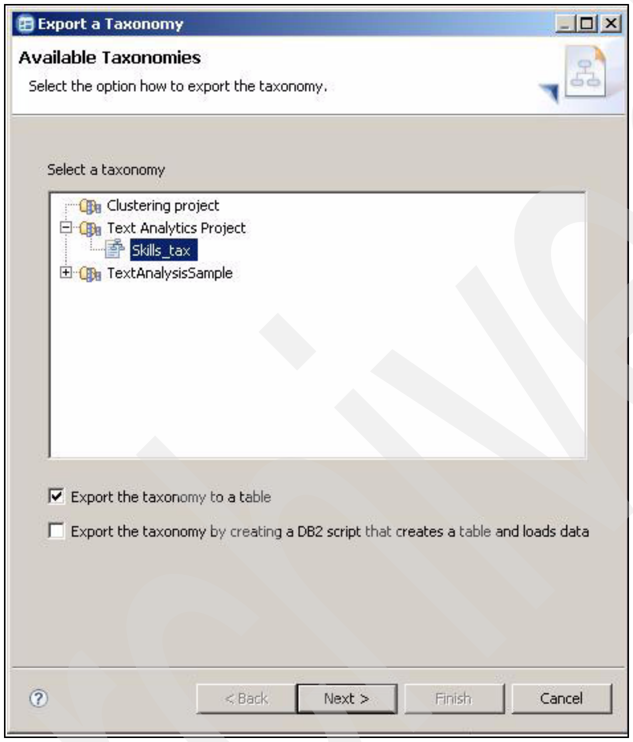
Root
- Database
  - Database skills
  - DB2
  - MS SQL Server
  - MySQL
  - Oracle
- Development/Programming
  - C#
  - C/C++
  - Java
  - JavaScript
  - Perl
  - PL/SQL
  - Python
  - Script

Figure 8-19 Completed taxonomy

Exporting a taxonomy

To export the taxonomy, perform the following steps:

1. Click the **Export Taxonomy** button.
2. Select your taxonomy and make sure that the **Export the taxonomy to a table** check box is selected, as shown in Figure 8-20.



[✓] Export the taxonomy to a table
[ ] Export the taxonomy by creating a DB2 script that creates a table and loads data

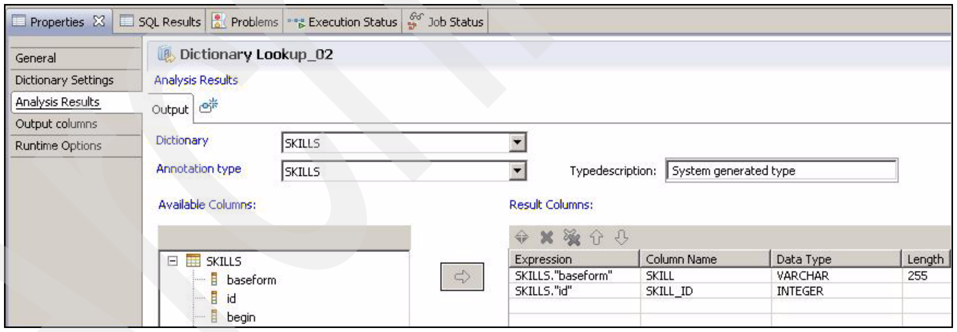
Figure 8-20 Exporting a taxonomy

3. Select the DWESAMP database and click **Next**.
4. Select TXTANL as the schema, and SKILLS_TAX as the new table name. Click **Next**.
5. Click **Next** and **Finish** to create and populate the table.

Creating a mining flow with the dictionary lookup operator

To create a mining flow with the dictionary lookup operator, perform the following steps:

1. Right click the Mining Flows folder and select **New** → **Mining Flow**.
2. Give the flow a name, and specify it to work against the DWESAMP database. Click **Finish**.
3. Drag a Table Source operator on to the canvas, and select the **TXTANL.JOBS** table.
4. Drag a dictionary lookup operator on to the canvas to the right of the table source operator. Connect the two operators.
5. Click the dictionary lookup operator to highlight it, and adjust the properties in the Properties view. On the Dictionary Settings tab, select **JOB_DESC** as the input column.
6. On the Analysis Results tab, select **SKILLS** as the dictionary and the annotation type.
7. Delete **BEGIN**, **END**, and **COVEREDTEXT** from the resulting columns table.
8. Rename the remaining BASEFORM column to `SKILL` to depict the baseforms specified in the dictionary. Rename the ID column to `SKILL_ID` to show the unique identification for each baseform. Figure 8-21 shows the complete Analysis Results tab.



Dictionary: SKILLS

Annotation type: SKILLS

Typedescription: System generated type

Available Columns:

SKILLS
- baseform
- id
- begin

Result Columns:

Expression	Column Name	Data Type	Length
SKILLS."baseform"	SKILL	VARCHAR	255
SKILLS."id"	SKILL_ID	INTEGER	

Figure 8-21 Analysis Results tab

9. On the Output Columns tab of the Properties view, select the columns **ID**, **COMPANY_NAME**, and **TIME** from the list of available columns, and move them to the list of output columns on the right.
10. Place a Distinct operator on to the canvas and connect it to the dictionary lookup operator. This prevents duplicate entries.

11. Drag another table source operator on to the canvas, and select the taxonomy table that you created in step 4 on page 349 in “Exporting a taxonomy” on page 349.

12. Place a Table Join operator on to the canvas. Connect the Distinct operator to the first input port and the taxonomy table to the second input port.

13. Join the two tables on SKILL_ID and TERM_ID.

14. Right-click the output port (inner) of the table join operator, and select **Create suitable table** from the context menu. Call the table `SKILL_REPORT`.

15. The finished mining flow is illustrated in Figure 8-22.

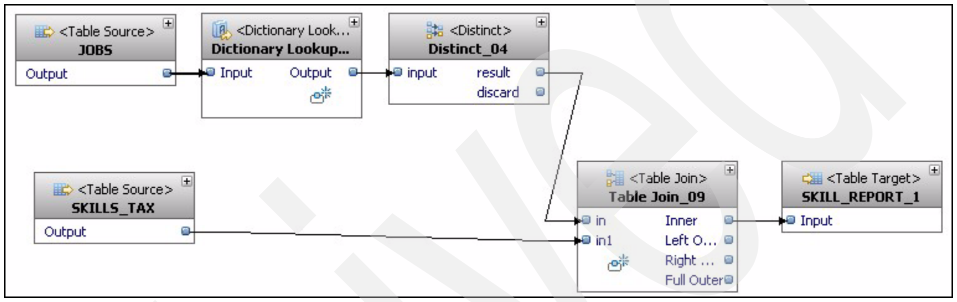
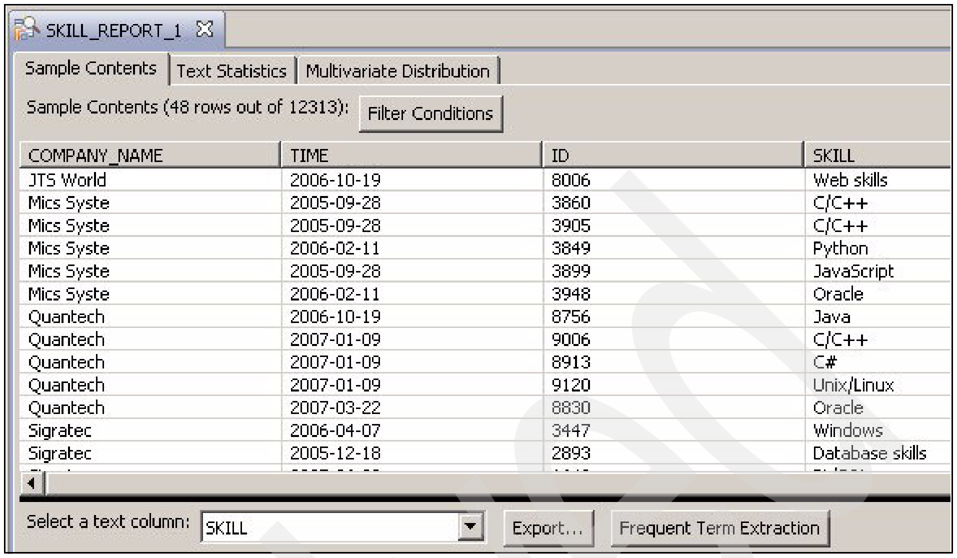


Figure 8-22 Text analysis mining flow

16. Execute your mining flow by selecting **Mining Flow** → **Execute**.

17. In the Data Source Explorer, right-click the report table and select **Distribution and Statistics** → **Data Exploration**. The sample contents of the report table will be shown, as in Figure 8-23 on page 352.



Sample Contents (48 rows out of 12313):

COMPANY_NAME	TIME	ID	SKILL
JTS World	2006-10-19	8006	Web skills
Mics Syste	2005-09-28	3860	C/C++
Mics Syste	2005-09-28	3905	C/C++
Mics Syste	2006-02-11	3849	Python
Mics Syste	2005-09-28	3899	JavaScript
Mics Syste	2006-02-11	3948	Oracle
Quantech	2006-10-19	8756	Java
Quantech	2007-01-09	9006	C/C++
Quantech	2007-01-09	8913	C#
Quantech	2007-01-09	9120	Unix/Linux
Quantech	2007-03-22	8830	Oracle
Sigratec	2006-04-07	3447	Windows
Sigratec	2005-12-18	2893	Database skills

Select a text column: SKILL

Figure 8-23 Skill report table: Sample contents

The text analysis capability in InfoSphere Warehouse can be integrated with IBM Cognos reporting, enabling business users to exploit the text analysis results.

9

Deploying and managing solutions

To provide the data required for analysis, an InfoSphere Warehouse (IW) solution typically contains several components. These components have been discussed in previous chapters of this IBM Redbooks publication. The starting point in an InfoSphere Warehouse implementation is the development of a physical data model to hold the required data. After a data model is in place, the OLAP models can be added to provide optimized access to the data and mining models can be created to enable data-mining analysis of the data. InfoSphere Warehouse further provides the ability to access the data through InLine analytics, which allow the OLAP and mining information to be seamlessly embedded into web pages. Underpinning all of this is the necessity to have SQL Warehousing (SQW) data flows and control flows in place to ensure that the data being used for analysis is current and being accessed in the most efficient way.

Once a solution has been developed, InfoSphere Warehouse provides the ability to deploy and manage it in a production environment. The range of tasks required for each component of the solution varies from ensuring that the production databases are enabled for Online Analytical Processing (OLAP) and mining through to the actual running of the routines to maintain the data. The component in InfoSphere Warehouse that is provided to deploy and manage these tasks is the InfoSphere Warehouse Administration Console.

The administration console component is installed in WebSphere Application Server, which is the infrastructure used for the runtime environment. The SQW data flows and control flows that maintain the production data have to be deployed and managed through the administration console. For other tasks, such as the deploying of the physical model, OLAP models, and mining models, we recommend using the administration console. However, other methods, such as directly deploying SQL scripts, are available.

In this chapter we discuss and describe deploying and managing solutions using the administration console. Topics included are:

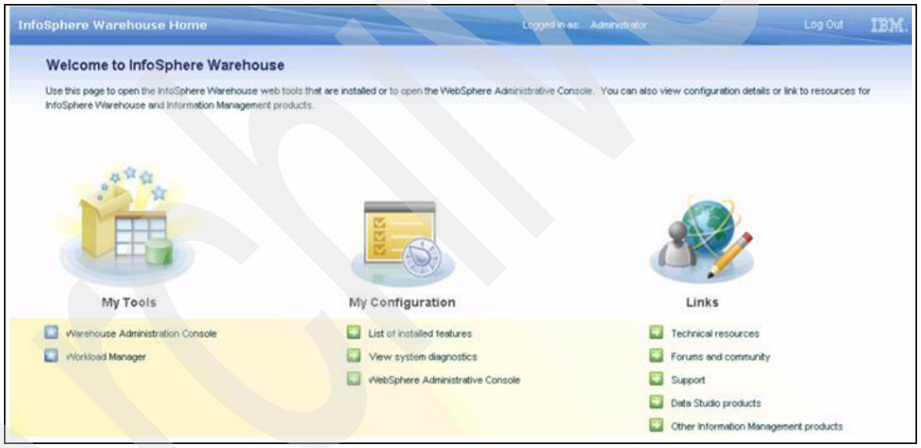
- Administration console
- Deployment of development code into a test or production environment
- Managing the solution with the administration console
  - SQL Warehousing
  - OLAP functionality
  - Data mining
  - Alphablox
- Location and use of diagnostic information

Note: All the functions, features, and operators are not the same on LUW and System z platforms. We point out the differences where appropriate in each chapter.

9.1 The InfoSphere Warehouse Administration Console

The InfoSphere Warehouse Administration Console is the centralized web-based administration component of InfoSphere Warehouse and enables the management and monitoring of production activities through a single client interface. The console provides the ability to deploy and manage SQW, OLAP models, and mining models. The administering of an Alphablox application is through a separate console accessed from the administration console.

With the release InfoSphere Warehouse 9.7 comes a new information portal. This portal provides links to a range of resources related to InfoSphere Warehouse, such as the InfoSphere Warehouse Administration Console, and links to online documentation, the online tutorial, and online support. It also provides interfaces to other InfoSphere components, such as the InfoSphere administration console and the Workload Management console. The information portal links to the online forums for InfoSphere Warehouse and related DB2 communities through the InfoSphere Warehouse home page. Figure 9-1 shows the home portal.


InfoSphere Warehouse Home
Logged in as: Administrator
Log Out
IBM.
My Tools
Warehouse Administration Console
Workload Manager
My Configuration
List of installed features
View system diagnostics
WebSphere Administrative Console
Links
Technical resources
Forums and community
Support
Data Studio products
Other Information Management products

Welcome to InfoSphere Warehouse

Use this page to open the InfoSphere Warehouse web tools that are installed or to open the WebSphere Administrative Console. You can also view configuration details or link to resources for InfoSphere Warehouse and Information Management products.

Figure 9-1 The InfoSphere Warehouse home portal

Figure 9-2 on page 356 depicts the introductory window for the administration console on the LUW platform, which shows the components of an InfoSphere Warehouse solution that can be accessed and managed.

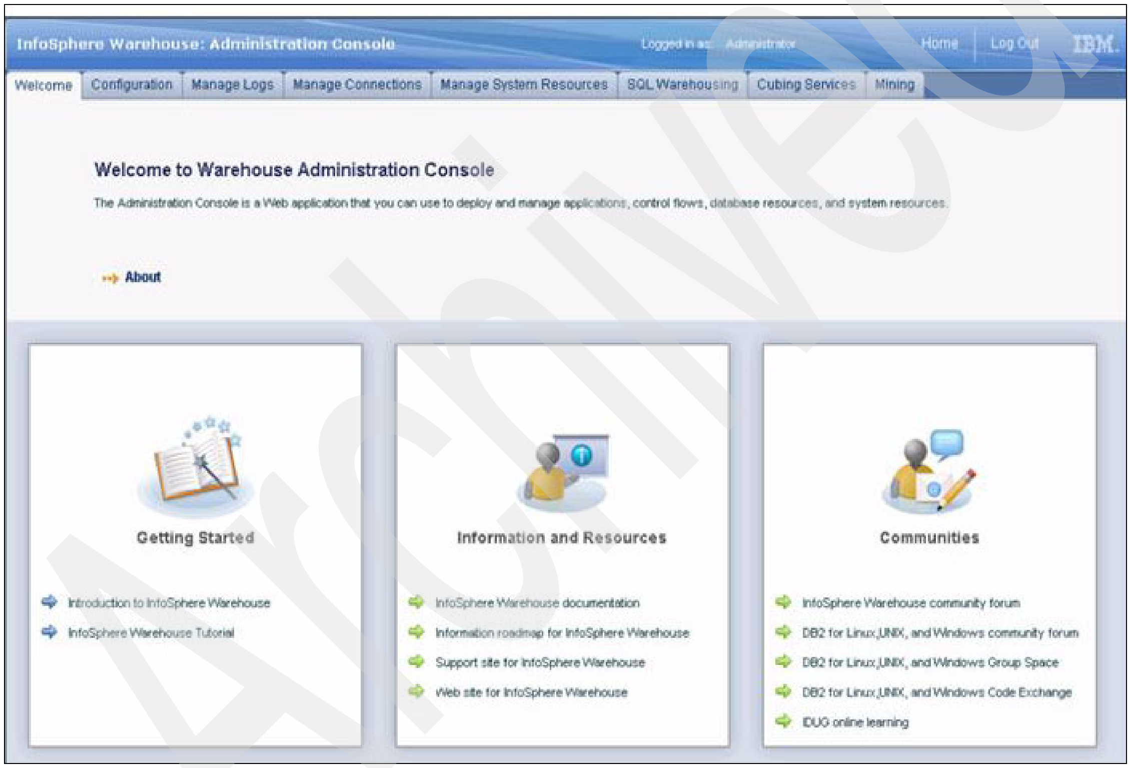
In this section we discuss and describe the following topics:

- Functionality provided by the administration console
- Architecture of the administration console
- Deployment of the administration console
- Security considerations
- General administration tasks
- Locating and using diagnostics

The administration console is accessed through the InfoSphere Warehouse home portal, located at `http://`*`hostname:portnumber`*`/ibm/warehouse`.

In this URL, *`hostname`* refers to the host where WebSphere Application Server is installed. The *`portnumber`* is the default host port number of the WebSphere Application Server profile where the InfoSphere Warehouse Administration Console application is installed.

To access the InfoSphere Warehouse home portal, in a Windows environment, click **Start** → **All Programs** → **IBM InfoSphere Warehouse** → **ISWCOPY01** → **Administration Console and Workload Manager**. Click the link called **Warehouse Administration Console** link, located under My Tools in the portal.



Welcome to Warehouse Administration Console

The Administration Console is a Web application that you can use to deploy and manage applications, control flows, database resources, and system resources.

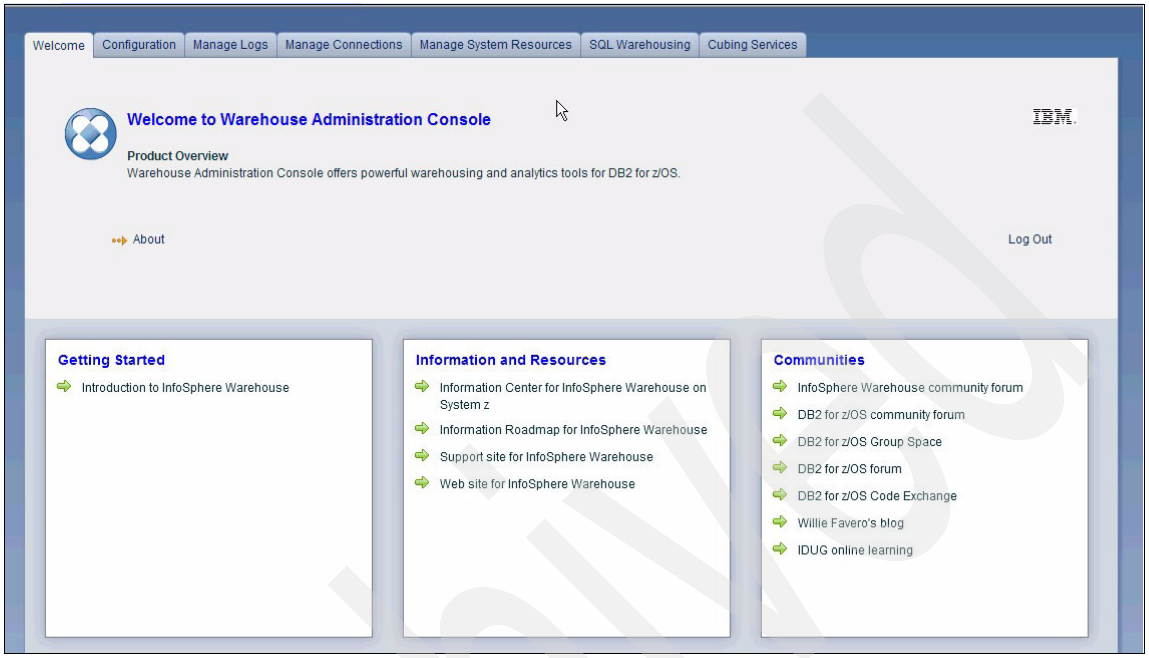
- Introduction to InfoSphere Warehouse
- InfoSphere Warehouse Tutorial

- InfoSphere Warehouse documentation
- Information roadmap for InfoSphere Warehouse
- Support site for InfoSphere Warehouse
- Web site for InfoSphere Warehouse

- InfoSphere Warehouse community forum
- DB2 for Linux,UNIX, and Windows community forum
- DB2 for Linux,UNIX, and Windows Group Space
- DB2 for Linux,UNIX, and Windows Code Exchange
- IDUG online learning

Figure 9-2 The InfoSphere Warehouse Administration Console for LUW platforms

Figure 9-3 depicts the introductory window for the administration console on the System z platform.



Welcome to Warehouse Administration Console

Product Overview

Warehouse Administration Console offers powerful warehousing and analytics tools for DB2 for z/OS.

Getting Started

- Introduction to InfoSphere Warehouse

Information and Resources

- Information Center for InfoSphere Warehouse on System z
- Information Roadmap for InfoSphere Warehouse
- Support site for InfoSphere Warehouse
- Web site for InfoSphere Warehouse

Communities

- InfoSphere Warehouse community forum
- DB2 for z/OS community forum
- DB2 for z/OS Group Space
- DB2 for z/OS forum
- DB2 for z/OS Code Exchange
- Willie Favero's blog
- IDUG online learning

Figure 9-3 The InfoSphere Warehouse Administration Console for System z

9.1.1 Administration console functionality

The functionality offered by the administration console is divided into several sections, identified as separate tabs, as illustrated in Figure 9-2 on page 356. The administration console provides the ability to configure, deploy, administer, and monitor InfoSphere Warehouse components in an operational environment. The following list introduces the features of the administration console described in detail in this chapter:

- The Configuration Management feature allows for the definition of email notification and system logging services.
- The Log Management feature provides the ability to view the system logs.
- The Connection Management feature provides for the definition of any required and supported data sources (DB2 and non-DB2). Before a database can be used for OLAP, mining, or SQW activity, it has to be defined in the administration console. The underlying connection to the database is a Type 4 JDBC driver.

- The System Resource Management feature enables users to define DataStage servers, or computers used in FTP operations. If your control flows do not contain any FTP operations or DataStage jobs, you do not need to define system resources.
- The SQL Warehousing Administration feature controls the deployment, running, scheduling, and monitoring of data warehouse applications that were created in Design Studio. The administration console provides for the viewing of statistics and logs associated with processes and for the troubleshooting of any runtime issues.
- The Cubing Services Management feature allows for the definition and management of cube servers and cubes, and provides the ability to import cube metadata. Cube security is also defined and managed through this interface.
- The Mining Administration feature provides the functionality for the viewing, exporting, updating, and deleting of models in the mining database. Mining models can be imported into the database and models loaded into the cache. The administration console further provides a mining visualization tool that provides graphic representations of the results of the mining model.

9.1.2 Architecture

The administration console is a platform for managing InfoSphere Warehouse components in a runtime environment. To support this functionality the architecture of the administration console interfaces with the individual InfoSphere Warehouse components, as well as the WebSphere Application Server environment.

Administration console interfaces with InfoSphere Warehouse components

The administration console is a J2EE application that is installed locally to a WebSphere Application Server as part of the InfoSphere Warehouse installation process. The console provides a common web interface that is based on Java Server Pages (JSPs)/Java Server Faces (JSF) technology. This common web interface is combined with a common administration infrastructure that provides services between the web client and the underlying administration interfaces for the SQW, OLAP, and data mining. These interfaces are illustrated in Figure 9-4 on page 359.

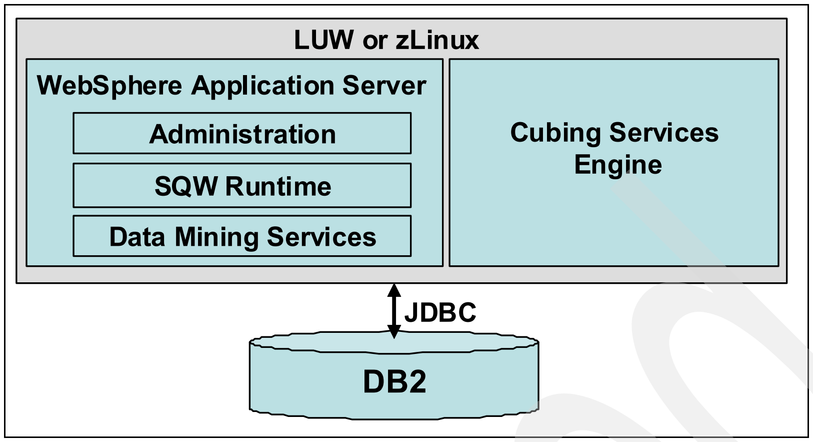


Figure 9-4 Administration console components

Using the WebSphere Application Server environment

in the WebSphere Application Server environment the administration console is packaged as an Enterprise Archive file (EAR) called `IBMDataToolsWeb.ear`. This archive contains the administration console interface as well as the individual components for SQW, OLAP, mining, and Enterprise JavaBeans (EJB) for scheduling. The topology of the administration console can be viewed from the administration console by selecting **Applications** → **Application Types** → **InfoSphere Enterprise Applications** → **IBMDataToolsWeb**.

Being an enterprise application, the administration console takes advantage of the following technologies:

- Scheduling

  The WebSphere Application Server scheduler works in conjunction with the scheduling defined as part of InfoSphere Warehouse and is used to schedule the SQW processes.

- JDBC Providers

  WebSphere Application Server provides configurable data source properties including connection pooling and J2C authentication.

- Mail providers

  WebSphere Application Server provides a default SMTP mail provider for InfoSphere Warehouse notification services.

- User account repositories

  WebSphere Application Server supports user account repositories including the local operating system, LDAP, and federated repositories.

9.1.3 Configuring the administration console

After the InfoSphere Warehouse software is installed, there are configuration steps required for the administration console.

Installing Security with pre-existing WebSphere Application Server

Security for the warehouse is controlled by the WebSphere Application Server. If you choose to install WebSphere Application Server during the InfoSphere Warehouse installation, the security configuration steps are included as part of the installation process. If, however, you choose to use a pre-existing WebSphere Application Server to support the InfoSphere Warehouse, then these security configuration steps must be performed.

Note: The following set of instructions are for WebSphere Application Server version 7.0. If you are using WebSphere Application Server version 6.1, see the following web page for appropriate information:

`http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.datatools.accommon.doc/configuring_security.html`

1. Log in to the WebSphere Application Server (typically at browser address `http://`*`host_name`*`:9060/ibm/console`).

   In Windows, click **Start** → **All Programs** → **IBM WebSphere** → **Application Server v7.0** → **Profiles** → **AppSrv01** → **Administrative Console**.
2. Expand the Security section on the left side of the WebSphere Application Server console window, click **Global Security** → **Local Operating System** (as the Available Realm Definition) → **Configure WebSphere**.
3. On the next window, under General Properties, provide a user name that is defined to the operating system. This user ID is assigned administrative privileges for the WebSphere Application Server environment.
4. Click OK, select the **Set as Current** check box, and click **Apply**.
5. Configure Global Security by selecting the boxes to enable administrative security and application security. The Java 2 security box should not be selected, to restrict application access to local resources. Click **Apply**.
6. Save the security settings so that WebSphere Application Server uses them each time the application server is started. At the top of the window, click the hyperlink to save directly to the master configuration. You can then re-start WebSphere Application Server.

User roles in the administration console

The InfoSphere Warehouse Administration Console uses three different user roles to grant privileges to the various functions that the console manages:

- Administrator
- Operator
- Viewer

Each of these roles has different privileges for each console component (SQW, Cubing Services, Mining, and Configuration).

Viewer role

The viewer role is designed for system monitoring. Most of the status information is available to the Viewer role, but users in this role cannot make any changes to parameters or settings. Users assigned to the viewer role might view any public connections or system resources, as well as any private connections or system resources to which they have been granted by an operator or an administrator. Members of the viewer role might not add any connections or system resources.

Operator role

The operator role is designed to perform operational tasks such as deploying applications and running control flows. In addition to the privileges of the viewer role, the operator might add, edit, or delete any connections or system resources, except private connections that the operator does not own.

Administrator role

Members of the administrator role are super users, and can do any task in the administration console. In addition to the privileges of the operator role, the administrator might add, edit, or delete any connections or system resources, including those that are private and owned by other users.

Table 9-1 on page 362, Table 9-2 on page 362, Table 9-3 on page 363, and Table 9-4 on page 363 describe the privileges that are granted to each role for each of the Console components.

Table 9-1 SQW user role privileges by object

Object	Viewer	Operator	Administrator
Applications	► View the Manage Application page ► View details ► Refresh	In addition to viewer privileges, operator might also: ► Deploy ► Deploy changes ► Enable ► Disable ► Delete	The administrator has same privileges as the operator.
Control flows	► View the Manage Control Flows page ► View the log ► View statistics ► Refresh	In addition to viewer privileges, operator might also: ► Run ► Manage schedules ► Manage profiles ► View details	The administrator has same privileges as the operator.
Instances	► View the Manage Instances page ► Monitor ► View the log ► View statistics ► View details ► Refresh	In addition to viewer privileges, operator might also: ► Stop ► Restart ► Delete	The administrator has same privileges as the operator.

Table 9-2 Cubing Services user role privileges by object

Object	Viewer	Operator	Administrator
Cube Servers	► View the Manage Cube Servers page ► View the Manage the XMLA Service page ► Query status	In addition to viewer privileges, operator might also: ► Start ► Stop ► Restart ► Manage cubes	In addition to operator privileges, administrator might also: ► Create ► Edit ► Delete
OLAP Metadata	► View the Manage OLAP Metadata page	In addition to viewer privileges, operator might also: ► View the Summary Table Usage page ► Optimize ► Run SQL	In addition to operator privileges, administrator might also: ► Import cube models ► Import security ► Edit cube models ► Delete
Cubing Services roles	► View the Manage Cubing Services Roles page ► View privileges	Operator has the same privileges as viewer.	In addition to operator privileges, administrator might also: ► Create ► Edit ► Delete ► Manage users

Table 9-3 Data Mining user role privileges by object

Object	Viewer	Operator	Administrator
Database Enablement	▸ View the Enable Database page ▸ View status ▸ Refresh	Operator has the same privileges as viewer.	In addition to operator privileges, administrator might also: ▸ Enable
Mining Models	▸ View the Manage Mining Models page ▸ View	In addition to viewer privileges, operator might also: ▸ Export	In addition to operator privileges, administrator might also: ▸ Import ▸ Update ▸ Delete ▸ Cache
Cached Mining Models	▸ View the Manage Cached Mining Models page	In addition to the Viewer privileges, the Operator might also: ▸ Reset the cache size ▸ Remove ▸ Remove all	Administrator has the same privileges as operator.

Table 9-4 Configuration user role privileges by object

Object	Viewer	Operator	Administrator
System Notification	▸ View the Configuration page ▸ Refresh	Operator has the same privileges as viewer.	In addition to operator privileges, administrator might also: ▸ Configure system notification
Logging	▸ View the Configuration page ▸ Refresh	Operator has the same privileges as viewer.	In addition to operator privileges, administrator might also: ▸ Configure logging

Configuring access to the administration console

User access to the InfoSphere Warehouse Administration Console and user role mapping is configured in the WebSphere Application Server. The following steps outline the process to assign users to the administration console roles.

1. Log in to the WebSphere Application Server (typically browser address `http://host_name:9060/ibm/console`.

   In Windows click **Start** → **All Programs** → **IBM WebSphere** → **Application Server v7.0** → **Profiles** → **AppSrv01** → **Administrative Console**.

2. Open the Enterprise Application page:

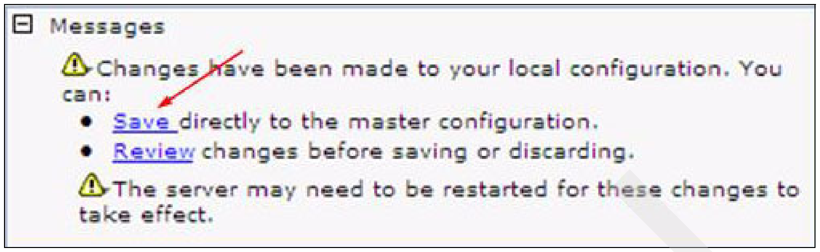
   For WebSphere Application Server V7, click **Applications** → **Application Types** → **WebSphere enterprise applications**

   For WebSphere Application Server V6, click **Applications** → **Enterprise Application**

3. Click **IBMDataToolsWeb**.
4. On the next window, under the Detail Properties section, click **Security role to user/group mapping**.
5. Select the role you want to provide mappings for, and click **Map groups button or the Map users**.
6. On the next window, click the **Search** button to bring up the list of groups (or users) that are defined to the system and therefore available to be mapped to the chosen role.
7. When the list of groups or users has been populated, highlight the ones that you want to map and click the right arrow button to move the group from the Available list to the Selected list. After you have selected all of the groups or users to be mapped, click **OK**.

   The list of mapped users and groups for the selected role has now been updated.

8. Click **OK** to exit the mapping function.
9. A warning message is displayed at the top of the Enterprise Applications properties window. Click the hyperlink to save directly to the WebSphere master configuration, as shown in Figure 9-5 on page 365.


Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

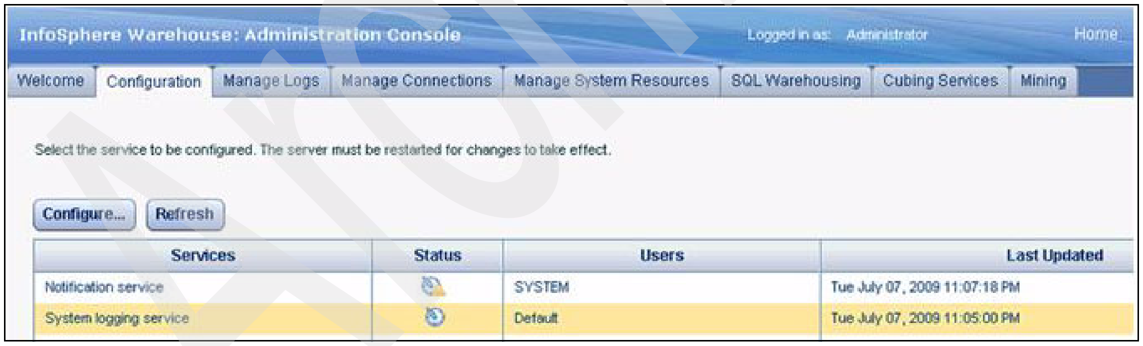
Figure 9-5 Applying changes to the WebSphere master configuration

10. Repeat these steps for each of the roles of Administrator, Operator, and Viewer.

Configuring administration console Services

The administration console allows for the configuration of system logging services and email notification services. These features are configured by accessing the Configuration tab from the Welcome page of the InfoSphere Warehouse Administration Console, as shown in Figure 9-6. Highlight the service you want to work with, and click **Configure**.

> **Note:** Any configuration changes made to the system logging service or the notification server require the WebSphere Application Server to be restarted.



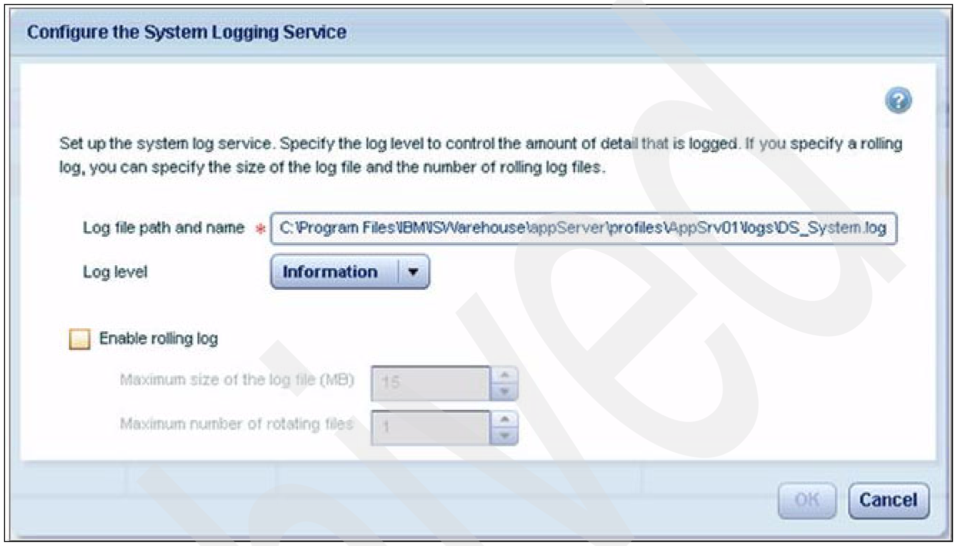
Select the service to be configured. The server must be restarted for changes to take effect.

Services	Status	Users	Last Updated
Notification service		SYSTEM	Tue July 07, 2009 11:07:18 PM
System logging service		Default	Tue July 07, 2009 11:05:00 PM

Figure 9-6 Configuration feature

Setting up the system logging service

The system logging service configuration allows you to define the location and settings for a system log file for the administration console. This gives you the viewer privilege to view the system log through the Manage Logs tab. The system logging configuration window is shown in Figure 9-7.


Configure the System Logging Service

Set up the system log service. Specify the log level to control the amount of detail that is logged. If you specify a rolling log, you can specify the size of the log file and the number of rolling log files.

Log file path and name * C:\Program Files\IBM\ISWarehouse\appServer\profiles\AppSrv01\logs\DS_System.log

Log level: Information

☐ Enable rolling log

Maximum size of the log file (MB): 15

Maximum number of rotating files: 1

Figure 9-7 Configuring System Logging

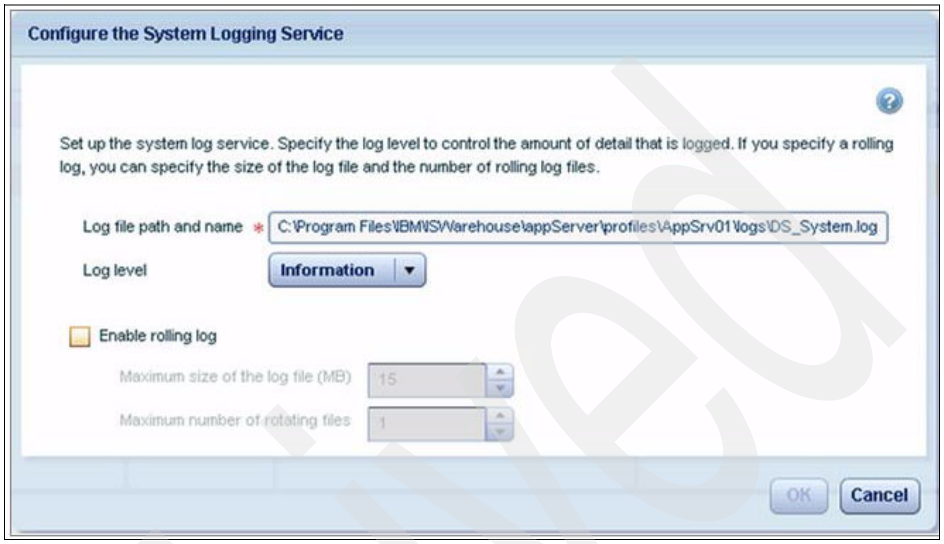
1. Specify a directory and file name for the system log.
2. Specify the amount of detail contained in the log by selecting a Log Level.
   - Error: Important runtime errors
   - Warning: Runtime events that are potential problems
   - Information: Runtime events of interest
   - Debug: Detailed information for debugging purposes
   - Trace: Highly detailed information used for debugging (consumes more system resources)

   Note: The logging level selected includes all higher-severity messages. For example, if the log level is set to Information, the system writes Information, Warning, and Error messages to the system log. If the log level is set to Error, only error messages are written to the log.

3. Specify whether you want a rolling log. If enabled, the system logger writes to a collection of system logs rather than one log file. If rolling log is specified, you can configure the size of the log file and the number of log files.

Defining email notification services

The InfoSphere Warehouse solution might be configured to send email alerts from the server. See Figure 9-8 for the required configuration information.


Configure the System Logging Service

Set up the system log service. Specify the log level to control the amount of detail that is logged. If you specify a rolling log, you can specify the size of the log file and the number of rolling log files.

Log file path and name *: C:\Program Files\IBM\ISWarehouse\appServer\profiles\AppSrv01\logs\DS_System.log

Log level: Information

Enable rolling log

Maximum size of the log file (MB): 15

Maximum number of rotating files: 1

Figure 9-8 Configuring the Email Notification service

Specify the outbound SMTP server and port. By default the port is 25. Change this value if your SMTP service is running on a different port. You can test the outbound notification service by clicking the **Send a Test Email** button.

Metadata configuration

The InfoSphere Warehouse solution uses a metadata repository to store the warehousing metadata. Metadata related to InfoSphere Warehouse applications such as common configurations, SQW, and Cubing Services are stored in this database. This database is often referred to as the control database. The control database is created during the installation process using a default name of SQWCTRL. Configuration details for the control database, including connection settings, are automatically saved by the installation process. This information is stored in a properties file called `metadb.properties`.

There are two approaches available if any changes to the properties file need to be made. InfoSphere Warehouse includes a tool called metadbconfig, located in the `ISW_directory/config/cmdutils` directory. See the readme file for instructions on how to run this utility. The second option is to edit the file directly. Any changes to this file require a restart of the administration console.

Note: If Cubing Services is also installed, the same database information is also stored in another properties file called `cubeserver.xml`. The content of the `metadb.properties` file and the `cubserver.xml` file are the same, but in different formats. It is important that these files are kept synchronized. Use the metadbconfig utility to make any updates to the metadata repository, as this utility updates both property files. If you choose to edit the property files directly, be sure to also update the `cubeserver.xml` file located in the `ISW_directory/CubingServices` directory.

9.1.4 Preparing for application deployment

In preparation for deploying and managing an InfoSphere Warehouse application, the administration console must have access to the following logical databases, shown in Figure 9-9.

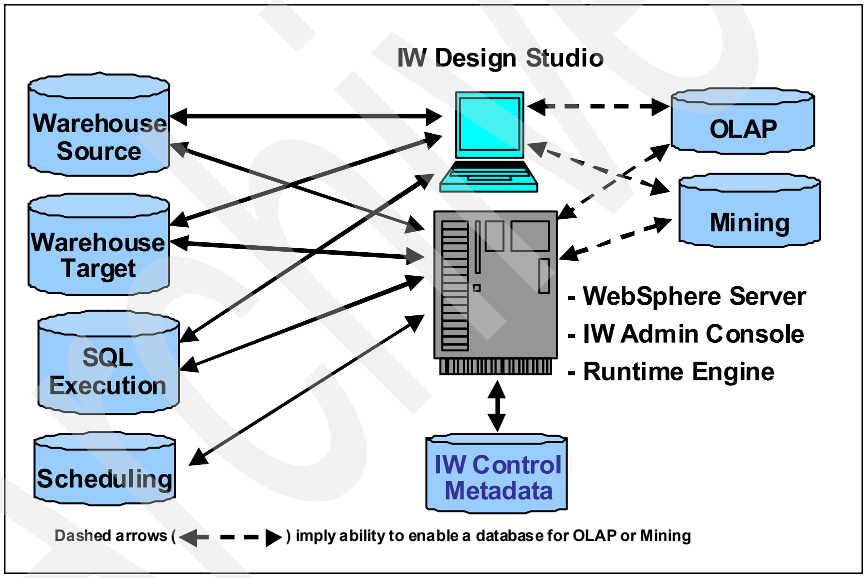


Figure 9-9 Logical databases in a typical runtime environment

The following list describes elements found in Figure 9-9:

- Warehouse source database

  The warehouse source database contains the source data for the data warehouse application. This can be DB2 database or any other supported database.

- Warehouse target database

  The warehouse target database contains the target data for the data warehouse application.

- SQL execution database

  The SQL execution database is declared as a property in a data flow in Design Studio and is a critical property in the run profile for the flow. This is because the code generated by the Design Studio data flow is executed in that execution database. In a data warehouse application that contains multiple data flows, different SQL execution databases might be used to run each data flow. This is beneficial, for example, in an environment where there are multiple SQW processes moving data between several sources and targets. The SQL execution database must be a DB2 database.

Tip: The best performance is likely to occur when the target and execution database are either the same database or part of the same DB2 instance.

- Scheduling database

  The scheduling database is a DB2 database that is used to schedule the various warehouse applications through WebSphere Application Server. Although used by the administration console, the scheduling database is primarily a WebSphere Application Server resource that might contain scheduling information for other enterprise applications.

- Control database

  The Control database is a DB2 database used to store the warehousing metadata about the warehouse source and target, OLAP, and mining databases. This metadata also includes the process that are currently running and historical data relating to average runtimes of processes.

- OLAP database

  The OLAP database is a DB2 database that contains the OLAP information. This database has the star schema and associated InfoSphere Warehouse OLAP metadata that has been defined in Design Studio. This is described in Chapter 6, “InfoSphere Warehouse Cubing Services” on page 169.

- Mining database

  The mining database is a DB2 database that is used to store the mining models. These mining models are created in Design Studio and used in data mining processes, control flows, and Alphablox applications. See Chapter 7, “Data mining” on page 243.

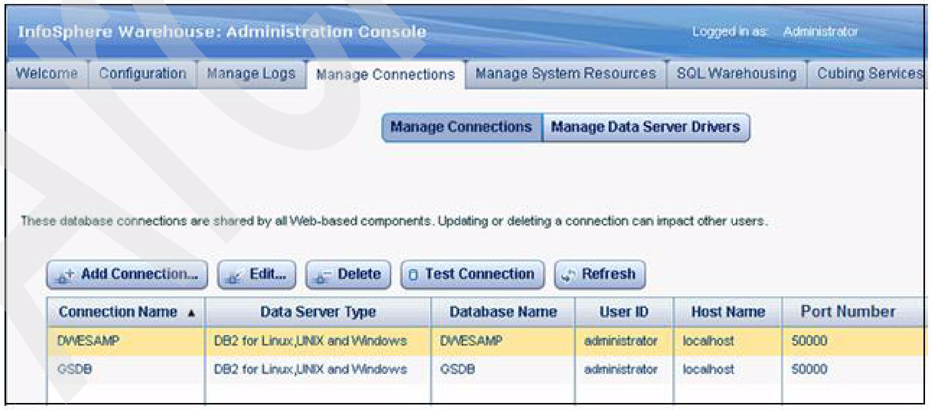
The runtime engine accesses the Control database and uses the information stored there to trigger an activity in the warehouse source or target databases. The administration console triggers the scheduling database and results in the creation of schedules in WebSphere Application Server, which are then automatically executed by the WebSphere Scheduler.

Depending on the characteristics of the environment, many of these logical databases can be combined into one physical database. For instance, the data warehouse target database might also be the OLAP database and include the InfoSphere Warehouse Mining models.

Note: These databases fulfill a role necessary for an InfoSphere Warehouse solution and do not replace any existing DB2 databases. One example is the DB2 Tools database, which is used to schedule DB2 maintenance activities.

Configuring database connections

Before an InfoSphere Warehouse application can be deployed, it is necessary to create the database connections for each database to which the application needs to connect. After a database connection is defined, it is available for deployment and import processes. To manage the database connections, log in to the administration console with a user ID that has either operator or administrator privileges, and click the Manage Connections tab. You are presented with a list of existing database connections, as shown in Figure 9-10. From this window you can add new connections, update the properties of existing connections, delete connections, and test the status of a defined database connection.



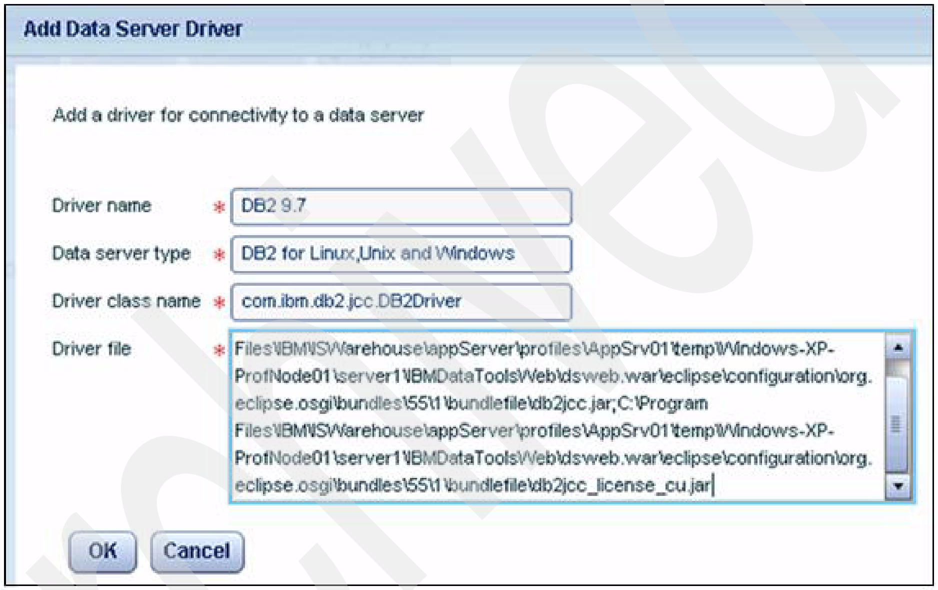
These database connections are shared by all Web-based components. Updating or deleting a connection can impact other users.

Connection Name ▲	Data Server Type	Database Name	User ID	Host Name	Port Number
DWESAMP	DB2 for Linux,UNIX and Windows	DWESAMP	administrator	localhost	50000
GSDB	DB2 for Linux,UNIX and Windows	GSDB	administrator	localhost	50000

Figure 9-10 Managing database connections

Adding data server drivers

The InfoSphere Warehouse has pre-installed the drivers for the IBM family of databases (DB2 for Linux, UNIX, and Windows, DB2 for z/OS, and Informix®). If you need to create a database connection for a non-IBM data server, you need to create a driver definition by clicking **Manage Data Server Drivers**, and clicking **Add driver**. As shown in Figure 9-11, you need to provide the proper values for the driver name, data server type, driver class name, and driver file. See the vendor's documentation to get the detailed information. The `.jar` files used for the new data server drivers must exist on the computer where the WebSphere Application Server resides.



Add a driver for connectivity to a data server

Driver name *	DB2 9.7
Data server type *	DB2 for Linux,Unix and Windows
Driver class name *	com.ibm.db2.jcc.DB2Driver
Driver file *	Files\IBM\ISWarehouse\appServer\profiles\AppSrv01\temp\Windows-XP-ProfNode01\server1\IBMDataToolsWeb\dsweb.war\eclipse\configuration\org.eclipse.osgi\bundles\55\1\bundlefile\db2jcc.jar;C:\Program Files\IBM\ISWarehouse\appServer\profiles\AppSrv01\temp\Windows-XP-ProfNode01\server1\IBMDataToolsWeb\dsweb.war\eclipse\configuration\org.eclipse.osgi\bundles\55\1\bundlefile\db2jcc_license_cu.jar

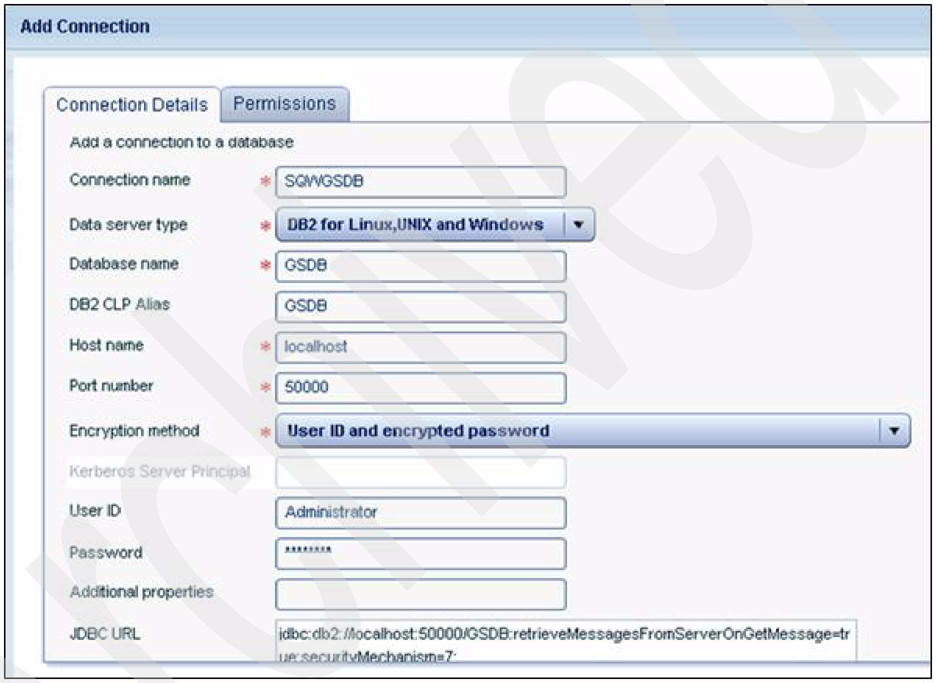
Figure 9-11 Adding a data server driver

Adding new database connections

Each physical database might have one of more database connections defined. For example, if you want to use the same physical database for both cube modeling and SQW applications, you can use the same database connection for both purposes. You can also create separate connections. Creating two separate connections is recommended for each physical database so you can clearly define the scope and purpose of each resource by using different connection or security properties and by applying a naming convention.

To launch the Add Connection wizard, click **Add Connection**. See Figure 9-12.

1. Provide a name for the connection, the data server driver type, and the other connection information (such as database name, hostname, and database port).
2. There are several options available for encryption of the database authentication data. Choose a method and provide the user ID and password.
3. Click the Permissions tab to define the scope for this database connection. Database connections might be Public, Private to a specific user ID, or Restricted to a particular console role (Viewer, Operator, or Administrator).



Connection name *	SQWGSDB
Data server type *	DB2 for Linux,UNIX and Windows
Database name *	GSDB
DB2 CLP Alias	GSDB
Host name *	localhost
Port number *	50000
Encryption method *	User ID and encrypted password
Kerberos Server Principal	
User ID	Administrator
Password	********
Additional properties	
JDBC URL	jdbc:db2://localhost:50000/GSDB:retrieveMessagesFromServerOnGetMessage=true;securityMechanism=7;

Figure 9-12 The Add Connection wizard

Note: Remote DB2 data sources (databases that are not local to the DB2 instance on the application server) must be cataloged in the local DB2 server or the administration client. This ensures that the administration console can successfully run processes that require DB2 command-line access to those sources and targets. For example, the DB2 command line is often used to run SQL scripts that contain LOAD and EXPORT statements. Most application processes require remote databases to be directly accessible from the local catalog. For assistance with the catalog process, see the CATALOG NODE command at the following web page:

`http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.client.doc/doc/t0005621.html`

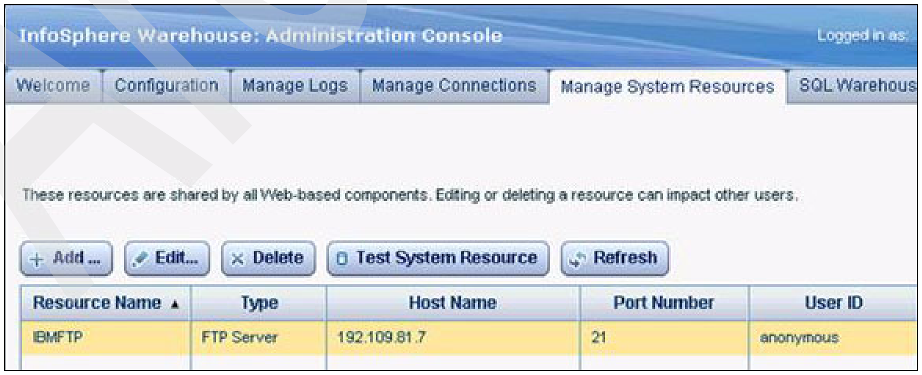
For the CATALOG DATABASE command, see the following web page:

`http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0001936.html`

Configuring system resources

If the InfoSphere Warehouse application that is being deployed requires any additional resources (such as ftp servers, DataStage servers, or other servers), those resources need to be defined in the administration console. To define these resources, perform the following steps:

1. Log in to the administration console with a user ID that has either operator or administrator privileges, and click the Manage System Resources tab.
2. You are presented with the list of existing system resources, as shown in Figure 9-13. From this window you can add new resources, update the properties of existing resources, delete resources, and test the status of a defined system resource.

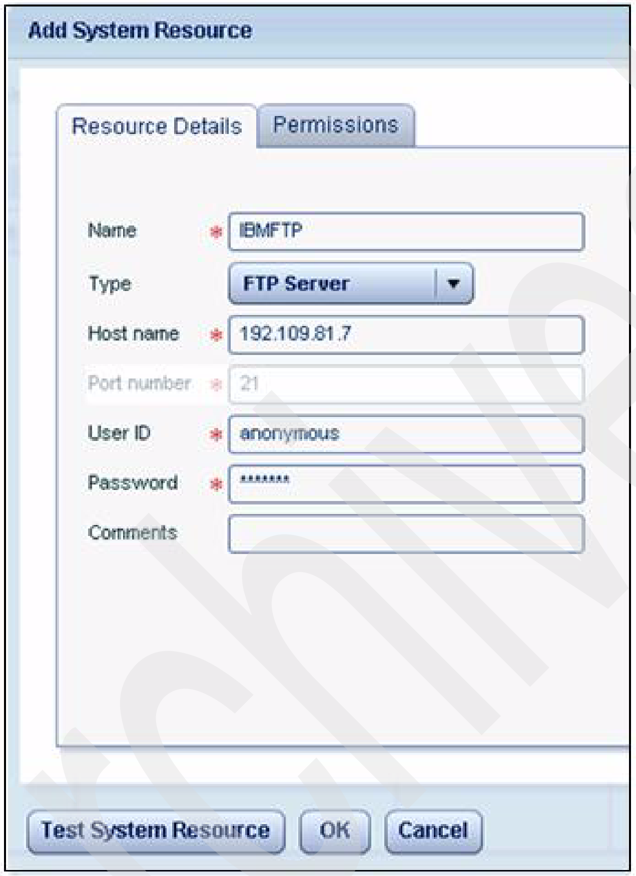


These resources are shared by all Web-based components. Editing or deleting a resource can impact other users.

Resource Name ▲	Type	Host Name	Port Number	User ID
IBMFTP	FTP Server	192.109.81.7	21	anonymous

Figure 9-13 Managing system resources

3. To configure additional system resources, click **Add**.
4. Fill in the fields, providing a name for the resource, the resource type, hostname, and connection information. An example is shown in Figure 9-14.



Name * IBMFTP
Type FTP Server
Host name * 192.109.81.7
Port number * 21
User ID * anonymous
Password * *******
Comments

Figure 9-14 Adding a new system resource

5. Click the Permissions tab to define what class of users might use this system resource.
6. Click **Test System Resource** to confirm that the information provided is accurate and that communication with the resource might be established.

9.1.5 General administration tasks

The administration console primarily works with databases. Two important initial tasks are to define the required databases and enable these databases for data mining activities, if needed. The definition of a database does not create the database. Rather, it links to an existing local or remote database. See **"Configuring database connections" on page 370** for information about defining the required databases.

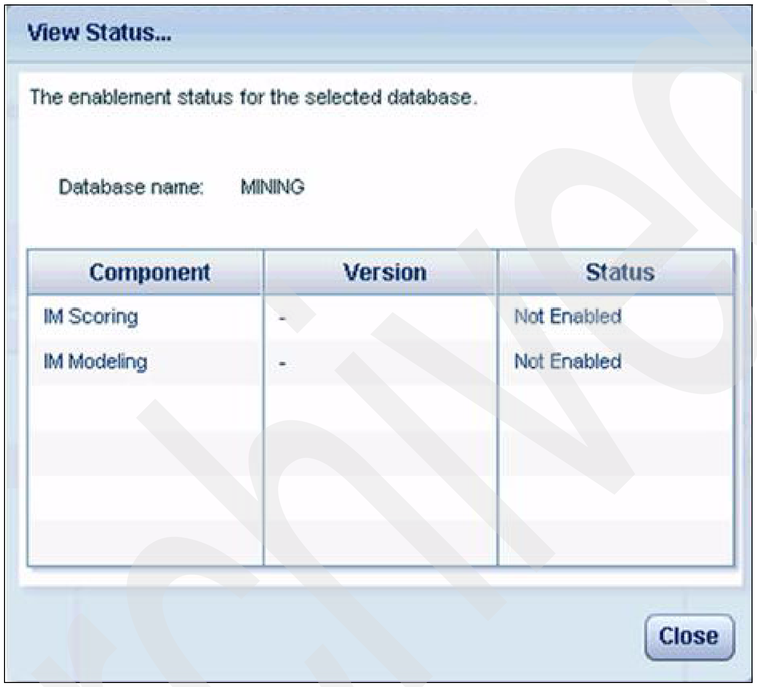
Enabling defined databases for mining

Once the database access has been configured, the database can be enabled for mining. Until a database is enabled for data mining, it lacks the necessary objects, such as stored procedures and tables, to support those activities. The data mining objects are created in the IDMMX schema. This enablement is also required if you want to view univariate and bivariate value distributions with Design Studio. There are three approaches that can be used to enable a database for mining. They are covered in the following sections:

- Using the administration console to enable data mining. See page 376.
- Using Design Studio to enable data mining. See page 378
- Using the command line to enable data mining. See page 379.

Using the administration console to enable data mining

Enabling a database in the administration console is performed through the Mining tab. Through the administration console you can check the status of a database to see if it has been enabled, as shown in Figure 9-15. If the database has not been enabled, the enabling process can be invoked to create the required database objects. Launch the enablement process by highlighting the database you want to work with, and clicking **Enable Database**.



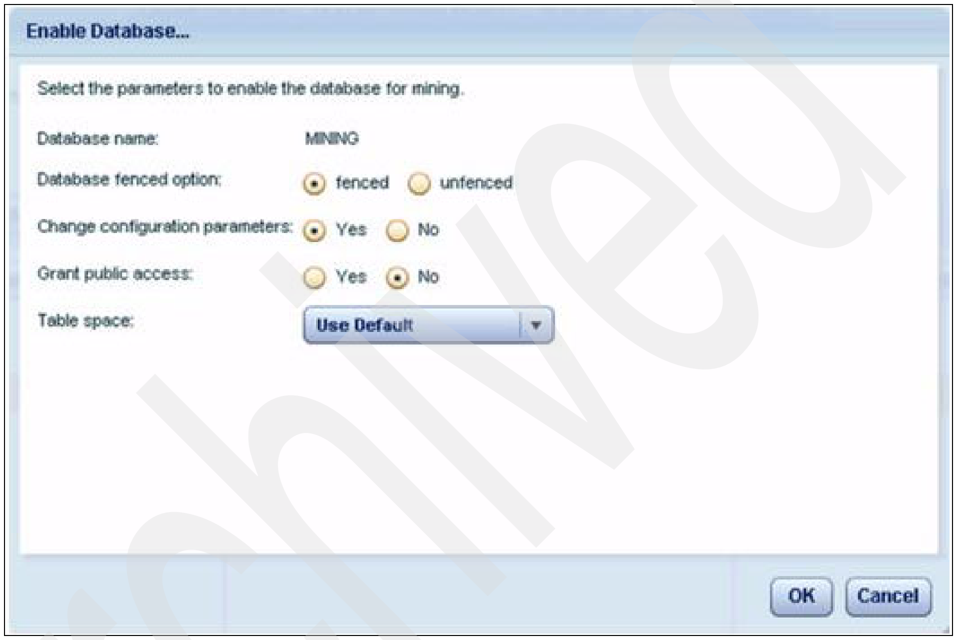
The enablement status for the selected database.

Database name: MINING

Component	Version	Status
IM Scoring	-	Not Enabled
IM Modeling	-	Not Enabled

Figure 9-15 Checking the mining enablement status of a database

Figure 9-16 show the options that must be provided as part of the enablement process. One option is whether the mining routines should be deployed as fenced or unfenced. The difference is that fenced routines are invoked directly by DB2, and unfenced routines are invoked externally and run in processes that are separate from the DB2 agents. If you plan to use caching for mining models, then the database must be enabled in fenced mode. This is because the unfenced mode for routines does not support model caching. You can also change the database configuration settings, and dictate the security permissions.



Select the parameters to enable the database for mining.

Database name:	MINING
Database fenced option:	[✓ fenced] unfenced
Change configuration parameters:	[✓ Yes] No
Grant public access:	Yes [✓ No]
Table space:	Use Default

Figure 9-16 Enabling a database for data mining through the administration console

Using Design Studio to enable data mining

It is possible to create the data mining metadata through the Data Source Explorer of Design Studio. Perform the following steps:

1. Highlight the database you need to work with, right click and choose connect.
2. Provide the appropriate connection properties.
3. After connecting to the database, right-click again, and choose the option to enable the database, as shown in Figure 9-17.

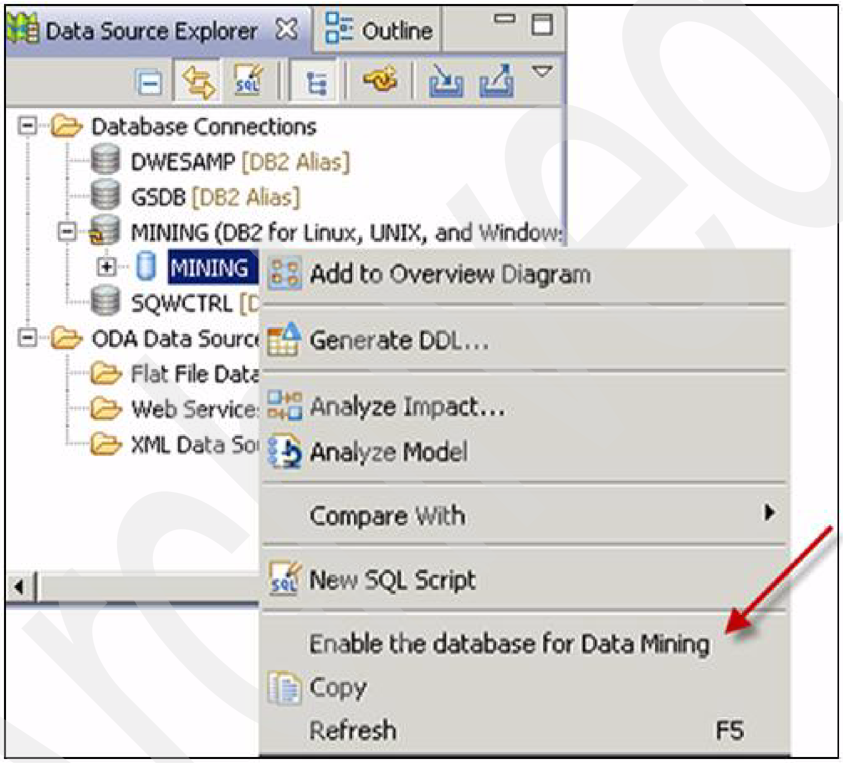
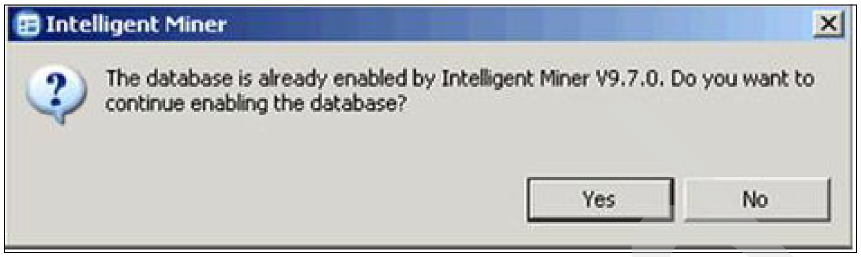


Figure 9-17 Enabling database mining through Design Studio

If the data mining objects already exist in the database, you will be given a warning message, as shown in Figure 9-18.



The database is already enabled by Intelligent Miner V9.7.0. Do you want to continue enabling the database?

Figure 9-18 Design Studio warning message if data mining is already enabled

Using the command line to enable data mining

Another approach to create the required database objects for data mining is to use the **`idmenabledb`** command, shown in Example 9-1. This command is stored in the `ISW_directory/sqllib/bin` directory of your InfoSphere Warehouse installation. The **`idmenabledb`** command requires at least the database name as an input parameter, with optional parameters for the fenced option and the configuration option. If you call the **`idmenabledb`** command with only the database name, the fenced mode is enabled by default.

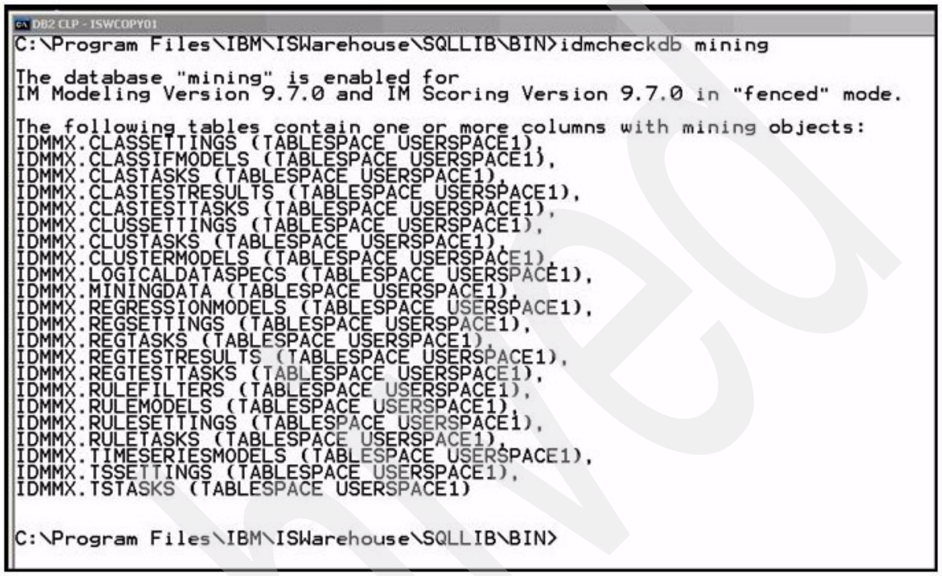
Example 9-1 Calling the idmenabledb command

```
idmenabedb <dbname> fenced dbcfg

idmenabledb MINING

idmenabledb MINING fenced
```

Note: To call the **`idmenabledb`** command, you must have SYSADM or DBADM authority.

Verifying the data mining configuration

To ensure that the data mining objects were successfully created, use the **`idmcheckdb`** command. Like the **`idmenabledb`** command, **`idmcheckdb`** is located in the `ISW_directory/sqllib/bin` directory of your InfoSphere Warehouse installation. An example of the command output is shown in Figure 9-19.



```
C:\Program Files\IBM\ISWarehouse\SQLLIB\BIN>idmcheckdb mining

The database "mining" is enabled for
IM Modeling Version 9.7.0 and IM Scoring Version 9.7.0 in "fenced" mode.

The following tables contain one or more columns with mining objects:
IDMMX.CLASSETTINGS (TABLESPACE USERSPACE1),
IDMMX.CLASSIFMODELS (TABLESPACE USERSPACE1),
IDMMX.CLASTASKS (TABLESPACE USERSPACE1),
IDMMX.CLASTESTRESULTS (TABLESPACE USERSPACE1),
IDMMX.CLASTESTTASKS (TABLESPACE USERSPACE1),
IDMMX.CLUSSETTINGS (TABLESPACE USERSPACE1),
IDMMX.CLUSTASKS (TABLESPACE USERSPACE1),
IDMMX.CLUSTERMODELS (TABLESPACE USERSPACE1),
IDMMX.LOGICALDATASPECS (TABLESPACE USERSPACE1),
IDMMX.MININGDATA (TABLESPACE USERSPACE1),
IDMMX.REGRESSIONMODELS (TABLESPACE USERSPACE1),
IDMMX.REGSETTINGS (TABLESPACE USERSPACE1),
IDMMX.REGTASKS (TABLESPACE USERSPACE1),
IDMMX.REGTESTRESULTS (TABLESPACE USERSPACE1),
IDMMX.REGTESTTASKS (TABLESPACE USERSPACE1),
IDMMX.RULEFILTERS (TABLESPACE USERSPACE1),
IDMMX.RULEMODELS (TABLESPACE USERSPACE1),
IDMMX.RULESETTINGS (TABLESPACE USERSPACE1),
IDMMX.RULETASKS (TABLESPACE USERSPACE1),
IDMMX.TIMESERIESMODELS (TABLESPACE USERSPACE1),
IDMMX.TSSETTINGS (TABLESPACE USERSPACE1),
IDMMX.TSTASKS (TABLESPACE USERSPACE1)

C:\Program Files\IBM\ISWarehouse\SQLLIB\BIN>
```

Figure 9-19 Confirming the data mining enablement

Additionally, you can verify each of the mining features by executing each of the following commands:

- IM Modeling

  ```
  db2 "values( IDMMX.DM_MiningData()..DM_defMiningData('TESTTABLE'))"
  ```

- IM Scoring

  ```
  db2 "values( IDMMX.DM_ApplData('Test',4))"
  ```

- Easy Mining procedures in IM Modeling or IM Scoring

  ```
  db2 "call idmmx.getLastError(?)"
  ```

Each of these commands should return without any errors. If errors are returned, check the installation and configuration of the InfoSphere Warehouse Data Mining feature.

Security housekeeping

Housekeeping might be required when passwords change or new users need access to the administration console. There might be a need to update access to the administration console if the security requirement for an existing user has changed. Additionally, an update to Windows service might be required for DB2.

Required activities when an operating system password changes

If a DB2 server is running as a Windows service, the password for the user running the DB2 services needs to be updated to include this new password. Go to **Control Panel** → **Services** to do so.

If the password of a user ID that establishes a database connection for SQW, Cubing Services, or data mining has changed, update the database connection configuration through the administration console. To perform the update, log in to the administration console with a user ID that has either operator or administrator privileges, and click the Manage Connections tab. You are presented with the list of existing database connections, as shown in Figure 9-10 on page 370. Highlight the connection that is affected by the password change, and click **Edit**. Make the changes to the password field, and click **OK** to save the changes.

If the password for the InfoSphere Warehouse metadata repository (by default this database is named SQWCTRL) is changed, update the metdata configuration. The process to update that configuration is outlined in “Metadata configuration” on page 367.

Changing user access to the administration console

It might be necessary to change access to the administration console if a new operating system user or group that requires such access is created after initial InfoSphere Warehouse installation and configuration has been completed. These assignments can be changed through the InfoSphere administration console, assuming that global security has been enabled:

- If a new operating system user/group is required to access the administration console, they need to be added to WebSphere. Perform the following steps:
  a. Add by navigating to **Users and Groups** → **Administrative User Roles**.
  b. At the top of the view, highlight the administrative role with which you would like to work.
  c. Leave the * (asterisk) in the search string field and click **Search**.
  d. Highlight the user ID from the list of available IDs, and click the right arrow button to move the ID to the Mapped to role column.
  e. Click **OK**, then save to the master configuration.

  Note: The OS user/group must already be created, or an error is returned.

- Once the user/group has been defined to WebSphere Application Server, they need to be mapped to the InfoSphere Warehouse administration roles.

  See “Configuring access to the administration console” on page 364 for the required steps.
- If a user/group needs to be prevented from accessing the administration console, perform the following steps:
  a. Follow steps 1–6 from “Configuring access to the administration console” on page 364.
  b. In the “Selected list of users/groups” select the user/group from which you want to remove access, and move them back to the Available list of users/groups column by clicking the left arrow.
  c. Choose **OK** and save to master configuration.

9.1.6 Locating and using diagnostics

The InfoSphere Warehouse Administration Console is an application in WebSphere Application Server, which means that the WebSphere Application Server tools can be used to provide diagnostic information. The primary source for WebSphere Application Server diagnostic information is the InfoSphere administration console, which in a default installation can be found at the secure site https://localhost:9043/ibm/console. Or at the non-secure site: http://localhost:9060/ibm/console

This console provides logging and tracing for WebSphere Application Server, runtime messages, and the ability to trace data sources that have been defined in WebSphere Application Server.

- Runtime messages

  The runtime messages are accessed from the InfoSphere administration console by navigating to **Troubleshooting** → **Runtime Messages**. These messages are grouped as either errors, warnings, or information. The runtime event logging needs to be enabled. The first time this view is accessed the user is prompted to enable the events.
- Logging and tracing

  For each instance of the WebSphere Application Server, the logging and tracing can be viewed by navigating to **Troubleshooting** → **Logs and Trace** and selecting the server for which diagnostics are to be viewed.

There are five options for logging and tracing:

- Diagnostic trace

  This option allows viewing and modifying of properties for the diagnostic trace service. By default, the diagnostic tracing is enabled and has a maximum file size of 20 MB.

- JVM logs

  This option allows viewing and modifying of the settings for the Java Virtual Machine (JVM) `System.out` and `System.err` logs for the server. The JVM logs are created by redirecting the `System.out` and `System.err` streams of the JVM to independent log files.

  The `System.out` log is used to monitor the health of the running application server. The `System.err` log contains exception stack trace information that is useful when performing problem analysis. There is one set of JVM logs for each application server and all of its applications.

- Process logs

  The process logs are created by redirecting the standard out and standard error streams of a process to independent log files. These logs can contain information relating to problems in native code or diagnostic information written by the JVM. By default, the logs are named `native_stdout.log` and `native_stderr.log` and can be viewed on the console.

- IBM service logs

  The IBM service log contains both the WebSphere Application Server messages that are written to the System.out stream and special messages that contain extended service information that can be important when analyzing problems. There is one service log for all WebSphere Application Server JVMs on a node, including all application servers, and their node agent (if present). A separate activity log is created for a deployment manager in its own logs directory. The IBM Service log is maintained in a binary format and is primarily used by support personnel.

- Change log detail levels

  This option is used to configure and manage log level settings. Log levels allow you to control which events are processed by Java logging. The default setting is *=info, which means that all traceable code running in the application server, including WebSphere Application Server system code and client code, is processed and that all informational messages are captured.

- InfoSphere Warehouse system diagnostics

  A new system diagnostics tool is provided as part of the new administration console infrastructure. This tool is accessed through the View System Diagnostics link under the My Configuration section of the InfoSphere Warehouse home portal. The tool generates a report providing details about the system environment that is running InfoSphere Warehouse and the InfoSphere Warehouse Administration Console.

- InfoSphere Warehouse system logs

  In "Setting up the system logging service" on page 366, the InfoSphere Warehouse logging service was described. Those logs might be viewed from the Manage Logs tab of the administration console.

- Database diagnostics

  Database-specific diagnostics are available through log files or through tracing of specific database.

  – Database diagnostic log files:

    There are two primary database diagnostic files, the notification log file and the DB2 Instance diagnostic log (`db2diag.log`) file. These two files are created for each instance of DB2 and are located on the database servers.

    • DB2 notification log

      This log file contains information about the DB2 instance that is useful to DBAs. On a Windows system this information is written to the Application Event log, which can be found by selecting **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Event Viewer** and choosing **Application**.

      If the log is sorted by source, the DB2 messages can be easily examined.

      On a UNIX/Linux system the notification log can be found in the location specified by the DIAGPATH database manager configuration parameter. The file name is `<instance_name>.nfy`, and a typical example would be `/home/db2inst1/sqllib/db2dump/db2inst1.nfy`.

    • DB2 Instance diagnostic log (`db2diag.log`)

      This log file contains detailed information about the operation of the DB2 instance. This log file is intended for use by IBM support personnel in the event that an issue requires being reported. If the DIAGPATH database manager configuration parameter is null, the diagnostic information is written to files in one of the following directories or folders:

In Windows environments, user data files, for example, files under instance directories, are written to a location that is different from where the code is installed, as follows:

In Windows Vista environments, user data files are written to `ProgramData\IBM\DB2\`.

In Windows 2003 and XP environments, user data files are written to `Documents and Settings\All Users\Application Data\IBM\DB2\Copy Name`, where Copy Name is the name of your DB2 copy.

In UNIX and Linux environments, this file is located in `<instance_home>/sqllib/db2dump/db2diag.log`, with a typical example being `/home/db2inst1/sqllib/db2dump/db2diag.log`.

– Database tracing

  If required, JDBC tracing can be enabled for the databases through the administration console or by using DB2 CLI traces.

  - JDBC tracing through the administration console

    JDBC connections can be traced by setting the JDBC properties in the Additional Properties field in the data source definition.

    For the DB2 JCC driver, trace can be enabled with the properties:

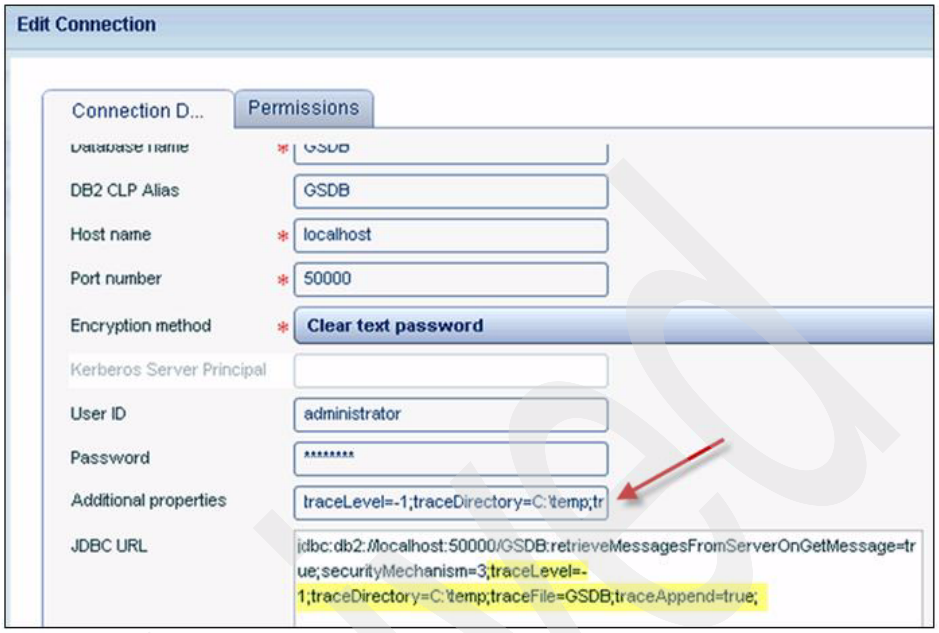
    traceLevel=-1

    traceDirectory=*your_trace_directory*

    traceFile=*your_data_source_name*

    traceAppend=true

    This creates a number of files in the trace directory you specified. The files start with the value you provided to the traceFile parameter. For example, using the settings provided in Figure 9-20 on page 386, we will find trace files starting with GSDB in the `c:\temp` directory.



Database name *	GSDB
DB2 CLP Alias	GSDB
Host name *	localhost
Port number *	50000
Encryption method *	Clear text password
Kerberos Server Principal	
User ID	administrator
Password	********
Additional properties	traceLevel=-1;traceDirectory=C:\temp;tr
JDBC URL	jdbc:db2://localhost:50000/GSDB:retrieveMessagesFromServerOnGetMessage=true;securityMechanism=3;traceLevel=-1;traceDirectory=C:\temp;traceFile=GSDB;traceAppend=true;

Figure 9-20 Enabling JDBC tracing on a data source connection

- If you need to trace the connections to the InfoSphere Warehouse control database (SQWCTRL), make the following updates to the metadb.properties file that was described in section “Metadata configuration” on page 367. This file is located in `<your_installation_directory>/Config`. See Example 9-2.

  Example 9-2 Tracing connections to the SQWCTRL metadata repository

  ```
  traceLevel=-1
  traceDirectory=/tmp/jccTrace
  traceFile=sqwctrl
  traceAppend=true
  ```

  The IBMDataToolsWeb application must be stopped and restarted for these changes to take effect. This is done through the WebSphere Application Server Console.

- DB2 CLI/ODBC/JDBC tracing

  If the DB2 Type 2 JDBC driver is being used and the database connection is not established through WebSphere Application Server, then the tracing facilities in the DB2 CLI driver can be used. Information about how to enable this tracing can be found in the DB2 Information

Center by selecting **Developing** → **Database Applications** → **Programming Applications** → **CLI** → **Diagnostics and error handling** → **CLI/ODBC/JDBC trace facility**.

The DB2 Information Center for LUW can be on the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

The System z information center can be found on the following web page:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp

9.2 Deploying the physical data model

When moving from a development InfoSphere Warehouse environment to a production environment, the first part of a deployment is the database structure that has been created from the physical model. There are a number of available methods to deploy the database structure, whether they are part of the administration console or through another method.

In this section we discuss deployment using the following methods:

- Design Studio
- The administration console
- Native DB2 functionality

9.2.1 Deployment using Design Studio

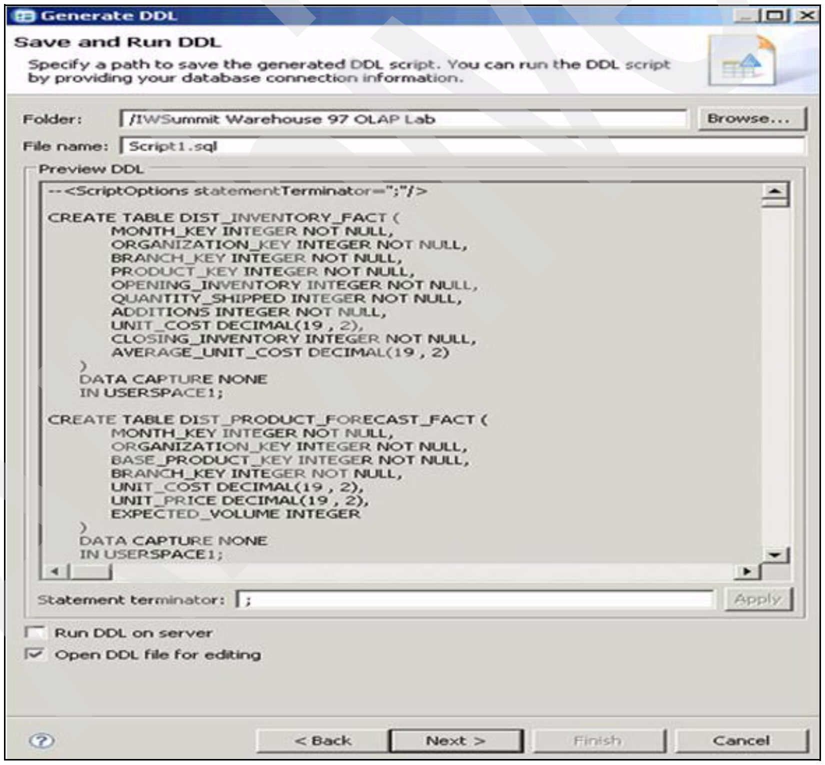
Design Studio has the capability to deploy the physical data model directly to a target database or to generate the DDL to a file. The physical data model can be generated by right-clicking a database in Design Studio after navigating to **Projects** → **Data Models** → ***<Model Name>*** → ***Database*** and choosing **Generate DDL**.

If Design Studio has a connection to the production database and users have the correct authority, the physical model can be directly deployed to the production database. This can be accomplished by selecting the **Run DDL on Server** option in the Generate DDL wizard. If you choose to run the DDL on the server, you are prompted to select a database connection or create a new one.

The option to generate a DDL script file should, however, be considered for a variety of reasons:

- Are there requirements to enter the data model into a source control system?
- If deployment is to be done as part of deploying the warehousing applications, it needs to be included as a DDL file.
- Will the production tablespaces have the same naming and paths as development?
- Can the performance of the physical data model be enhanced by modifying the DDL to add multi-dimensional clustering (MDC) or additional indexing?

The generation or deployment of a data model is shown in Figure 9-21. In this example the DIST_INVENTORY_FACT table has been defined without any indexes and the tablespace is the default of USERSPACE1. The **Open the DDL file for editing** check box has been selected, so the values can be changed in the generated model.



Specify a path to save the generated DDL script. You can run the DDL script by providing your database connection information.

Folder: /IWSummit Warehouse 97 OLAP Lab

File name: Script1.sql

```
--<ScriptOptions statementTerminator=";"/>

CREATE TABLE DIST_INVENTORY_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
        BRANCH_KEY INTEGER NOT NULL,
        PRODUCT_KEY INTEGER NOT NULL,
        OPENING_INVENTORY INTEGER NOT NULL,
        QUANTITY_SHIPPED INTEGER NOT NULL,
        ADDITIONS INTEGER NOT NULL,
        UNIT_COST DECIMAL(19 , 2),
        CLOSING_INVENTORY INTEGER NOT NULL,
        AVERAGE_UNIT_COST DECIMAL(19 , 2)
    )
    DATA CAPTURE NONE
    IN USERSPACE1;

CREATE TABLE DIST_PRODUCT_FORECAST_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
        BASE_PRODUCT_KEY INTEGER NOT NULL,
        BRANCH_KEY INTEGER NOT NULL,
        UNIT_COST DECIMAL(19 , 2),
        UNIT_PRICE DECIMAL(19 , 2),
        EXPECTED_VOLUME INTEGER
    )
    DATA CAPTURE NONE
    IN USERSPACE1;
```

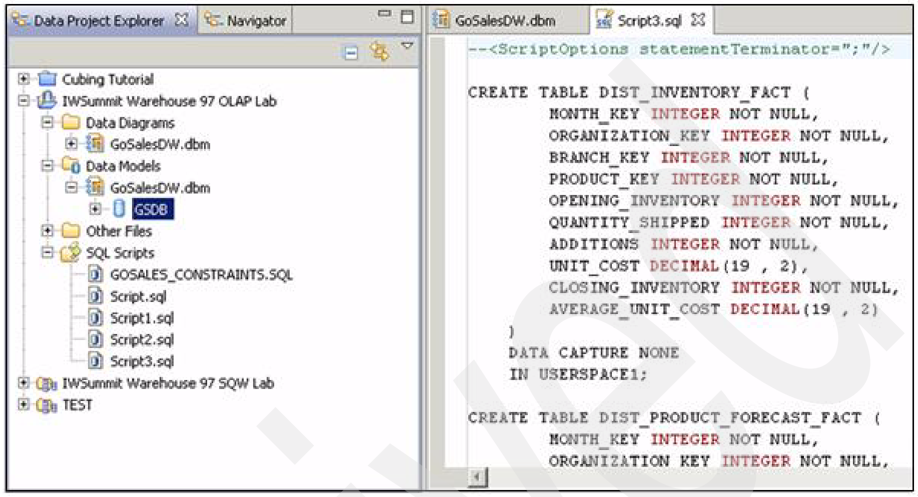
Statement terminator: ;

☐ Run DDL on server

☑ Open DDL file for editing

Figure 9-21 Generating DDL for deployment

Figure 9-22 shows the script editor. When the DDL scripts are saved, they are stored into the logical SQL Scripts folder in the data design project. The scripts can then be forwarded to the DBA staff to be executed.



```
--<ScriptOptions statementTerminator=";"/>

CREATE TABLE DIST_INVENTORY_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
        BRANCH_KEY INTEGER NOT NULL,
        PRODUCT_KEY INTEGER NOT NULL,
        OPENING_INVENTORY INTEGER NOT NULL,
        QUANTITY_SHIPPED INTEGER NOT NULL,
        ADDITIONS INTEGER NOT NULL,
        UNIT_COST DECIMAL(19 , 2),
        CLOSING_INVENTORY INTEGER NOT NULL,
        AVERAGE_UNIT_COST DECIMAL(19 , 2)
    )
    DATA CAPTURE NONE
    IN USERSPACE1;

CREATE TABLE DIST_PRODUCT_FORECAST_FACT (
        MONTH_KEY INTEGER NOT NULL,
        ORGANIZATION_KEY INTEGER NOT NULL,
```

Figure 9-22 The SQL Script Editor in Design Studio

9.2.2 Deployment using the administration console

The InfoSphere Warehouse Administration Console provides two methods for deploying a data model. These two methods both require that the data model be provided in a file, whether this file is generated by Design Studio or by another method, such as db2look. The file containing the model can be deployed as part of an SQW deployment or run directly as an SQL file.

In this section we discuss the following two deployment options:

- Deploying the data model in a warehousing deployment
- Deploying the data model as an SQL file

Deploying the data model in a warehousing deployment

It is possible to package the data model together with the SQW applications that use the deployed tables. If one or more DDL files are included with the deployment package in Design Studio, the files are executed by the administration console as part of the application deployment.

The procedure is as follows:

1. In the project in Design Studio the DDL can be generated from the development database into a file or a file can be imported containing the DDL into the project. These files are typically found in the SQL Scripts folder in the project.
2. Copy the DDL files to the Application Misc Files folder in the Data Project Explorer, as shown in Figure 9-23. Notice that in the Navigator view, this folder is displayed as the `package-misc` subdirectory of your data warehousing project.

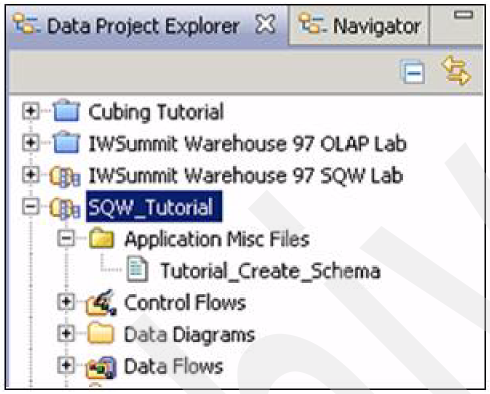
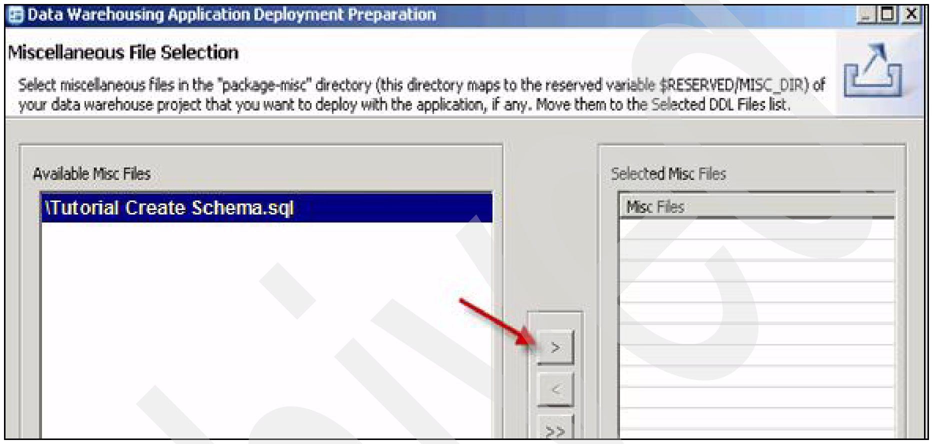


Figure 9-23 Copying the DDL scripts to the Application Misc Files folder

3. When a new data warehouse application is created through the Deployment Preparation wizard, the packaging of the deployment code allows the specification of one or more DDL files that are included with the application. This option is available on the Miscellaneous File Selection page of the wizard. Highlight the scripts to be included in the deployment package, and click the arrow to move the file from the Available column to the Selected column, as shown Figure 9-24.


Data Warehousing Application Deployment Preparation

Miscellaneous File Selection

Select miscellaneous files in the "package-misc" directory (this directory maps to the reserved variable $RESERVED/MISC_DIR) of your data warehouse project that you want to deploy with the application, if any. Move them to the Selected DDL Files list.

Available Misc Files: \Tutorial Create Schema.sql

Selected Misc Files: Misc Files

Figure 9-24 Adding DDL scripts to the deployment package

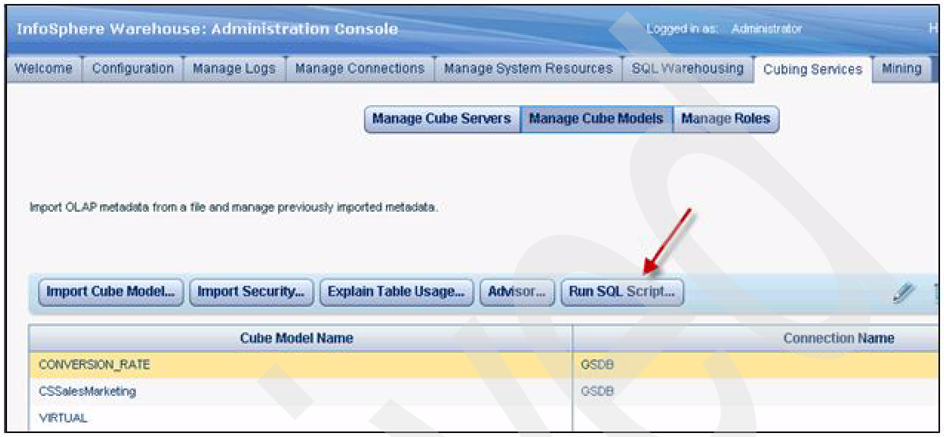
4. When the deployment process is completed, the DDL files that have been specified are contained in the `application_home/applications/application_name/etl/misc` folder located on the application server. The DDL file is now executed as part of the data warehouse application.

Deploying the data model as an SQL file

The deployment of the data model as part of a warehousing deployment is the recommended means of initially creating the tables on a production system. If there is a requirement for creating tables after a deployment or performing additional SQL operations, the administration console provides this functionality. The ability to run SQL statements is provided as part of the InfoSphere Warehouse OLAP functionality and is designed to run statements containing optimizations for any OLAP models.

However, any SQL statements can be run, including DDL. Perform the following steps:

1. On the Cubing Services tab, highlight a cube model, and select **Run SQL Script**, as shown in Figure 9-25**.**



Import OLAP metadata from a file and manage previously imported metadata.

Cube Model Name	Connection Name
CONVERSION_RATE	GSDB
CSSalesMarketing	GSDB
VIRTUAL	

Figure 9-25 Running any SQL script through the administration console

2. You are then prompted for the script file name. Click **Browse** to locate the file, then click **Next**.

   This takes you to the Run SQL wizard. After providing the file name and clicking **Next**, the script will be executed and a results window is displayed. You can

3. (Optional) Save the log file.

The only restriction associated with this approach to data model deployment is that if the database is remote, it must be cataloged locally to the DB2 client.

Tip: Define the statement terminator for the DDL file as a semi-colon (;).

9.2.3 Deployment using native DB2 functionality

As the generated DDL file is a standard DB2 file, it can be deployed in the following ways:

- DB2 Command Center

  The DDL file can be opened in the DB2 Command Center and executed. The statement terminator specified in the Command Center needs to match the terminator in the DDL file.

- DB2 CLP

  The DDL file can be executed from a DB2 command line. If the default statement terminator, the semi-colon, is used, the command is:

  ```
  db2 -tf ddlscript.sql
  ```

  If a different terminator is specified, the command is:

  ```
  db2 -td<terminator> -f ddlscript.sql
  ```

9.3 Deploying an SQL warehousing application

Now that the database objects have been created in a data warehouse, there is a need to design, develop, and deploy a means of maintaining the data in these objects. If a table is created and initially populated with data, the contents of this table have to be maintained to ensure that we are using the correct data. For example, if in one of the dimensional tables there is a product that moves in its hierarchy, this change needs to be reflected in the dimensional table. If this change is not reflected, incorrect business decisions might be made based on an old version of the data.

There are a number of ways to maintain the contents of database tables, such as operating system scripts, SQL scripts, or an ETL tool such as DataStage. Design Studio provides the functionality to develop, validate, and test warehousing applications that primarily work with data after it is in a database. The administration console takes these applications that have been created and deploys them into a production environment. After the application is deployed, the administration console can be used to run and monitor data warehouse applications that contain specific executable processes. The administration console can also be used to set up scheduling of these processes, view execution statistics, and analyze log files.

Before continuing, we want to define what we mean by an application and a process:

- An *application* represents one or more processes that Design Studio users have assembled into a package for deployment. These processes might consist of a set of control flows that build or modify a data warehouse according to a fixed or on-demand schedule. Alternatively, an application might contain a single data flow inside a single control flow that updates one dimension table.
- An individual *process* in the runtime environment is equivalent to a control flow in the design-time environment. When a schedule or process is started, all of the activities in a particular control flow, including its data flows, are executed.

In this section we concentrate on the SQW functionality in the administration console, the runtime deployment of warehousing applications, and the management of these applications after deployment.

Full information about the development of an SQW application can be found Chapter 5, “Data movement and transformation” on page 75.

9.3.1 An overview of the SQW deployment process

The creation of an SQW application and the subsequent deployment of this application are part of the same overall process, which has the aim of populating and maintaining the production databases. However, the different requirements of developing and deploying a warehousing application requires different components and processes. Figure 9-26 illustrates the components involved in the building of an application in Design Studio and the deployment in a runtime environment.

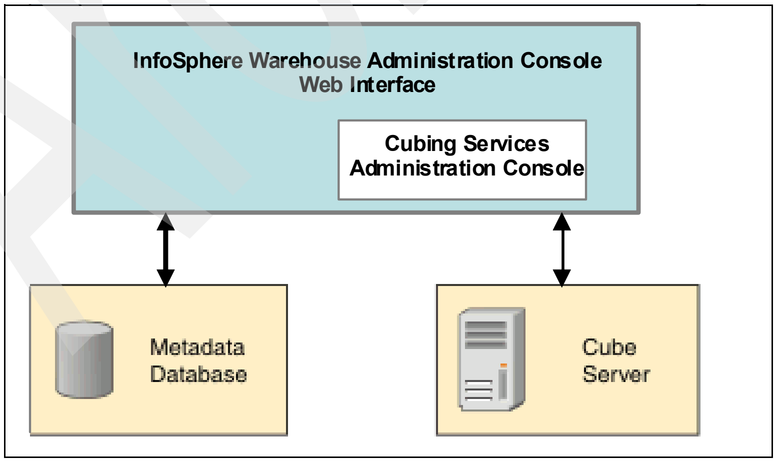

InfoSphere Warehouse Administration Console Web Interface
Cubing Services Administration Console
Metadata Database
Cube Server

Figure 9-26 SQW components in development and production

The preparation of the warehouse application takes place in Design Studio, but the actual deployment takes place in the administration console. The deployment of the new application makes the application processes available for scheduling, execution, and administration. The resources that are referenced in the control flows and data flows must be ready to accommodate the new application. The process for creating these resources is discussed in **“Configuring database connections” on page 370**, and “Configuring system resources” on page 373. Each application to be deployed contains one or more control flows, which, in turn, contains data flows or other activities (such as SQL scripts, executables, FTP commands, or OS scripts). When the application is deployed each control flow becomes a runtime process.

In this section we discuss the:

- Contents of the warehouse deployment file
- Deployment process
- Structures created by the SQL application deployment
- Management of the applications after deployment
- Control flow management
- SQW diagnostics

Contents of the warehouse deployment file

Before the deployment of the warehouse application is discussed, it is worthwhile to understand the contents of the deployment package. The deployment package is a compressed file that is generated by Design Studio for deployment. This compressed file contains data flows and control flows that were created during the design phase. These data flows are compiled and validated based on an application profile, and the appropriate deployment package containing the code is generated as a result. An application profile is a combination of control flows, database definitions, variables, and script files.

For example, you can package as many control flows into a data warehouse application as desired. Multiple applications can contain the same flows. Flows contained in the same application share the same application home/log/work directory after the data warehouse archive has been deployed in the administration console. They can also be enabled and disabled together, but are otherwise independent of each other.

The deployment package file contains the following:

- Deployment files

  The deployment folder contains two control flow EPG (.epgxmi) files for each control flow: one for the deployment of the application and one for the undeployment of the application. The deployment EPGs are run one time when the data warehouse application is deployed to the SQW application-server-based runtime. The undeployment EPGs contain code that is run when the data warehouse application is uninstalled from the SQW runtime. It is the opposite of the deployment EPG and is again only running once.

  Two DDL (epgxmi) files are also created for each application, which contains any DDL files that are to be deployed as part of the application.

- Application metadata (meta-inf)

  The meta-inf folder contains an ibm-etl-app-metadata.xmi file and is generated for each application that contains information, such as the location of log files, what variables are being used, and which databases are being accessed.

- Applicable miscellaneous files (Misc)

  The misc folder contains deployment files that are database setup files bundled with the data warehouse application. They are to prepare the database for use before the control flows are executed. Commands in these files only need to be run once rather than each time a process is run. Examples of the types of commands in these files would be DDL statements to create tables or a database configuration update.

- The runtime execution plan graphs (EPGs)

  The runtime folder contains one runtime EPG (.epgxmi) file for each operator in the control flow, whether they be a data flow, email, or a DB2 command. This file is the XML representation of the graphical element in Design Studio and is run each time the process is invoked.

Deployment process

The deployment process extracts code units from the compressed file generated by Design Studio and creates a deployment file structure. The process also populates control tables in the SQL Data Warehousing control database. See 5.4, "Moving from development to runtime" on page 153 for detailed instructions on the SQW deployment process.

Deployment file structure

Once the application is installed, the relevant application, process, activity, and variable parameters are written to the SQL warehouse control tables. Also, the deployment and execution code units, which are used at process execution time, are written to the application home directory.

Figure 9-27 illustrates what the application deployment directory looks like. The deployment file mirrors the folder structures in the packaged application.

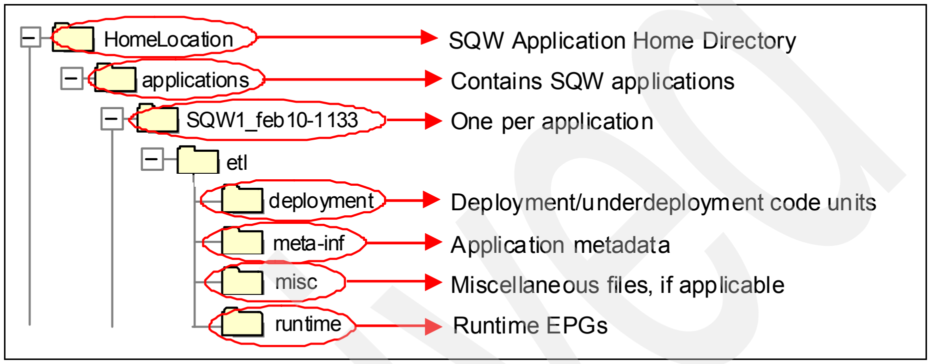

HomeLocation
SQW Application Home Directory
applications
Contains SQW applications
SQW1_feb10-1133
One per application
etl
deployment
Deployment/underdeployment code units
meta-inf
Application metadata
misc
Miscellaneous files, if applicable
runtime
Runtime EPGs

Figure 9-27 Deployment file structure for a warehousing application

During application deployment the log directory is also created. The log directory is the default location for the log files generated when a runtime process is executed. The SQW application deployment also writes log and error messages in a file named `*application name*.html` into the log directory specified at application deployment time.

Managing the applications after deployment

The administration console can be used to manage the data warehouse applications that have been deployed. During the life cycle of an application it might be necessary to update application properties and eventually uninstall the application.

When navigating to **SQL Warehousing** → **Manage Applications** there are four options for the selected application:

- View Details

  This option opens a properties page for the application, where attributes can be changed, such as the log directory for the application or the current value of a variable.

- Enable

  This option enables an application that is currently disabled. An application has to be enabled as a prerequisite for running the processes in the application. When an application is deployed the default is that they are enabled, but you can choose that they not be enabled.

- Disable

  This option disables an application, making its processes unavailable for execution.

- Delete

  This option performs the opposite set of tasks of the deployment task. After an application has been uninstalled, you can no longer use it from the console. The deployment history for an application contains a reference to the fact that the application was undeployed, but all of the metadata about the application is physically removed from the WebSphere Application Server environment.

An application can also be directly selected by clicking it, which allows the properties of the application to be viewed and updated, as illustrated in Figure 9-28 on page 399. The following information is a brief description of the figure tabs:

- Properties

  Allows the log directory to be updated. The comments can also be updated. The tab also displays information about when and by whom the application was last updated.

- Control Flows

  Shows the control flows that are part of this SQW application.

- Data Sources

  Allows you to view the data sources used by the application. This information is read-only, as any updates to the data sources need to be done through the Manage Connections section of the administration console.

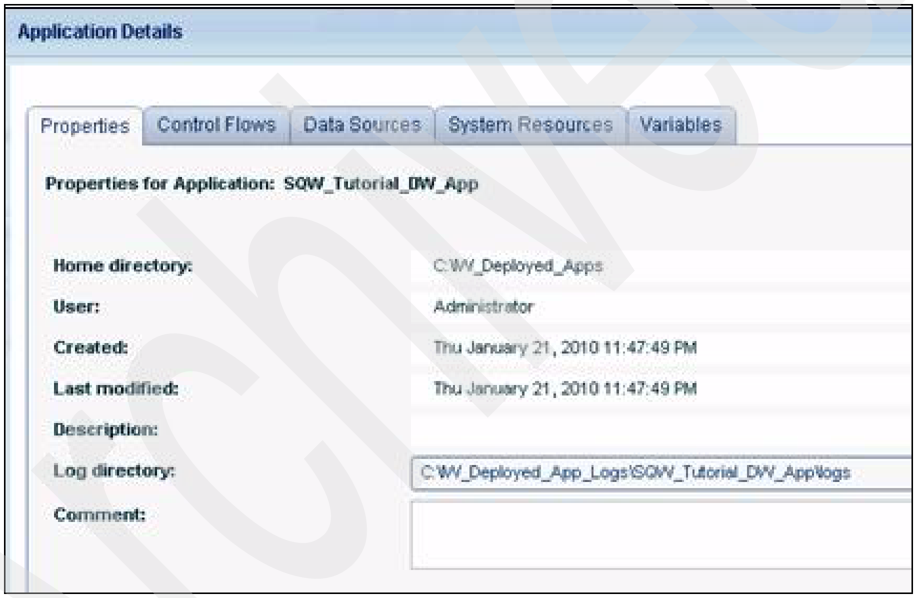
- System Resources

  Allows viewing of the system resources used by the application. This information is read-only, as updating the system resources is done through the Manage System Resources section of the administration console.

- Variables

  Allows the viewing and inserting of new values for the variables that are used by the application. Each variable can be selected, which brings up a page with more information about the variable value and definition.

  Note: This is the only location where a user can modify variables in the runtime and execution phases.



Properties for Application: SQW_Tutorial_DW_App

Home directory:	C:\WV_Deployed_Apps
User:	Administrator
Created:	Thu January 21, 2010 11:47:49 PM
Last modified:	Thu January 21, 2010 11:47:49 PM
Description:	
Log directory:	C:\WV_Deployed_App_Logs\SQW_Tutorial_DW_App\logs
Comment:	

Figure 9-28 Properties of a deployed application

Control Flow management

When an application is deployed to a runtime environment it might contain one or more control flows, which perform warehousing activities such as populating tables, invoking DataStage jobs, or sending out confirmation emails. The management of these tasks includes updating control flow properties such as variable values, scheduling, and monitoring control flows. For more details about managing control flows, see 5.5, “Executing and monitoring control flows” on page 159.

SQW diagnostics

Knowing what diagnostic information is available and where to find it is important for the support of a production InfoSphere Warehouse solution. When specific control flow instances and activities fail to run, you can use the administration console to view information about the failure and troubleshoot the problem.

To monitor and troubleshoot control flow instances, go to **SQL Warehousing** → **Manage Instances.** From this view you can see the list of deployed control flows, and can view the execution status for each activity in the control flow instance. This view also allows you to stop, restart, and refresh control flows.

Log information about the control flows is also available for viewing from the administration console. To view these logs, the logging characteristic of the control flow must have been previously set. To check the logging characteristics, go to **SQL Warehousing** → **Manage Control Flows.** Click the control flow to see the properties. Click the Logging tab to view or change the logging and statistics level. To view the log for a control flow, from the Manage Control Flows view, highlight the control flow of interest and choose to view statistics and log files.

9.3.2 Deploying application changes

After an application has been deployed, you might need to make changes to control flows or data flows, add new control flows, or make other application changes. To support these needs, InfoSphere Warehouse provides the ability to perform a delta deployment as an efficient way to make changes to SQW applications.

The delta deployment feature allows you to deploy just the changes to the application, rather than having to deploy the application again under a new name or removing and replacing an application. Delta deployment supports the following tasks:

- Adding a new control flow to an application
- Adding a new activity to a control flow
- Deleting an activity from a control flow
- Modifying a control flow activity
- Changing resource settings for an application
- Changing variable properties for an application

Delta deployment involves an update to an existing application. The attributes of the original application, such as data sources, variables, and schedules are retained when you deploy changes. During the delta deployment process, existing resources and variables can be mapped to new values and new resources can be defined as required if the change phase is equal to or later than deployment. If an attribute in the original application is not used by the updated application, then this attribute will be removed in the updated application.

Note: Deleting a control flow from an application is not supported for delta deployment. Instead, you must disable the control flow instance in the console or delete and redeploy the entire application.

Note: You must prepare the entire control flow for deployment even if only one property in one operator in one data flow was changed. Additionally, you cannot deploy changes by packaging only an activity, an operator, or a property; the deployment package that you prepare must be functionally complete; it must contain one or more fully functional control flows.

9.4 Deploying InfoSphere Warehouse OLAP functionality

The Cubing Services section of the administration console enables the deployment and management of the InfoSphere Warehouse OLAP solution. Specifically, the console provides OLAP and cubing functions to manage cube servers and cubes, import metadata, and optimize OLAP queries. The administration console shares much of the OLAP functionality with the InfoSphere Warehouse Design Studio. The OLAP models and cubes are created in Design Studio and the administration console manages them after they have been deployed.

9.4.1 Administration topologies and architecture

The administration console supports the administration of both local and remote cube servers, as shown in Figure 9-29.

Cube server administration can be performed for:

- Cube servers that are located on the host system where the administration console server is installed.
- Cube servers that are located on any system other than the host system, assuming that both systems are connected to each other.

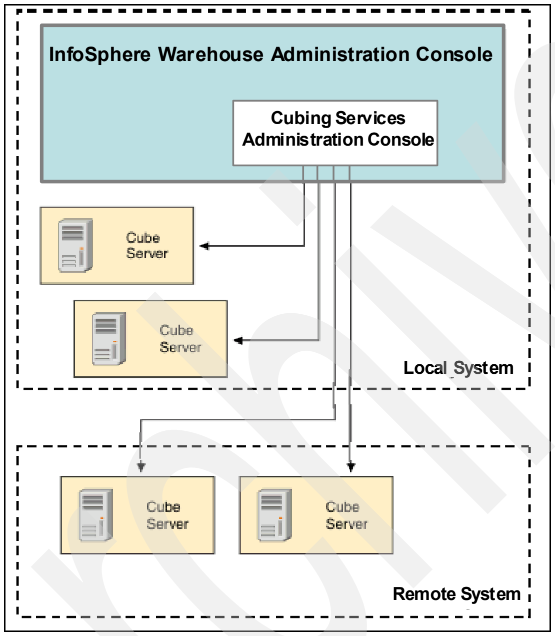

InfoSphere Warehouse Administration Console
Cubing Services Administration Console
Cube Server
Cube Server
Local System
Cube Server
Cube Server
Remote System

Figure 9-29 Administration console for Cubing Services topologies

The administration console web interface provides uniform access to the Cubing Services core components: the metadata database and the Cube Server repository. This is illustrated in Figure 9-30.

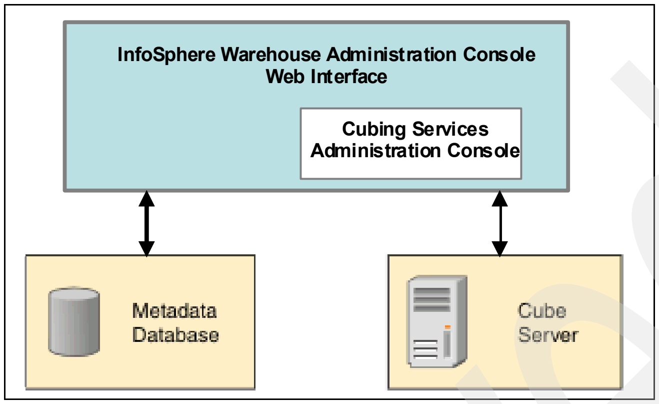

InfoSphere Warehouse Administration Console Web Interface
Cubing Services Administration Console
Metadata Database
Cube Server

Figure 9-30 Administration architecture for Cubing Services

9.4.2 OLAP deployment process

The steps involved in deploying cube models to the InfoSphere Warehouse production environment are contained in the Cubing Services section of the administration console. The following workflow describes the OLAP deployment and administration process:

1. Create the data source. This process was previously described in **"Configuring database connections" on page 370**.
2. Import the OLAP metadata. This step saves the cube model metadata into the InfoSphere Warehouse production database.
3. Configure a cube server.
4. Add cubes to the cube server.
5. Start the cube server.

Creating the data source

The InfoSphere Warehouse cube servers need a named data source connection that points to the underlying relational database for the cube. The data source needs to be defined before the OLAP metadata can be imported into the data warehouse. For more information about defining the data source for your cube server, see **"Configuring database connections" on page 370**.

Importing the OLAP metadata

After the cube models and cubes are defined in Design Studio, that metadata is exported to an XML file in preparation for deployment to the production database server. 6.6.14, "Exporting a cube model from Design Studio" on page 222 describes the process to generate the XML file.

The next step in the deployment process is to log in to the administration console and go to the Cubing Services tab. To launch the import OLAP metadata wizard, click **Manage Cube Models**, and click **Import Cube Model**. Answer the prompts as outlined in Table 9-5. Upon completion, the cube definition is deployed to the production database.

Table 9-5 Import OLAP metadata prompts

Page	Action
Map Metadata to Database	Choose your database from the list of database resources.
Specify Metadata XML File to Import	Browse for and select the xml file that you created from Design Studio.
Review File Content	Review the cube definitions that are contained in the file.
Start Import	Specify an appropriate replace option. Best practice: Select the "Do not import metadata if it affects a cube that is still in use" check box to avoid synchronization issues. Click **Finish** to complete the wizard.

Configuring a Cube Server

The Cubing Services page of the administration console provides the mechanism to create and manage both cube servers and cubes. Cube servers must be defined first, and the cubes associated with the cube server. To create a cube server, open the Cubing Services page of the administration console, click the **Manage Cube Servers** button, and click the **Create** button, which is designated by the diamond icon. Complete the prompts of the Create Cube Server wizard, which are listed in Table 9-6.

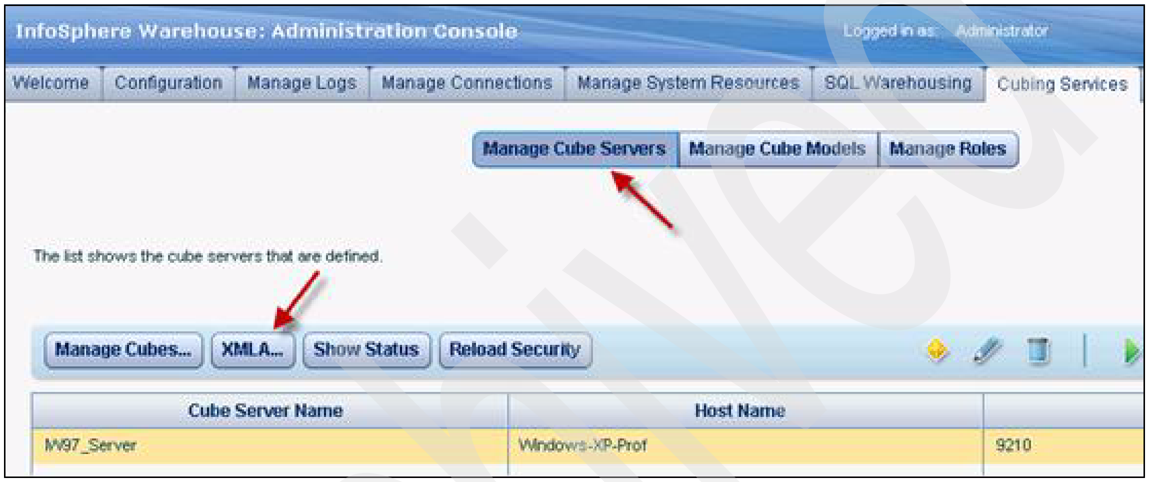
Table 9-6 Create Cube Server wizard

Page	**Action**
General	Specify a unique name for the cube server, select whether you want to start cubes when the cube server starts, and click Next.
Host Detail	1. Specify the hostname or IP address for this cube server. Note: You cannot specify *localhost* or *127.0.0.1* as the hostname. 2. Provide a port number, which can be any unused port. The combination of host name and port number must be unique. The cube server uses three or four consecutive ports, depending on your options. For example, if your cube server port is 9000, then 9000 and 9001 are used, the default XMLA port is 9002, and the default XMLA HTTPS port is 9003. A port cannot be used by more than one process; if any of the ports are already in use, the cube server will not start. 3. Select the connection type. If the cube server you create is on a remote host, you need to select the connection type that the remote host uses. 4. Test the connection before proceeding.
Cube Server Java VM Properties	Specify the Cubing Services installation path, and the cube server JVM information such as the Java path, the start and maximum heap size, and additional JVM arguments. The JVM arguments that you specify will be appended to the JVM command argument string, which appears in the Start command arguments field.

Page	Action
Performance and Security	Specify performance and security parameters for the cube server: ► Member cache file system location If you enabled member caching, specify a relative directory name on the WebSphere host for the member cache data. Tip: To increase cube scalability, dimension member information can be cached in the file system. If member caching is enabled, this file system location will be used as a temporary storage for all cubes that are currently being made available by this cube server. ► Maximum concurrent cubes Specify the maximum number of cubes that can run concurrently. A running cube can consume a significant amount of resources, which can decrease system performance. When the limit is reached, no cube can be started. ► Maximum concurrent MDX queries Specify the maximum number of concurrently processed MDX queries. The amount of resources that are used depends on the types of queries that are being issued. Many queries use little machine resources, but long-running queries might consume significant resources. Each operating system limits the number of threads that can run concurrently. For example, on Windows, the value you specify must be less than 10,000. ► Security level Select one of the four security levels
Common Logging Properties	Specify the logging settings that are common to all the loggers. For the Using rolling log field, choose whether to retain previously generated logs. If you select Yes, specify the log file size and how many copies to retain. The changes to the logging configuration take effect after this cube server is restarted (if it is currently running) or when it is started. The administration console does not provide log viewing capabilities for cube server files. The log files are stored on the machine where the cube server is running. They are stored in `\CS_HOME\CUBE_SERVER\Logs\`, where CS_HOME = Cubing Services installation directory (a subdirectory under the InfoSphere Warehouse installation directory) and CUBE_SERVER = the name of the cube server.
Logging details	Specify properties for specific loggers if you need to. Logging properties such as log format, display on console, and using rolling log are common in the logger's settings, but the logging level is different among loggers. With Performance Log, you can filter and collect performance statistic records based on keywords, such as MDX, member_cache_local, and member_cache_global. For example, setting the log level to member_cache_local logs local member cache details. If you set the level to member_cache_global, both local and global member cache details are added to the logger. You can select multiple levels.

Configuring the XMLA provider

When a cube server is defined, the XMLA provider is enabled by default. You can disable the XMLA provider, or change the port number after the cube server is created. By default, cube_server_port_number+2 is designated as the port number for XMLA service over HTTP. For example, if the port number of the cube server is 9000, then 9002 is the default port number for HTTP. To change the XMLA settings, select the Cubing Services tab, and select Manage Cube Servers. Select the cube server you want to work with, and click the **XMLA** button, as illustrated in Figure 9-31.



The list shows the cube servers that are defined.

Cube Server Name	Host Name	
IW97_Server	Windows-XP-Prof	9210

Figure 9-31 Configuring XMLA

When the cube server is not running, you can change the XMLA port, or disable the XMLA provider. When the cube server is running, you can use the same XMLA management view to start or stop the XMLA provider for a cube server, or query its runtime status.

Adding cubes to the Cube Server

Once the cube server is defined, and the metadata for the cube has been imported, you can associate cubes with it. Follow these steps to add a cube to a cube server:

1. Select the Cubing Services tab, and click **Manage Cube Servers**. A list of available cube servers is displayed.
2. Select the cube server from the list, then click **Manage Cubes**. A window is displayed with a list of cubes that are deployed to this cube server.
3. Click **Add**. A list of available cubes that can be added to the cube server is displayed.

4. Select the cube or cubes from the list and click **Add**. The list of cubes for the cube server is displayed again and includes the cube or cubes that you added to the cube server.

Note: To make the cubes available to OLAP clients, you must start the cubes that you added in this operation.

Starting the Cube Server

Now that the cube server has cubes associated with it, we are ready to start it. The steps to start a cube server are as follows:

1. Select the Cubing Services tab, and click **Manage Cube Servers**. A list of available cube servers is displayed.
2. Select one or more cube servers from the list, then click one of the icons to start, stop or restart the cube servers.

Table 9-7 lists the options for Cube Server management.

Table 9-7 Options for Cube Server management

Option	**Description**
Start(Green arrow icon)	This command starts one or more cube servers. ► Starting a cube server makes the cube server's cubes available to OLAP clients. ► Cubes that were added to a cube server will be automatically started if the option Start cubes when cube server starts is enabled for the selected cube server. ► Cubes might not be immediately available to OLAP clients after the cube server starts because cube startup might take several minutes for large cubes. ► If the WebSphere Application Server for the administration console is stopped, the cube server will continue to run. ► Log messages that are generated by the cube server during startup are logged in the administration console log file. Log messages that are generated by a cube server after startup completes are recorded in the cube server log files.
Stop (Red square icon)	This command stops one or more cube servers. ► All cubes on the selected cube server will be stopped and will no longer be available for OLAP clients. ► Log messages that are generated by the cube server during shutdown are logged in the administration console log file.
Restart (Blue circular arrow icon)	Restarts one or more cube servers

Note: If you shut down the application server, and therefore the administration console, any running cube servers are not automatically shut down because the cube server is independent from the application server.

9.4.3 Using the administration console to manage cube security

With InfoSphere Warehouse v9.7, security for Cubing Services can be controlled at the cube server level, or at level of the cube objects such as cubes, hierarchies, and facts.

Cubing Services users who query cubes are first authenticated by DB2. The cube server establishes user authentication by using the security configuration that you established for DB2, such as LDAP. Access to the cube and the objects of the cube is then authorized by the cube server security service. The cube server security service controls access to the cube and dimension members.

You can configure security in Cubing Services at two levels:

- Server-level security is user-based.

  Use the administration console to grant users permission to access to the cube server.
- Cube object-level security is role-based.

  Use Design Studio to specify the cubes, hierarchies, and facts that members of a role can access, and you use the administration console to assign roles to users.

Configuring server-level security

Four levels of security are implemented for a cube server. You can switch among these levels according to your needs. To configure cube security, perform the following steps:

1. Select the Cubing Services tab, and click **Manage Cube Servers**. A list of available cube servers is displayed.
2. Highlight the cube server you plan to work with, and click **Edit** on the toolbar.
3. Go to the Performance and Security page and select a security level. Security levels are shown in Table 9-8.

Table 9-8 Security levels for cube servers

Level	Explanation
Disabled	Security is disabled for this cube server. Users can query the active cubes on this cube server, regardless of the security settings of these cubes.
Enabled, authenticate for each request	Security is enabled for this cube server. Users must enter a user ID and password to connect to the cube server. After a connection is established, the credentials are authenticated each time you send a request to the server.
Enabled, authenticate for each connection	Security is enabled for this cube server. Users must enter a user ID and password to connect to the cube server. The credentials are valid until the connection is closed.
Enabled, authenticate for each refresh (Default)	Security is enabled for this cube server. Users must enter a user ID and password to connect to the cube server. The credentials are authenticated automatically at each refresh interval unless the cube server is stopped. The default refresh interval is 3600 seconds. If you specify a refresh interval of 0, the authentication credentials are valid until the cube server is stopped.

4. Click **Finish** to save the changes to the cube server properties. The changes will take effect when the cube server restarts.

Configuring cube object-level security

Security for cubes, hierarchies, and facts is defined in a security model which is developed in Design Studio. You must use both Design Studio and the administration console to configure role-based security. In Design Studio, you can design your security model to control which cubes and dimensions that your users can access when they run OLAP queries. In the administration console, you assign roles to your users, and import your security model for enforcement by the cube server and to configure cube object-level security:

1. Create a security model in Design Studio, and export the model to an XML file. This process is described in 6.7.2, “Dimensional security” on page 229.
2. Configure or define user roles, if necessary. To do so, select the Cubing Services tab, and click **Manage Roles**. Manage the roles from this page. You can use this page to:
   – Create, edit, or delete a role
   – View the privileges and users that are associated with a role
   – Assign users to a role
   – Remove users from a role
3. To activate the cube security, import the security model XML file by choosing the **Manage Cube Models** option from the Cubing Services tab, and selecting the **Import Security** option.
   a. Select the database connect that is associated with the cube, and click **Next**.
   b. Browse to the security model XML file location, and click **Next**.
   c. On the next window, map the roles from the security model to the user roles that are defined on the server, and click **Next**.
   d. On the Start Import page, specify how to handle conflicts between the security model and the currently active cube security, and click **Finish**.
4. Reload the security configuration for those cubes that are running:
   a. Select the Cubing Services tab, and click **Manage Cube Servers**.
   b. For each cube server that hosts a cube whose security you just modified, select the cube server and choose **Reload Security**.
5. Once the security configuration has been reloaded, the server begins enforcing the cube security settings.

Note: If you need to modify a security model that has already been activated, update the model in Design Studio, and follow the steps above to re-deploy it. When you import the updated security model, the changes override the existing model

9.4.4 Deploying a virtual cube

Chapter 6, “InfoSphere Warehouse Cubing Services” on page 169 described the benefits of virtual cubes. That chapter also provides information about defining virtual cubes and preparing them for deployment by exporting the virtual cube model to XML. To deploy the virtual cube to the production warehouse environment, we must then import the model through the administration console. The import model process is the same for both regular cubes and virtual cubes. See “Importing the OLAP metadata” on page 404 to review the process. After the virtual cube model has been imported, it needs to be added to a cube server. This step has been described in “Adding cubes to the Cube Server” on page 407. The last step in the process is to restart the cube server.

Note: Virtual cubes are not supported on System z.

9.4.5 Optimizing cube models

The benefits of MQTs and cube model optimization are described in Chapter 6, “InfoSphere Warehouse Cubing Services” on page 169.

In addition to Design Studio, the Optimization Advisor utility can also be executed from the administration console.

Note: The Optimization Advisor cannot be run from the administration console on System z.

To run the Optimization advisor, perform the following steps:

1. Select the Cubing Services tab, and click **Manage Cube Models**. A list of available cubes displays.
2. Click the **Advisor** button. The Optimization Advisor window opens with a list of running or completed Optimization Advisor recommendations.
3. Click the **Run Optimization Advisor** button (the green arrow icon) to launch the advisor wizard.
4. Select the cube model from the available list and click **Next**.

5. Specify the summary table options, as shown in Table 9-9, and click **Next**.

Table 9-9 Summary table parameters

Options	**Explanation**
Summary table update option	**Deferred** If you choose this option, the summary tables are updated less frequently. The summary tables are based on a snapshot of the data at the time that the summary tables are created. Each update recreates the summary table based on the current data, but has no knowledge of how the data changed since the summary table was last created. **Immediate** If you choose this option, the summary tables are continually synchronized with your base tables. DB2 tracks the changes to the base tables so that it can incrementally update the summary tables by changing only the portion of the summary tables that corresponds to the changed portion of the base tables. If you need to synchronize the summary tables with the base tables often, use the Immediate option. For example, if your base tables are regularly updated, use the Immediate option.
Table space for explain tables	Select a target table space for the summary tables from the list.
Table space for indexes	Select the index tablespace that will be used when the DDL is generated.

6. Specify disk space and time limits for the Optimization Advisor, as outlined in Table 9-10, and click **Finish**.

Table 9-10 Optimization Advisor disk space and time limit options

Options	**Explanation**
Disk space limit	**Unlimited** Select this option if you don't want to set the space limit. **Limited** If you choose this option, type the maximum available disk space in the disk space field and select either MB or GB.
Time limit	**Unlimited** The Advisor might run for several hours if no limit is specified. **Limited** If you choose this option, type the maximum amount of time in the maximum time field and select either Hours or Minutes. The time limit is the maximum amount of time that you allow the advisor to run.
Data Sampling	Select the Data Sampling check box to sample the data from the tables that are described by the metadata. Sampling data can potentially improve the performance of large queries. It is recommended to select the check box.

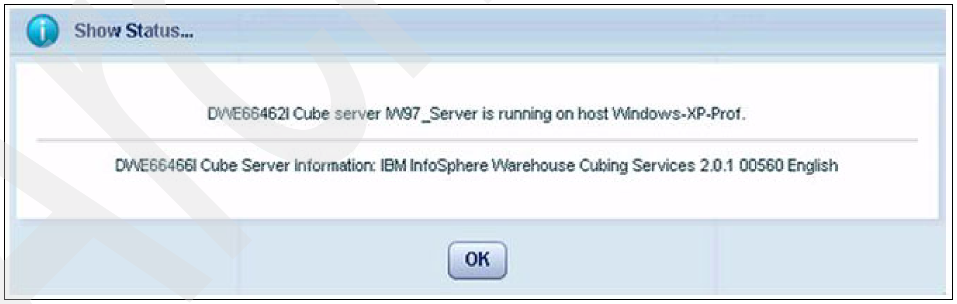
7. In the Optimization Advisor window, select the recommendation that you just created in the list, and click the **View Advisor Results** button to view the results. You can click either the **Save SQL** script or **Save Refresh** script link to initiate a download operation to your machine that allows you to view or save the respective file. Save these files to create or refresh summary tables.
8. To implement the Optimization Advisor recommendations, run the resulting SQL script. From the Manage Cube Models view, highlight the cube model and click **Run SQL Script**. Browse to the script file and click **Next**. The script is executed.

9.4.6 Management of the Cubing Services environment

Management of the Cubing Services environment is done through the administration console. Use the administration console for cube server management, cube model management, managing cube performance, and to manage cube security.

Cube server management

You can use the administration console to create, remove, start, stop, or restart cube servers. Additionally, you can view the status or properties of a selected cube server. See Table 9-7 on page 408 for instructions on starting, stopping, and restarting cube servers. You can also delete a cube server from this view. To monitor the cube server status, go to the Manage Cube Servers view, select the cube server of interest, and click the **Status** button. Cube status is depicted in Figure 9-32.



DWE66462I Cube server MW97_Server is running on host Windows-XP-Prof.

DWE66466I Cube Server Information: IBM InfoSphere Warehouse Cubing Services 2.0.1 00560 English

Figure 9-32 Checking Cube Server status

Configuring log services for cube servers

Cube server log types and log parameters are defined through the administration console. You can configure logging services for the following log types:

- Activity log

  This log records regular status information.

- Performance log

  This log records performance statistics.

- Exception log

  This log records diagnostic information related to failures.

- Trace log

  This log records debugging information.

- MDX log

  This log records MDX queries that were sent to the cube server for processing.

- SQL log

  This log records SQL queries that were generated based on MDX queries and sent to the cube server for processing.

- XMLA log

  This log records information about requests that are sent to the XMLA provider.

Select a cube server from the Manage Cube Servers view, and click **Edit**. Click **Next** until you come to Step 5; Common Logging Properties. The logging property options are described in Table 9-11.

Table 9-11 Logging Property options

Option	Description
Log format	Select XML, HTML, or Text (default).
Display on console	If the cube server is started in standalone mode, select this option to view the log as the log information is being captured on the command line console.
Use rolling log	Select this option to indicate that you want to retain previously generated logs. If you select this option, enter the log file size in the Maximum log file size field and how many copies to retain in the Maximum number of files saved field.

Click **Next** to get to the Logging Details window. The logging detail options are described in Table 9-12.

Table 9-12 Logging detail options

Option	Description
Log Level: Activity log	For the Activity Log, you can select from the following options: ► All: Records all activity ► Info: Records only information messages ► Warning: Records only warning messages ► Error: Records only error messages ► Fatal: Records only fatal messages ► Off: Suspends logging
Log Level: MDX log, XMLA log, SQL log	For the MDX Log, the XMLA Log, and the SQL Log types, you can select from the following options: ► All: Records all activity ► Off: Suspends logging
Log Level: Performance log	With Performance log, you can filter and collect performance statistic records based on keywords, such as MDX, Member Cache Local, and Member Cache Global. For example, if you set the log level to Member Cache Local, the local member cache details are logged. If you set the level to Member Cache Global, both local and global member cache details are added to the logging. You can select multiple levels from the following options: ► MDX: Filters performance records on the keyword MDX ► Member Cache Local: Filters performance records on the keyword Member Cache Local ► Member Cache Global: Filters performance records on the keyword Member Cache Global
Log Level: Trace log	For the Trace Log, you can select from the following options: ► All: Records all activity ► Fine: Records summary trace information ► Finer: Records high level trace information ► Finest: Records detailed trace information ► Off: Suspends logging

Changes to the logging options take effect with the next restart of the cube server.

Note: The administration console does not provide log viewing capabilities for cube server files. The log files are stored on the machine where the cube server is running. They are stored in `\CS_HOME\CUBE_SERVER\Logs\`, where `CS_HOME` is the Cubing Services installation directory (a subdirectory under the InfoSphere Warehouse installation directory) and `CUBE_SERVER` is the name of the cube server.

Managing cubes

Similar to cube server management, you can use the administration console to create, remove, start, stop, or restart cubes. Additionally, you can view the status or properties of a selected cube. Perform the following steps:

1. Select the Cubing Services tab, and click **Manage Cube Servers**. A list of available cube servers is displayed.
2. Select a cube server, and click **Manage Cubes**. A list of cubes that are deployed on this cube server is displayed.
3. Select one or more cubes and use the toolbar to edit remove, start, stop, or restart the cube.

Managing cube performance

The InfoSphere Warehouse Cubing Services feature provides cache capabilities to improve the performance of a cube. There is a data cache and a member cache that are available. The more data that is stored in the caches, the less often queries to the cube need to retrieve results from the underlying database, providing faster query response time. However, if the cache grows too large, it uses memory on the machine, potentially slowing the performance for all users. To determine the optimal size of the cache, test and experiment with the settings. The cache settings are accessible through the properties page of a particular cube.

Data cache

The data cache stores cube cells fetched from the relational database. After loading into the data cache, the stored data is shared among concurrent and subsequent queries when available.

Member cache

The member cache stores dimension metadata (members) and can be tuned to either completely or partially cache members. For member caching, there are two modes available:

- Static caching (default)
- Dynamic caching

When static cache is configured, the dimensional members are read from the relational source and loaded into memory at cube startup. When dynamic cache is configured, the dimensional members are read from the relational source and stored in compressed data files housed in a user-specified location. When the cube is started, the member cache files are created and any previous files are overwritten. While the cube is running, dimensional members are read into memory from the compressed dimension files on an as-needed basis.

Configuring data cache and member cache

To configure a cube for data cache and member cache, perform the following steps:

1. Select the Cubing Services tab, and click **Manage Cube Servers**. A list of available cube servers is displayed.
2. Select a cube server, and click **Manage Cubes**.
3. Select a cube and click **Edit**. A description of the options when editing cube properties is shown in Table 9-13.

Table 9-13 Options when editing cube properties

Options	**Explanation**
Enable this cube	Select this option to enable the cube for analysis. An enabled cube can be started, which makes the cube available to OLAP clients. An enabled cube is started automatically if you selected the **Start cubes when cube server starts** option when you created the cube server and the current number of enabled cubes on this cube server does not exceed the setting for the **Maximum number of concurrently running cubes** option. **NOTE**: A cube is enabled by default. Disabling a cube does not stop it if it is currently active.
Refresh settings	Select **Automatically refresh cube caches** to refresh the cache after the cube's data is updated. Cache refresh is performed only if the cube server is running. For example, you should refresh the cache after the cube data has been updated, such as by an SQW process. After this option is enabled, you can specify a value for hours and select a number for minutes from 0 to 59. For example, if you select 5 hours 10 minutes, the cube cache is cleared on this cube server 5 hours and 10 minutes after the cube server was started and is cleared every 5 hours and 10 minutes subsequently. **NOTE**: The automatic refresh option is enabled by default.
Member cache settings	Select **Enable dynamic caching** to enable dynamic caching. If you select this option, you can select one of the following options: ▸ Do not limit the number of cached members ▸ Limit the number of cache members to: If you select this option, you can specify the number by which to limit the cache members in the members field.

Options	Explanation
Data cache settings	You can select one of the data cache settings: ► Cache all cells ► Cache a limited number of cells: If you select this option, you can specify the number by which to limit the cache cells in the cells field. A cube's data cache is empty after the cube is started. If you want to pre-populate the data cache, you can select Populate cache using the following MDX query, and specify an MDX query that selects data cells.

4. Click **OK** to save the settings. The changes take effect when the cube is restarted.

You can rebuild member cache for a running cube. This process implicitly empties the data cache for the cube. From the Manage Cube window, highlight the cube and click **Rebuild member cache**. You can also empty the data cache by highlighting the cube and clicking **Empty Data Cache**.

Analyzing summary table usage

After you have optimized the OLAP metadata described in 9.4.5, "Optimizing cube models" on page 412, and have deployed the suggestions made by the advisor, you can want to analyze the summary table usage for your queries.

Note: Existing history must be available in the Explain tables of optimized queries before you can analyze summary table usage. The Explain tables are populated and the optimized information is captured if the following circumstances exist:

- The Explain tables exist and are configured appropriately for the user ID. For more information, see the IBM DB2 Information Center.
- An applicable summary table exists.
- An applicable query is run with explain mode set to populate the Explain tables.

To analyze the summary table usage for a query, perform the following steps:

1. Select the Cubing Services tab, and click **Manage Cube Model**.
2. Select the cube model and click **Explain Table Usage**.
3. Select one of the Explain plan options shown in Table 9-14 to specify the Explain data source and define the parameters that are applicable to the option that you select:

Table 9-14 Explain Plan Options

Option	Description
Use existing Explain data	If the Explain tables were populated from previously executed SQL queries, select **Use existing Explain data** and complete the following steps: 1. Click **Next**. 2. Select a beginning date and time and an ending date and time. 3. Enter the name of the explain table schema to be used to in the analysis in the Explain table schema field. For example, if the explain tables were created under a schema named myschema, you would enter `myschema`. Explain tables can be created under any schema name. 4. Click **Next**. The number of queries that were run in explain mode during the time specified are displayed. It also shows the number of queries that used a particular summary table and how much each summary table was used based on the total number of queries that were analyzed.
Generate new Explain data	To select specific queries to be evaluated, select **Execute Queries**. This option runs a script that contains queries in explain mode to evaluate the analysis results and populate the Explain tables with the information. 1. Click **Next**. The “Execute SQL Queries” page opens. 2. Click **Browse** to select a file that is already created on your computer that contains these queries. 3. Click **Next**.

4. In the Review non-optimized queries page, you can view the MQTs that were applied and the number of queries analyzed. If queries were not rerouted to any of the existing relevant summary tables, the queries are listed in the “Review non-optimized queries” window. Review any queries listed that were not rerouted to any of the existing relevant summary tables.

If queries are displayed in this panel, review the following conditions:

- The summary tables that are being used. The advisor process might recommend creation of certain summary tables that do not improve performance for queries that you can issue; you can then drop those summary tables because there is no benefit to maintaining them.
- The queries that are not rerouted to a summary table. Changing one or more advisor parameters might cause additional summary tables to be created that might improve performance for these queries.

Click **Next**.

5. On the "Review summary table usage" page, review the summary table usage for the analyzed queries. Click the summary table check box and click **Next**.
6. On the "Review optimized query details" page, review the queries that are rerouted to a particular summary table. Click **Finish** to close the wizard.

9.5 Deploying a data mining application

The business value behind data mining and its implementation in InfoSphere Warehouse is discussed in Chapter 7, "Data mining" on page 243. This section introduces the data mining features present in the administration console. The mining model management section in the administration console allows the user to view, export, update, and delete models in the mining-enabled database. Mining models can also be imported into the mining database and loaded into a cache for quicker retrieval in the future. The mining visualization tool in the administration console further provides a graphic representation of the results of the mining model.

Note: Data mining is not supported on System z.

9.5.1 Deploying and managing the data mining models

The ability to import and manage the mining models in the administration console is important, as they can be used in a variety of ways. Mining models can be embedded into a mining flow, which can be a part of a deployed data warehouse application. An example is where new data is processed through the database tables. This data is scored against an existing mining model. For this application to run in a production environment the mining model needs to have been deployed to the appropriate database. Mining models can also be used in a

Java class as part of a Web-based application. For this application to score or visualize the data, the mining model to needs to be deployed again into the correct production database.

Deploying the mining application

Before a data mining application can be deployed, recall that the production database must be enabled for mining. See "Enabling defined databases for mining" on page 375 for instructions on database enablement.

The process to deploy a data mining application is similar to the SQL warehousing deployment process. The first step is to prepare the application for deployment in Design Studio:

1. In the Data Project Explorer, right-click the project and choose **New** → **Data Warehousing Application**.
2. Answer the prompts, using Table 9-15 as a guide.

Table 9-15 Application deployment prompts from Design Studio

Page	**Steps**
Project Selection	Select your data mining project
Application Profile	In the Profile Name field, type a name for the application. You can also provide a description.
Control Flow Selection	Select the control flows that are needed and click **Next**.
Resource Profile Management	Accept the default.
Variable Management	Accept the default.
Miscellaneous File Selection	Accept the default.
Saving Application Profile	Accept the default.
Code Generation	Accept the default.
Package Generation	Specify a location in the Zip file directory

3. Deploy the mining compressed file through the SQW tab. See "Installing the deployment package" on page 156 for instructions. The application is now available on the production server.

After you deploy the data mining application to your production server, your data mining application can maintain tables that contain association rules that are based on live transaction data. See "Scheduling control flows" on page 163 for instructions on scheduling the mining control flow.

Managing data mining models

There are several administrative functions that can be performed on the data mining models.

Viewing mining models

To view the data mining models that have been deployed to the data warehouse, perform the following steps:

1. Click the Mining tab and click Manage Mining Models.
2. Select the appropriate database from Connection Name drop-down box.

   A list of the available mining models are displayed
3. Select a model by highlighting it and click **View Results**. A second web browser window opens and displays the Intelligent Miner Visualizer, as shown in Figure 9-33.

   For an explanation of how to work with the visualizer, see 7.5.9, “Visualizing models and test results” on page 305.

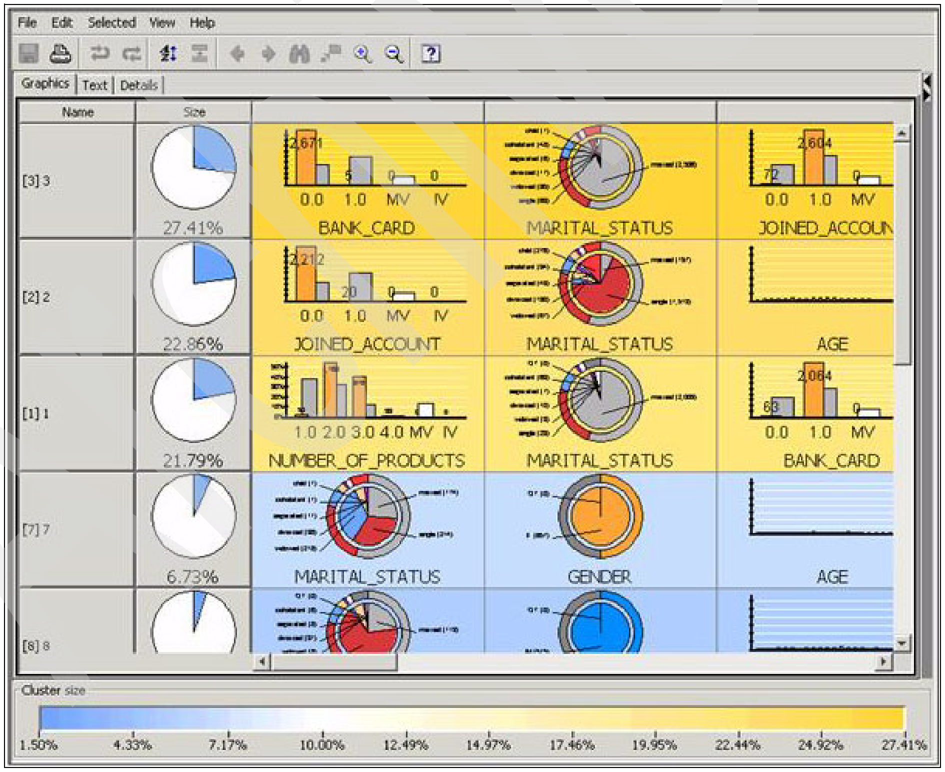
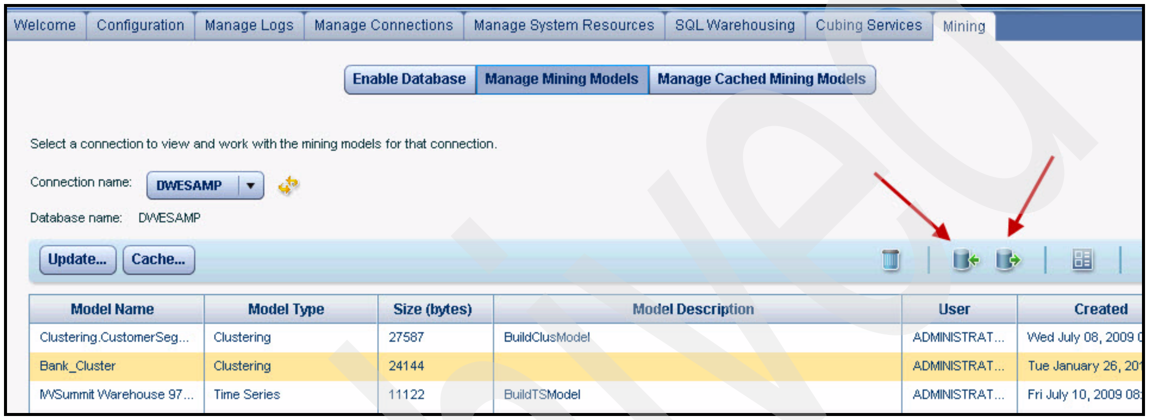


Figure 9-33 The Intelligent Visualizer launched from the administration console

Importing and exporting mining models

You might want to exchange data mining models between databases, or exchange them with vendor applications such as mining workbenches from SAS or SPSS, or archive a mining model outside of the database in a source code control system. InfoSphere Warehouse provides the ability to import and export models in support of these needs.

From the list of deployed models in the Manage Mining Models view, click either the **Import** button or the **Export** button, as illustrated in Figure 9-34.



Select a connection to view and work with the mining models for that connection.

Connection name: DWESAMP

Database name: DWESAMP

Model Name	Model Type	Size (bytes)	Model Description	User	Created
Clustering.CustomerSeg...	Clustering	27587	BuildClusModel	ADMINISTRAT...	Wed July 08, 2009 0
Bank_Cluster	Clustering	24144		ADMINISTRAT...	Tue January 26, 20
IWSummit Warehouse 97...	Time Series	11122	BuildTSModel	ADMINISTRAT...	Fri July 10, 2009 08:

Figure 9-34 Importing and exporting data mining models

Updating and deleting mining models

Data mining models might be updated with a new model file name or description. Highlight the model you want to work with, and click **Update**.

To delete a mining model permanently, highlight the model and click the delete icon (trash can). Deleted models cannot be recovered.

Mining model caching

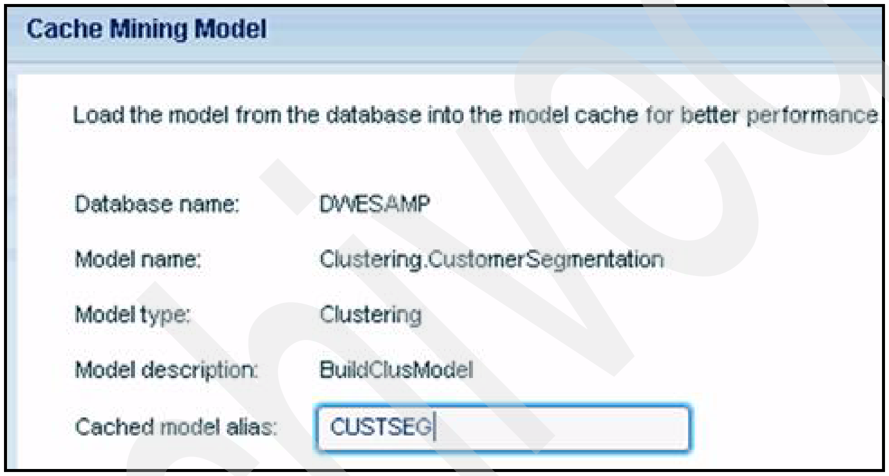
Data mining models might be cached in memory to improve the performance of the scoring processes. Applications that perform real-time scoring are good candidates for model caching.

Note: The database must have been enabled in fenced mode to support model caching.

To load a mining model into cache, perform the following steps:

1. Go to the Manage Mining Models view.
2. Select the database, and highlight the model you want to load into cache.
3. Click the **Cache** button.
4. Provide an alias name. The alias is the name of the model to be used in the Scoring SQL function.
5. Click **OK**.

A sample cache mining model is depicted in Figure 9-35.



Cache Mining Model

Load the model from the database into the model cache for better performance

Database name: DWESAMP

Model name: Clustering.CustomerSegmentation

Model type: Clustering

Model description: BuildClusModel

Cached model alias: CUSTSEG

Figure 9-35 Loading a data mining model into cache for real-time scoring

For more information about caching mining models, and managing cached models, see the following web pages:

- http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.datatools.datamining.doc/cdweacmmcache_rt_scoring.html
- http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.datatools.datamining.doc/sdweacmmmanagecached2.html

10

Workload management

Note: This chapter applies only to Linux, UNIX and Windows environments.

The purpose of workload management is to ensure that limited system resources are prioritized based on the needs of the business. Work designated as being of the most importance to the business are given the highest priority access to system resources.

In a data warehouse environment this means that online query users can expect predictable and acceptable response times at the possible expense of long running batch extract jobs.

It is possible to monitor and manage the work everywhere from the application through the network, the DBMS, the operating system and the storage sub-system. In this chapter we deal primarily with the workload management capabilities of the database with optional integration with the resource management capabilities of UNIX and Linux.

Workload management is a continuous process of design, deployment, monitoring and refinement. In a data warehouse environment business priorities and the volume and mix of work are constantly changing. This change must also be reflected in the management of resources.

DB2 Workload Manager (WLM) is the primary tool available for managing an InfoSphere Warehouse database workload. It was introduced with DB2 for LUW version 9.5 and enhanced with DB2 for LUW version 9.7.

In prior releases Query Patroller and the DB2 Governor were used to manage workloads. These tools were useful but additional requirements were identified and WLM was designed to address those additional requirements.

The database work described in this chapter includes almost all of the activity that can run on a data server including SQL data manipulation queries, data definition requests, and utilities such as LOAD.

InfoSphere Warehouse version 9.7 introduced the ability to design and implement a simplified sub-set of DB2 WLM functionality through the administration console. In this chapter we concentrate on describing the use of this tool, but also provide an introduction to the capabilities of the underlying DB2 WLM.

Many InfoSphere Warehouse users will find that the workload management capabilities of the administration console GUI tool are sufficient for their needs. Those with large, complex query workloads or special requirements have access to the WLM command line interface that allows the administrator to take advantage of its full functionality.

10.1 Workload management introduction

Workload management has four clearly defined stages:

- Definition of business goals
- Identification of the work entering the data server
- Management of the work when it is running
- Monitoring to ensure that the data server is being used efficiently

Workload management for a data warehouse is a continuous process, because workloads and requirements change. Performance must be monitored to ensure that requirements continue to be met.

In a data server environment, you must define goals. Sometimes the goals are clear, especially when they originate from service level agreement (SLA) objectives. For example, queries from a particular application should run in under 30 seconds. Goals can also be tied to a particular time of day. For example, overnight batch ETL work might have to complete by 8 a.m. so that the daily sales reports are on time. In other situations, the goals can be difficult to quantify. A goal might be to keep the database users satisfied and to prevent aberrant database activity from hampering day-to-day work.

DB2 WLM is available on all platforms supported by DB2 9.5, or later, for Linux, UNIX, and Windows. The optional tight integration offered between DB2 service classes and operating system service classes is available with AIX (starting with DB2 9.5) and Linux (starting with DB2 9.7).

DB2 WLM represents a shift of emphasis in workload management strategy to focus primarily on the ability to monitor and control active work after it has entered the DB2 execution environment. To provide the degree of control and monitoring desired for higher volumes of concurrent work requests while they are actually executing, yet with minimal overhead, workload management technology has been incorporated into the DB2 engine infrastructure.

DB2 WLM introduces an independent approach to workload management and does not rely on or interact with Query Patroller or DB2 Governor in any way. While Query Patroller and DB2 Governor are still functional, they are deprecated. They are no longer central to DB2 workload management strategy and no further investment is planned for them in future releases.

The stages of workload management are shown in Figure 10-1.

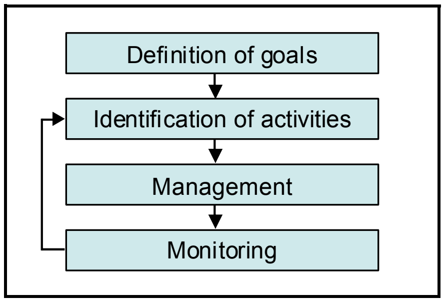


Figure 10-1 WLM stages

10.1.1 Definition of goals

Understand the work that needs to run on your data server and how it relates to your business commitments. If formal SLAs exist, make sure that the work associated with those specific agreements can be identified. More frequently there will be something less formal, where you are able to set priorities in broad ranges of LOW, MEDIUM, and HIGH.

As an alternative, if clear business requirements do not exist, approach the problem of priority assignment from a pure database perspective. Classify the work by the expected impact on the server using a short to medium to long classification system in which you treat short requests as being the most important and long ones as least important. In this paradigm, the thinking is that the more short queries that you can push through your system, the better the service provided to the enterprise as a whole.

Identification of activities

Identify the types and components of the workload, such as those having different performance or monitoring objectives. Examples of possible activity groupings are:

- Queries from specific user departments
- Queries from a specific application
- Batch data extracts
- ETL processes
- DDL work by the DBA

Management

Determine whether your system has the capacity to run this work without any restrictions. You can only put through as much work at any one time as your system can handle. Divide the work to run on your system as necessary to have the work execute based on its priority to the business without exceeding your system's capabilities.

Work can be managed in the following ways:

- Queuing queries

  Set a maximum concurrency level for a query type and make queries that exceed the limit wait until a running query has come to completion.

- Setting resource priorities

  When one segment of the workload is allocated a higher priority it is more likely to obtain the resources it needs to continue.

- Aborting jobs that exceed a preset limit

  Setting a fail safe to kill queries that might have been submitted in error and run for too long.

Monitoring

Monitoring consists of real-time operational monitors to show the state and health of the system in addition to gathering data for analysis and refinement.

10.1.2 An approach to implementing workload management

As with any new capability, it is usually not a good idea to implement a complex design or to make many different changes at the same time. Complexity and churn often hide the real problem and sometimes introduce new problems.

In the case of the control capabilities of DB2 workload management, start with a simple, straight-forward implementation and evolve it one change at a time. This approach allows you to understand and appreciate the impact of each change, as well as detect any unintended consequences or interactions.

It is best to keep the scope of any controls as narrowly focused as possible, as opposed to applying them globally. Broad controls might be desired to govern the global behavior of the system but keeping the control focused on the problem areas ensures that you see only the effects that you intended.

As an aid in taking this approach InfoSphere Warehouse includes GUI-based workload management design and deployment tools focussed on typical Data Warehouse requirements and simplifying deployment.

10.2 WLM concepts

DB2 WLM introduces a number of new concepts which should be understood to make best use of its functionality.

The following list details and explains basic concepts behind DB2 WLM:

- Workloads

  A workload is an object that is used to identify incoming work based on its source so that it can later be properly managed. The workload attributes are assigned when the workload establishes a database connection. Examples of identifiers for workloads are the application name, system authorization ID, client user ID, and connection address.

- Work classes and work class sets

  The type of work can be identified through work classes (which are grouped into a work class set). Examples of work types that can be identified using work classes include READ (such as SQL select and xquery), WRITE (such as SQL insert, update, delete and merge), CALL, DML, DDL, LOAD, and ALL.

- Service class

  The purpose of a service class is to define an execution environment in which the work can run. This environment can include available resources and various execution thresholds. When you define a workload, you must indicate the service class where work associated with that workload runs. There is always a default workload, which ensures that all data server work is running inside a service class. Service classes consist of a two level hierarchy with a service superclass containing service subclasses, which is where the work actually runs.

- Thresholds

  Thresholds permit you to maintain stability in the system. You create threshold objects to catch work that behaves abnormally, either predictively (before the work begins running based on the projected impact) or reactively (as it is running and consuming resources). An example of work that can be controlled with thresholds is a query that consumes large amounts of processor time at the expense of all other work running on the system. Such a query can be controlled either before it begins executing, based on estimated cost, or after it has begun executing and is consuming more than the permitted amount of resources.

- Threshold actions

  The kind of actions that can be taken when a threshold is exceeded depends on the threshold itself and includes the following actions:

  – Collect data
  – Stop execution
  – Continue execution
  – Queue activities

- Work actions and work action sets

  A work action provides an action that can be applied to a work class. For a work class to be active and have activities assigned to it, there must be a work action defined for the work class. A work action set can contain one or more work actions that can be applied to activities in either a specific service superclass or to the database as a whole as follows:

  – If you apply a work action set to a database they could define thresholds, prevent execution, collect data, and count activity.

  – If you define the work action set for a service superclass they could map activities to a service class, prevent execution, collect activity or aggregate data, and count the activities. Typically, the work action set maps an activity to a service subclass and has thresholds defined on the subclass to help manage the activity.

10.2.1 Workload management monitoring

Workload management monitoring allows you to access real-time operational data (such as a list of running workload occurrences and the activities running in a service class or average response times) by using table functions to access in-memory data. Event monitors capture detailed activity information (per service class, work class, or workload or when a threshold is exceeded) and aggregate activity statistics for historical analysis.

Real-time operational monitoring

Access real-time monitor data through DB2 table functions. Table functions provide access to a set of data that exists inside a DB2 database (such as the workload management statistics) as a virtual DB2 table against which you can execute a SELECT statement. This offers you the ability to write applications to query data and analyze it just as if it were any physical table on the data server.

General statistical information is available at a number of different levels:

- Service superclass and service subclass
- Workloads
- Work action sets
- WLM queues

In addition to statistics, there are table functions that provide information about the work that is currently executing on a system. This information is available at various levels:

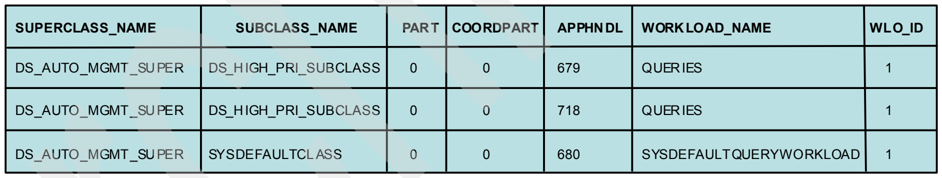
- Workload occurrence
- Service class agents
- Workload occurrence activities
- Activity details

Example 10-1 shows the SQL required to list current workload occurrences.

Example 10-1 Example SQL using WLM Table Function

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME
       , SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME
       , SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART
       , SUBSTR(CHAR(COORD_PARTITION_NUM),1,4) AS COORDPART
       , SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL
       , SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME
       , SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
    FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97
        (CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2)) AS SCINFO
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART, APPHNDL, WORKLOAD_NAME, WLO_ID
```

Figure 10-2 shows the sample output of Example 10-1.



SUPERCLASS_NAME	SUBCLASS_NAME	PART	COORDPART	APPHNDL	WORKLOAD_NAME	WLO_ID
DS_AUTO_MGMT_SUPER	DS_HIGH_PRI_SUBCLASS	0	0	679	QUERIES	1
DS_AUTO_MGMT_SUPER	DS_HIGH_PRI_SUBCLASS	0	0	718	QUERIES	1
DS_AUTO_MGMT_SUPER	SYSDEFAULTCLASS	0	0	680	SYSDEFAULTQUERYWORKLOAD	1

Figure 10-2 Example real-time monitor output

Historical monitoring

DB2 WLM uses event monitors to capture historical information for analysis. Event monitors for DB2 WLM include:

- Activity event monitor

  This monitor captures information about individual activities in a service class, workload, or work class or activities that violated a threshold.

- Threshold violations event monitor

  This monitor captures information when a threshold is exceeded. It indicates what threshold was exceeded, the activity that was the source of the exception, and what action was taken when it occurred.

- Statistics event monitor

  This monitor serves as a low-overhead alternative to capturing detailed activity information by collecting aggregate data (for example, the number of activities completed and average execution time).

Note: There is no overhead for unused WLM event monitors. Create them so that individual workloads, service classes and work actions can be altered to capture events when needed.

To store the events in DB2 tables, see the sample DDL in `~/sqllib/misc/wlmevmon.ddl`

For examples of perl WLM analysis scripts see `~/sqllib/samples/perl/wlmhist.pl` and `wlmhistrep.pl`

10.3 WLM configuration

In this section we describe how to configure the WLM by using the administration console. The WLM configuration utility is selected from the main window of the InfoSphere Warehouse Administration Console, as shown in Figure 10-3.

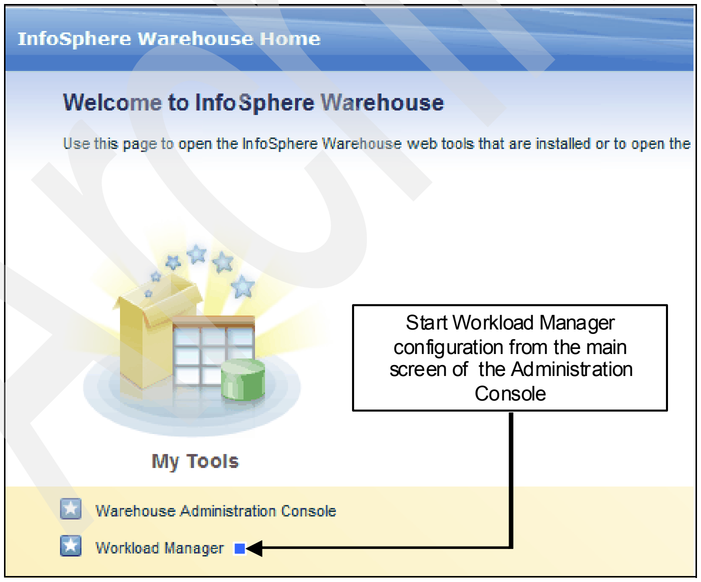
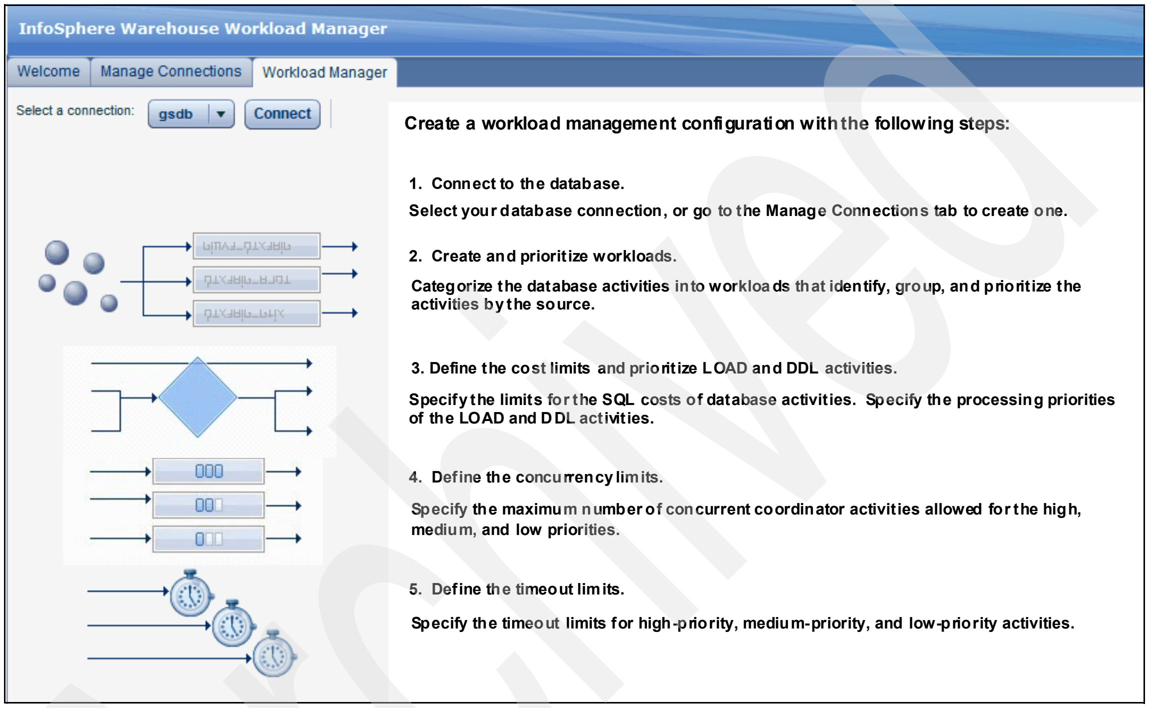

InfoSphere Warehouse Home
Welcome to InfoSphere Warehouse
Use this page to open the InfoSphere Warehouse web tools that are installed or to open the
Start Workload Manager configuration from the main screen of the Administration Console
My Tools
Warehouse Administration Console
Workload Manager

Figure 10-3 InfoSphere Warehouse Administration Console

The initial WLM window (Figure 10-4) provides an explanation of the steps to be followed in setting up the configuration. This includes the following steps:

1. Connect to the database.
2. Create and prioritize workloads.
3. Define the cost limits and prioritize LOAD and DML activities.
4. Define the concurrency limits.
5. Define the timeout limits.



Create a workload management configuration with the following steps:

1. Connect to the database.

Select your database connection, or go to the Manage Connections tab to create one.

2. Create and prioritize workloads.

Categorize the database activities into workloads that identify, group, and prioritize the activities by the source.

3. Define the cost limits and prioritize LOAD and DDL activities.

Specify the limits for the SQL costs of database activities. Specify the processing priorities of the LOAD and DDL activities.

4. Define the concurrency limits.

Specify the maximum number of concurrent coordinator activities allowed for the high, medium, and low priorities.

5. Define the timeout limits.

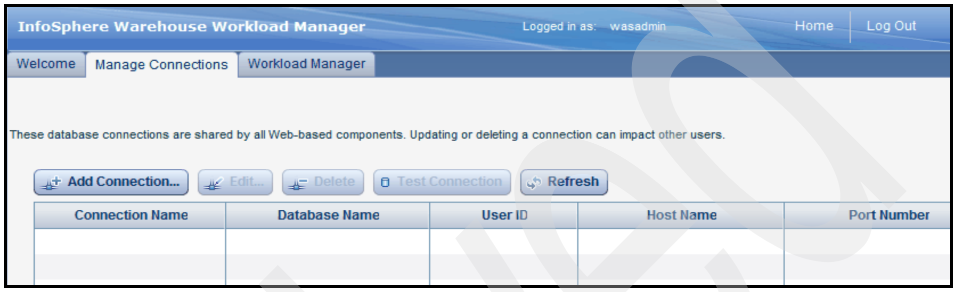
Specify the timeout limits for high-priority, medium-priority, and low-priority activities.

Figure 10-4 WLM configuration overview

10.3.1 Connecting the WLM database

Perform the following steps to connect the WLM database:

1. Make sure that there is a connection defined for the database running the workload. Select the Manage Connections tab from the configuration overview window to display the blank connections summary window (Figure 10-5).

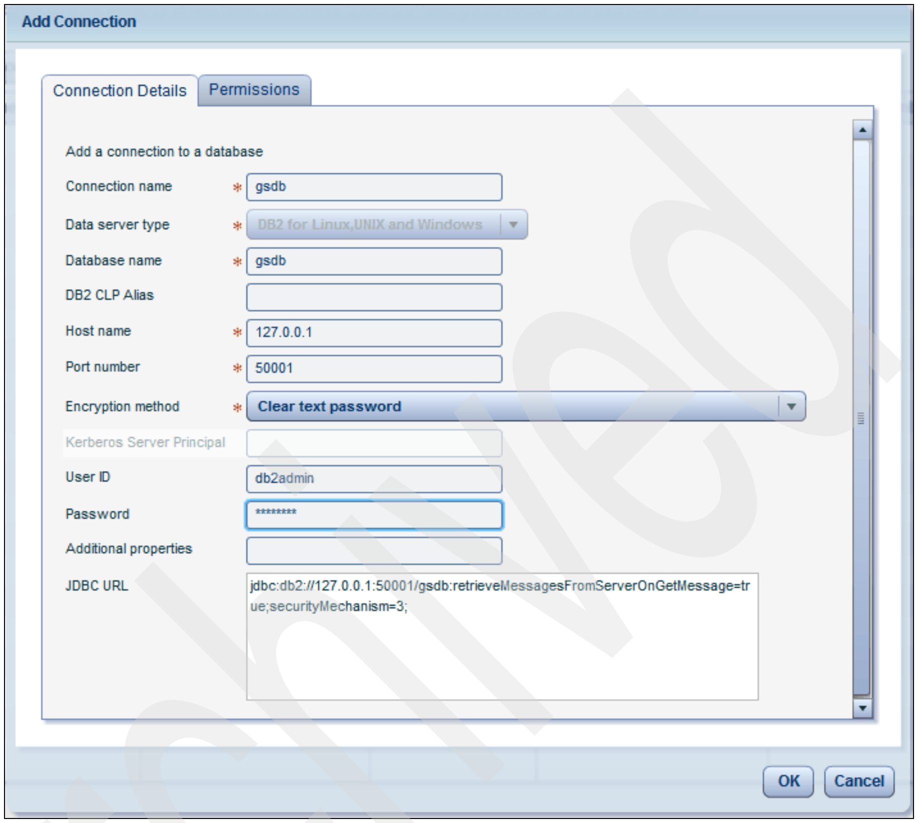


These database connections are shared by all Web-based components. Updating or deleting a connection can impact other users.

Connection Name	Database Name	User ID	Host Name	Port Number

Figure 10-5 WLM database connection summary

2. Click the **Add Connection** button. The “Add Connection” window (Figure 10-6) opens. Enter the connection details for the database.

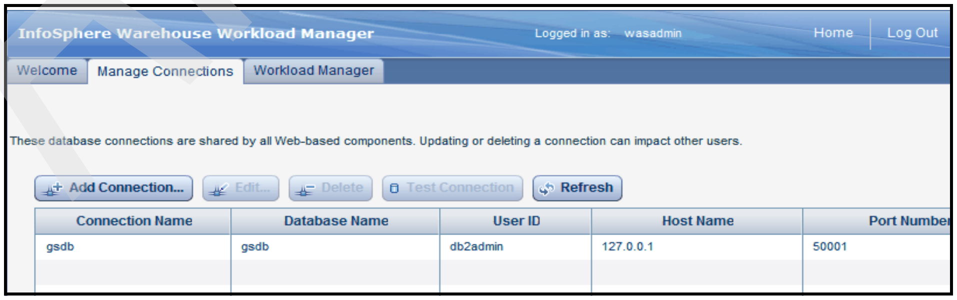


Add a connection to a database

Connection name *	gsdb
Data server type *	DB2 for Linux,UNIX and Windows
Database name *	gsdb
DB2 CLP Alias	
Host name *	127.0.0.1
Port number *	50001
Encryption method *	Clear text password
Kerberos Server Principal	
User ID	db2admin
Password	********
Additional properties	
JDBC URL	jdbc:db2://127.0.0.1:50001/gsdb:retrieveMessagesFromServerOnGetMessage=true;securityMechanism=3;

Figure 10-6 Database connection entry

3. Selecting the **OK**. The summary window (Figure 10-7) is shown again with the new connection.



These database connections are shared by all Web-based components. Updating or deleting a connection can impact other users.

Connection Name	Database Name	User ID	Host Name	Port Number
gsdb	gsdb	db2admin	127.0.0.1	50001

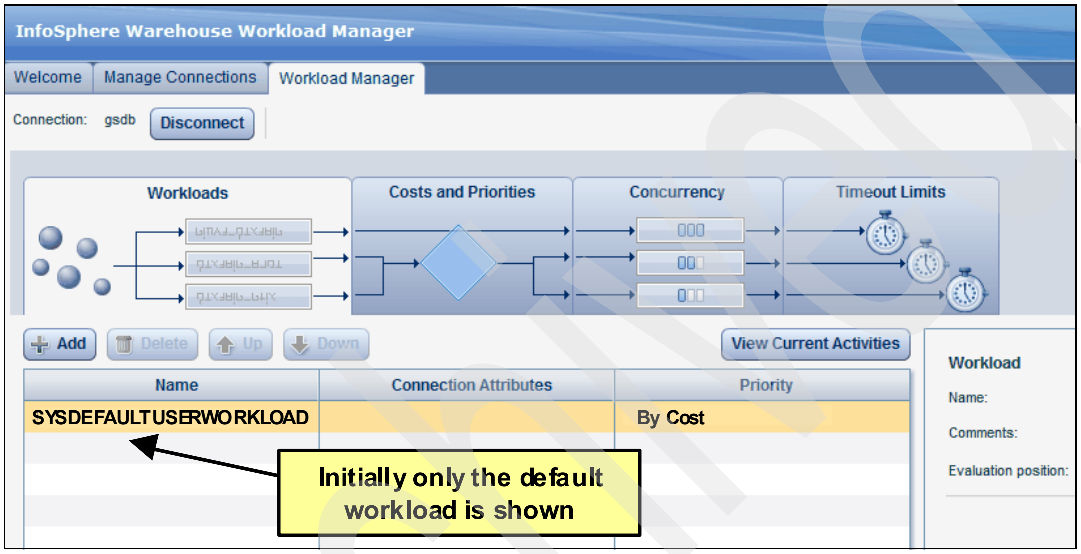
Figure 10-7 WLM connection summary with new connection

10.3.2 Configuring the workload

Perform the following steps to configure the workload.

1. Return to the Workload Manager tab and press the **Connect** button next to the database name.

   The existing workload summary for that database is displayed. Initially only the default workload (SYSDEFAULTUSERWORKLOAD) is shown, as depicted in Figure 10-8.



Name	Connection Attributes	Priority
SYSDEFAULTUSERWORKLOAD		By Cost

Figure 10-8 Initial workload management window

2. Identify and classify workloads based on connection information, such as group of users submitting queries and the applications they are using. The objective is to differentiate between types of workload that should be assigned different priorities.

   As an example there might be:

   – A group of executives that submitting high priority and short running queries.

   – Actuaries requesting large data extracts that typically run a long time.

   – Cognos users submitting a mix of short and long running queries.

The executives are waiting at their computers for the results, and their workload should be given a HIGH priority. The actuaries have likely submitted their extracts as batch jobs, as they are less urgent and can be assigned a LOW priority. The Cognos users will have their queries designated as HIGH, MEDIUM or LOW priority depending on the expected query cost.

Sometimes there might be different groups of users submitting queries from the same application under the same authorization identifier. Where possible the application (such as Cognos) should be configured to differentiate between these groups and supply a client identifier along with the SQL query.

In our example we use the Client User ID to differentiate the executives from other Cognos query users.

Workloads can be defined in terms of user and connection identifiers including a combination of:

- Application name
- User ID
- Session user ID
- Group ID
- Role ID
- Client user ID
- Client application name
- Client workstation
- Client accounting string
- IP address

3. Once the workloads have been identified, enter them one at a time, as shown in Figure 10-9.

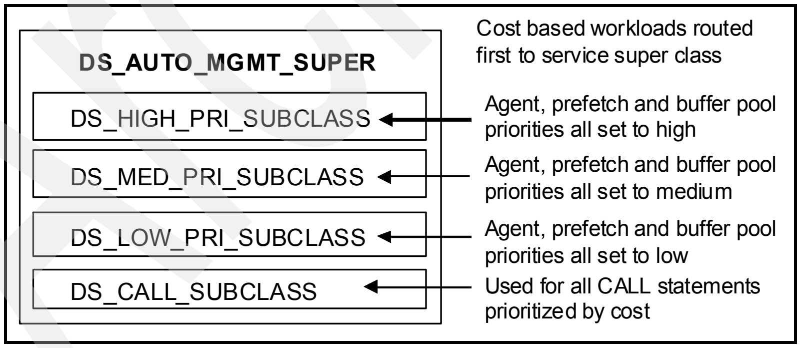

DS_AUTO_MGMT_SUPER
DS_HIGH_PRI_SUBCLASS
DS_MED_PRI_SUBCLASS
DS_LOW_PRI_SUBCLASS
DS_CALL_SUBCLASS
Cost based workloads routed first to service super class
Agent, prefetch and buffer pool priorities all set to high
Agent, prefetch and buffer pool priorities all set to medium
Agent, prefetch and buffer pool priorities all set to low
Used for all CALL statements prioritized by cost

Figure 10-9 Workload entry

4. Click the **ADD** button to enter details of each of the workloads to be managed.

5. Enter the name of the first workload in the pop up window and press **OK**.

   The evaluation position defines the sequence which the workloads are evaluated (the SYSDEFAULTUSERWORKLOAD is always considered last). We want the Cognos workload for the executives to be evaluated before the workload for other Cognos users.

   Processing priority is selected for the workload (High, Medium, Low or Cost Based).

   A summary of the workloads are immediately shown in the panel on the left of the entry window. See Figure 10-10.

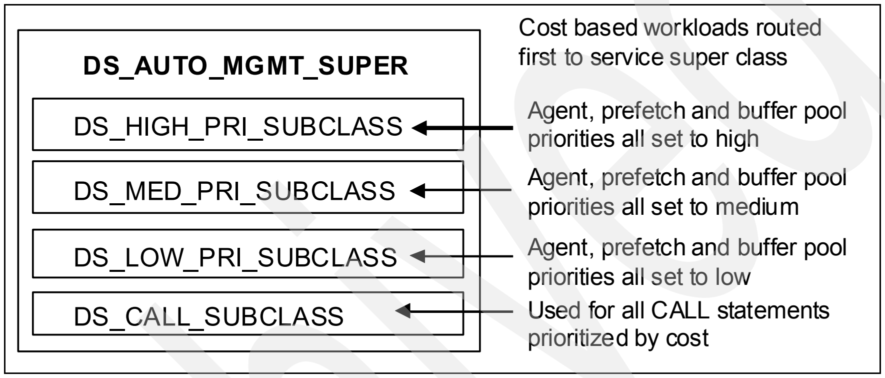
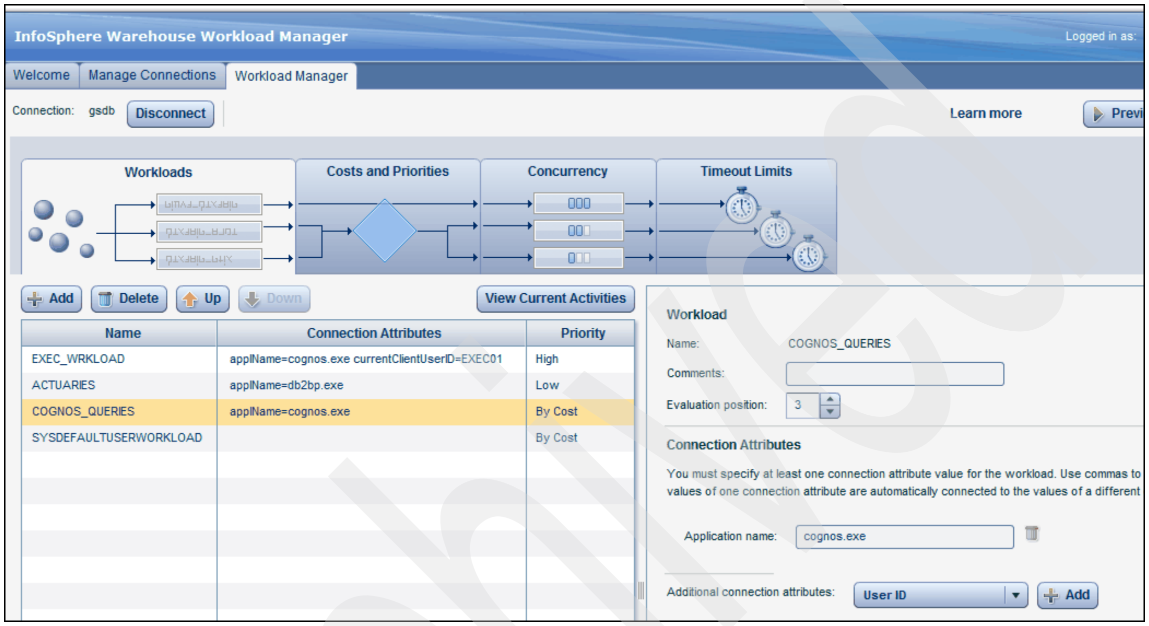

DS_AUTO_MGMT_SUPER
Cost based workloads routed first to service super class
DS_HIGH_PRI_SUBCLASS
Agent, prefetch and buffer pool priorities all set to high
DS_MED_PRI_SUBCLASS
Agent, prefetch and buffer pool priorities all set to medium
DS_LOW_PRI_SUBCLASS
Agent, prefetch and buffer pool priorities all set to low
DS_CALL_SUBCLASS
Used for all CALL statements prioritized by cost

Figure 10-10 Workload detail entry

Once the workloads for executives, actuaries and Cognos query users have been entered they are shown on the workload summary window as depicted in Figure 10-11.

There is always a default workload configured for work not explicitly identified by one of the other workload definitions.



Connection: gsdb Disconnect

Name	Connection Attributes	Priority
EXEC_WRKLOAD	applName=cognos.exe currentClientUserID=EXEC01	High
ACTUARIES	applName=db2bp.exe	Low
COGNOS_QUERIES	applName=cognos.exe	By Cost
SYSDEFAULTUSERWORKLOAD		By Cost

Workload

Name: COGNOS_QUERIES

Comments:

Evaluation position: 3

Connection Attributes

You must specify at least one connection attribute value for the workload. Use commas to values of one connection attribute are automatically connected to the values of a different

Application name: cognos.exe

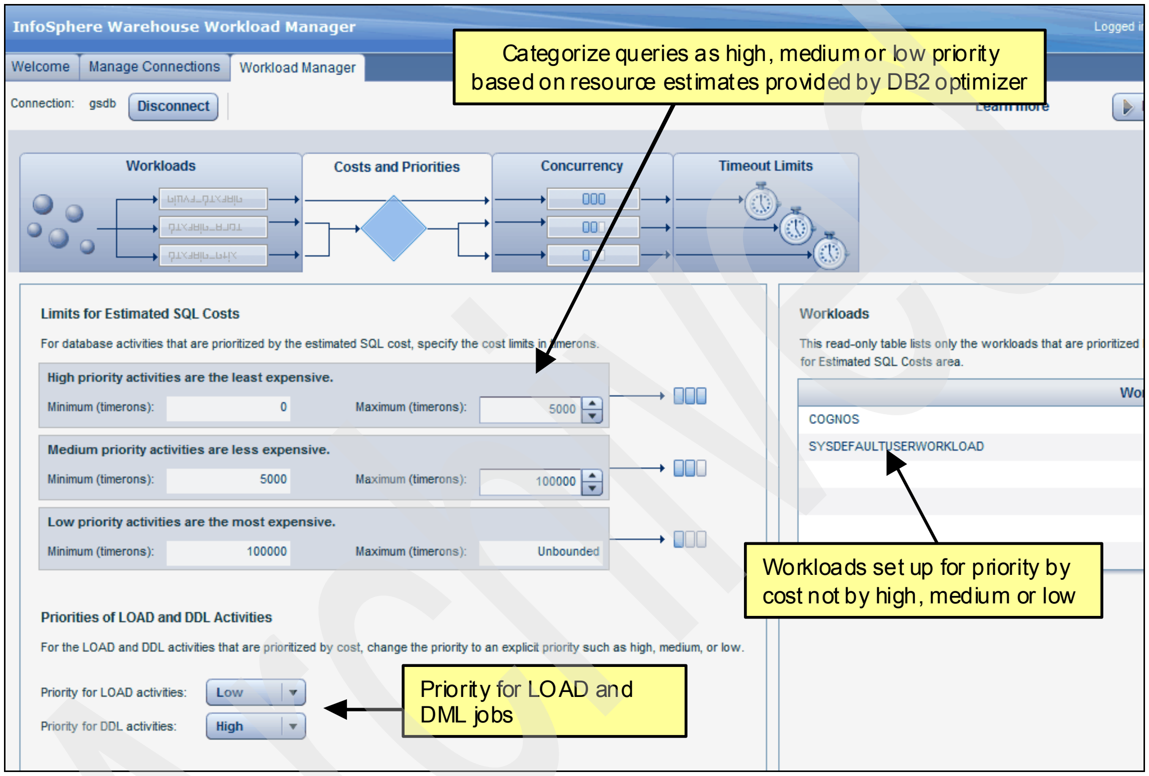
Additional connection attributes: User ID

Figure 10-11 Workload summary

10.3.3 Categorizing WLM costs and priorities

The next step is to categorize the workloads defined as having priority by cost. Perform the following steps to categorize WLM costs and priorities.

1. Click the Costs and Priorities icon and adjust the cost ranges as necessary for each of these workloads, as shown in Figure 10-12.


InfoSphere Warehouse Workload Manager
Welcome
Manage Connections
Workload Manager
Connection: gsdb
Disconnect
Learn more
Workloads
Costs and Priorities
Concurrency
Timeout Limits
Categorize queries as high, medium or low priority based on resource estimates provided by DB2 optimizer
Workloads set up for priority by cost not by high, medium or low
Priority for LOAD and DML jobs

Limits for Estimated SQL Costs

For database activities that are prioritized by the estimated SQL cost, specify the cost limits in timerons.

High priority activities are the least expensive.
Minimum (timerons): 0 Maximum (timerons): 5000

Medium priority activities are less expensive.
Minimum (timerons): 5000 Maximum (timerons): 100000

Low priority activities are the most expensive.
Minimum (timerons): 100000 Maximum (timerons): Unbounded

Priorities of LOAD and DDL Activities

For the LOAD and DDL activities that are prioritized by cost, change the priority to an explicit priority such as high, medium, or low.

Priority for LOAD activities: Low
Priority for DDL activities: High

Workloads

This read-only table lists only the workloads that are prioritized for Estimated SQL Costs area.

Wo
COGNOS
SYSDEFAULTUSERWORKLOAD

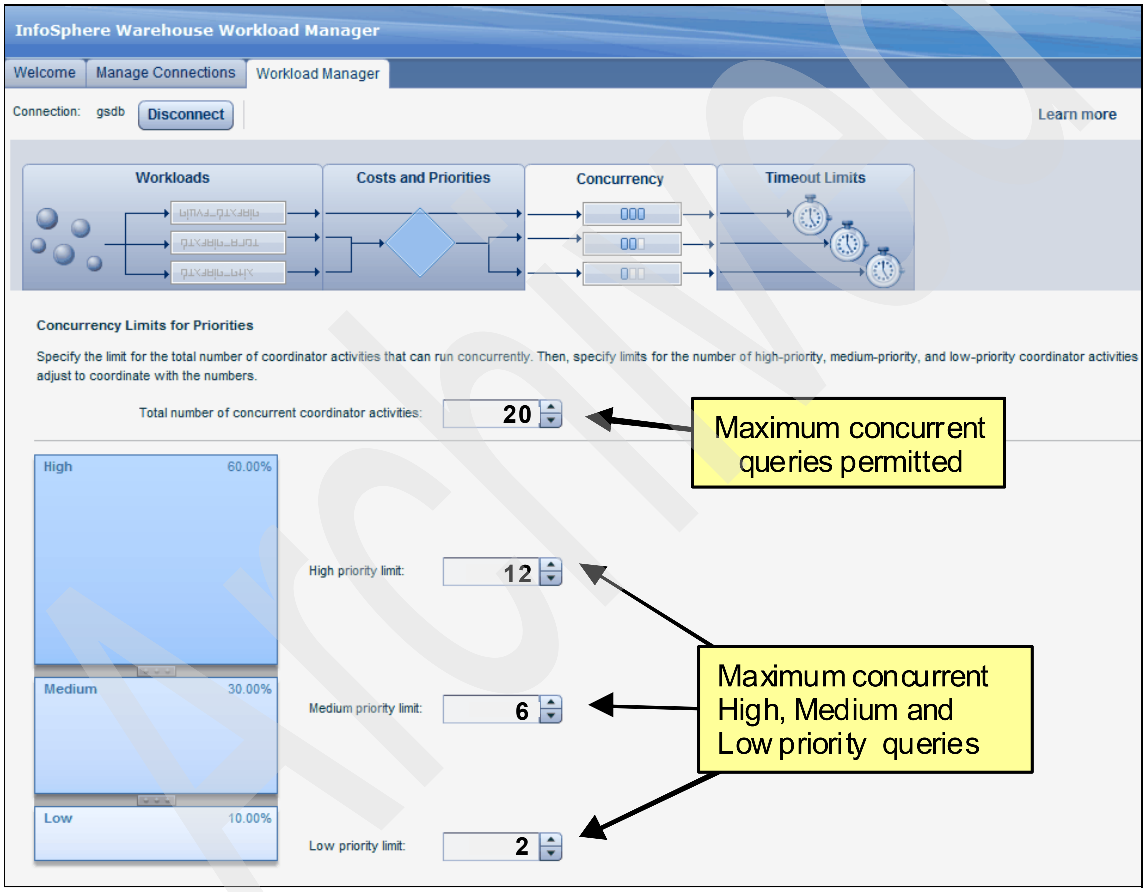
Figure 10-12 Workloads prioritized by cost estimates

2. Set resource estimates (as defined by DB2 optimizer timerons) for High, Medium, and Low ranges. These ranges apply identically to all cost-based workloads.
3. Assign cost-based LOAD and DML jobs as having High, Medium or Low priorities. If workloads defined in 10.3.2, “Configuring the workload” on page 439 are explicitly given HIGH, MEDIUM or LOW priorities, these cost ranges have no affect.

10.3.4 WLM concurrency control

The next step is to decide how many HIGH, MEDIUM and LOW priority queries are allowed to run concurrently, as shown in Figure 10-13. When the number of queries submitted for one or more of these categories is exceeded, subsequent queries are queued until running queries complete execution.

This process is invisible to the person who submits the query. No return code is passed back, the query just waits until there is room in the queue for it to continue.



InfoSphere Warehouse Workload Manager
Welcome
Manage Connections
Workload Manager
Disconnect
Learn more
Workloads
Costs and Priorities
Concurrency
Timeout Limits
Maximum concurrent queries permitted
Maximum concurrent High, Medium and Low priority queries

Connection: gsdb

Concurrency Limits for Priorities

Specify the limit for the total number of coordinator activities that can run concurrently. Then, specify limits for the number of high-priority, medium-priority, and low-priority coordinator activities adjust to coordinate with the numbers.

Total number of concurrent coordinator activities: 20

High 60.00%

Medium 30.00%

Low 10.00%

High priority limit: 12

Medium priority limit: 6

Low priority limit: 2

Figure 10-13 Defining maximum query concurrency

10.3.5 WLM timeouts

Decide whether long running queries should be aborted rather than being allowed to run to conclusion. The time limits (in minutes) can be entered separately for HIGH, MEDIUM and LOW priority queries (see Figure 10-14).

Make sure that any occasional long running but legitimate queries are not caught by this timeout.

Note: Timeouts must be applied to all categories of queries or none. It is not possible, for example, to use this tool to selectively apply a timeout to only low running queries. It is possible, however, to set a sufficiently high value that the timeout abort is never triggered.

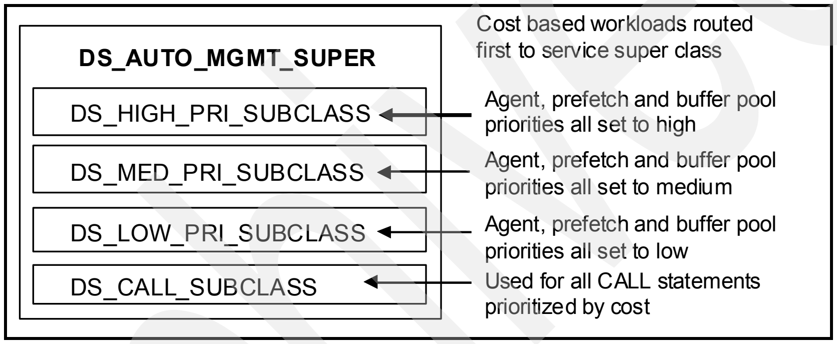
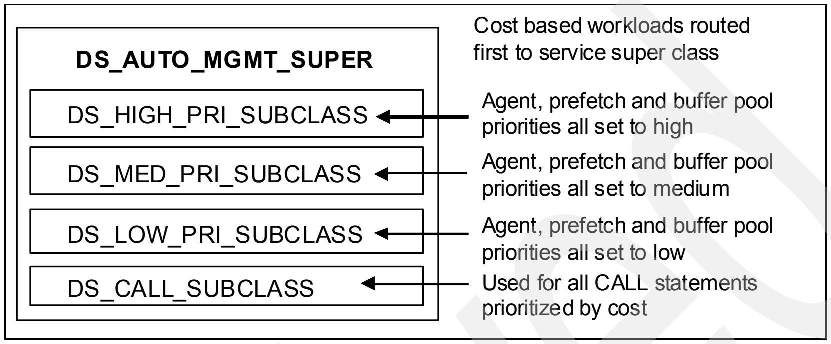

DS_AUTO_MGMT_SUPER
DS_HIGH_PRI_SUBCLASS
DS_MED_PRI_SUBCLASS
DS_LOW_PRI_SUBCLASS
DS_CALL_SUBCLASS
Cost based workloads routed first to service super class
Agent, prefetch and buffer pool priorities all set to high
Agent, prefetch and buffer pool priorities all set to medium
Agent, prefetch and buffer pool priorities all set to low
Used for all CALL statements prioritized by cost

Figure 10-14 Set timeout limits for long running queries

10.3.6 Reviewing WLM and running SQL

Save your configuration draft with the **Save Configuration** button. Review and implement the underlying DB2 WLM commands generated by the administration console using the **Review and Run SQL** button.



DS_AUTO_MGMT_SUPER
DS_HIGH_PRI_SUBCLASS
DS_MED_PRI_SUBCLASS
DS_LOW_PRI_SUBCLASS
DS_CALL_SUBCLASS
Cost based workloads routed first to service super class
Agent, prefetch and buffer pool priorities all set to high
Agent, prefetch and buffer pool priorities all set to medium
Agent, prefetch and buffer pool priorities all set to low
Used for all CALL statements prioritized by cost

Figure 10-15 Workload configuration SQL

The SQL to configure the underlying DB2 WLM objects can be run directly from the administration console using the **Run SQL** option. It can also be copied to any other query tool using copy and paste from the review window.

10.3.7 WLM underlying DB2 objects.

If you examine the DB2 WLM SQL you see that there are five sections:

- Service classes
- Workloads
- Work Class Sets
- Thresholds
- Work Action Sets

Service classes

All workloads configured by the InfoSphere administration console run in a single service super class with four underlying subclasses, as shown in Figure 10-16.

Workloads either map directly to the service subclasses (when the priority is explicitly assigned) or indirectly when cost based priority is chosen.

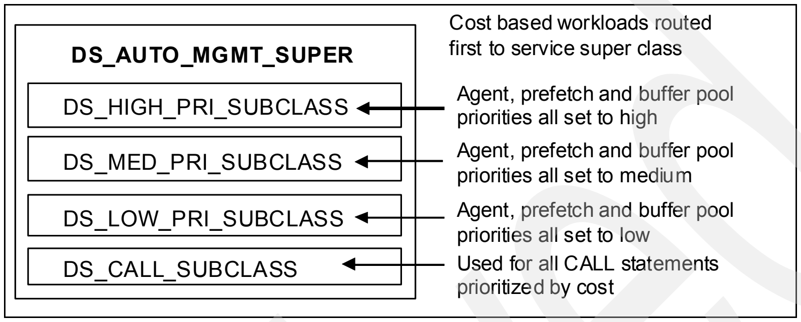

DS_AUTO_MGMT_SUPER
DS_HIGH_PRI_SUBCLASS
DS_MED_PRI_SUBCLASS
DS_LOW_PRI_SUBCLASS
DS_CALL_SUBCLASS
Cost based workloads routed first to service super class
Agent, prefetch and buffer pool priorities all set to high
Agent, prefetch and buffer pool priorities all set to medium
Agent, prefetch and buffer pool priorities all set to low
Used for all CALL statements prioritized by cost

Figure 10-16 WLM service classes

Example 10-2 shows the DDL for creating a service superclass and a service subclass under that superclass for high priority work.

Example 10-2 DDL for creating WLM Service Class

```
CREATE SERVICE CLASS "DS_AUTO_MGMT_SUPER" DISABLE;

CREATE SERVICE CLASS "DS_HIGH_PRI_SUBCLASS"
   UNDER "DS_AUTO_MGMT_SUPER"
   PREFETCH PRIORITY HIGH
   BUFFERPOOL PRIORITY HIGH
   COLLECT AGGREGATE ACTIVITY DATA BASE DISABLE;
```

Workloads

Each of the workloads entered in the application will have a corresponding CREATE WORKLOAD section. Those with explicit priorities are mapped to the corresponding service subclasses and those configured with cost priorities are mapped to the overall service superclass for subsequent mapping by a combination of the Work Class Set and Work Action Set.

An sample workload definition is shown in Example 10-3 on page 448. This workload applies to users of the `cognos.exe` application running under user ID `EXEC01`. The work is explicitly allocated to the DS_HIGH_PRI_SUBCLASS service sub-class, which has been defined for high priority queries. The workload is evaluated at position 1, which means, for example, that it would take precedence over another workload defined to application `cognos.exe` without an explicit user ID. Aggregate statistical activity is collected for this workload.

Example 10-3 DDL for creating a WML Workload

```
CREATE WORKLOAD "EXEC_WRKLOAD" APPLNAME ('cognos.exe')
CURRENT CLIENT_USERID ('EXEC01') DISABLE
SERVICE CLASS "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" POSITION AT 1
COLLECT AGGREGATE ACTIVITY DATA BASE
ACTIVITY LIFETIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
ACTIVITY QUEUETIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
```

Work Class Sets

One Work Class Set is defined with a work class for each of the cost priority ranges (using optimizer timer on values to define the ranges) and work classes for CALL statements, DDL, LOADs and any other workload type.

Example 10-4 shows the ranges used to identify the three work classes. Separate work classes are create for CALL, DDL, and LOAD statement with a catch-all work class for anything else not otherwise defined.

Example 10-4 DDL for creating WML Work Class Set

```
CREATE WORK CLASS SET "DS_WORK_CLASS_SET"
( WORK CLASS "DS_LOW_COST_DML_WC"
   WORK TYPE DML FOR TIMERONCOST FROM 0.0 TO 5000.0
   POSITION AT 1,
WORK CLASS "DS_MED_COST_DML_WC"
   WORK TYPE DML FOR TIMERONCOST FROM 5000.0 TO 100000.0
   POSITION AT 2,
WORK CLASS "DS_HIGH_COST_DML_WC"
   WORK TYPE DML FOR TIMERONCOST FROM 100000.0 TO UNBOUNDED
   POSITION AT 3,
WORK CLASS "DS_CALL_WC"
   WORK TYPE CALL
   POSITION AT 4,
WORK CLASS "DS_DDL_WC"
   WORK TYPE DDL POSITION AT 5,
WORK CLASS "DS_LOAD_WC"
   WORK TYPE LOAD
   POSITION AT 6,
WORK CLASS "DS_OTHER_WC"
   WORK TYPE ALL POSITION AT 7);
```

Thresholds

Two sets of thresholds are defined at the service superclass level:

- One set for controlling concurrency with separate thresholds for high, medium, and low priority, as depicted in Example 10-5.

Example 10-5 DDL for creating WLM Threshold for concurrency

```
CREATE THRESHOLD "DS_HIGH_PRI_CONC_DB_TH"
   FOR SERVICE CLASS "DS_HIGH_PRI_SUBCLASS"
   UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
   WHEN CONCURRENTDBCOORDACTIVITIES > 12
   AND QUEUEDACTIVITIES UNBOUNDED  CONTINUE;
```

- One set for controlling timeouts with separate activity time limits for high, medium, and low priority workloads, as depicted in Example 10-6.

Example 10-6 DDL for creating WLM Threshold for timeouts

```
CREATE THRESHOLD "DS_HIGH_PRI_TIMEOUT_TH"
   FOR SERVICE CLASS "DS_HIGH_PRI_SUBCLASS"
   UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
   WHEN ACTIVITYTOTALTIME > 5 MINUTES
   COLLECT ACTIVITY DATA STOP EXECUTION;
```

Work Action Sets

The Work Action Set controls the mapping of cost based priorities from the service superclass to the appropriate service subclass depending on the definitions stored in the work classes.

Work actions and work classes act in tandem with the work class defining the condition on which the corresponding action is taken.

The actions in Example 10-7 are used to map queries prioritized by cost to the appropriate service sub-class.

Example 10-7 DDL for creating WLM Work Action Set

```
CREATE WORK ACTION SET "DS_WORK_ACTION_SET"
   FOR SERVICE CLASS "DS_AUTO_MGMT_SUPER"
   USING WORK CLASS SET "DS_WORK_CLASS_SET"
   ( WORK ACTION "DS_MAP_LOW_COST_DML_WA"
      ON WORK CLASS "DS_LOW_COST_DML_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_HIGH_PRI_SUBCLASS",
   WORK ACTION "DS_MAP_MED_COST_DML_WA"
      ON WORK CLASS "DS_MED_COST_DML_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_MED_PRI_SUBCLASS",
   WORK ACTION "DS_MAP_HIGH_COST_DML_WA"
      ON WORK CLASS "DS_HIGH_COST_DML_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_PRI_SUBCLASS",
   WORK ACTION "DS_MAP_CALL_WA"
      ON WORK CLASS "DS_CALL_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_CALL_SUBCLASS",
   WORK ACTION "DS_MAP_DDL_WA"
      ON WORK CLASS "DS_DDL_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_HIGH_PRI_SUBCLASS",
   WORK ACTION "DS_MAP_LOAD_WA"
      ON WORK CLASS "DS_LOAD_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_PRI_SUBCLASS",
   WORK ACTION "DS_MAP_OTHER_WA"
      ON WORK CLASS "DS_OTHER_WC"
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_PRI_SUBCLASS" )  DISABLE;
```

10.4 WLM summary

InfoSphere Warehouse provides a web-based GUI application that assists users in configuring a subset of DB2 WLM functionality chosen as being most appropriate for data warehousing.

The underlying DB2 WLM commands are available as templates for those with specific requirements or a particularly large and complex workload.

11

Providing the analytics

In this chapter we describe the use of IBM Cognos, Alphablox, and Microsoft Excel as client tools that can access InfoSphere Warehouse Cubing Services cubes.

11.1 Cognos and the cube model

IBM Cognos 8 Business Intelligence (BI) can use your existing Cubing Services investment by providing the ability to access the existing dimensions, levels, and hierarchies as a Cognos 8 data source.

11.1.1 Cognos architecture

Cognos 8 uses a service-oriented architecture (SOA) that allows different components to be applied in a single application framework. For the purpose of this document we focus on the components involved in reporting using the BI components of the Cognos 8 platform. The base structure, as shown in Figure 11-1, consists of a tiered architecture. The individual services of the Cognos 8 server run in an application server and can be distributed across multiple application server instances.

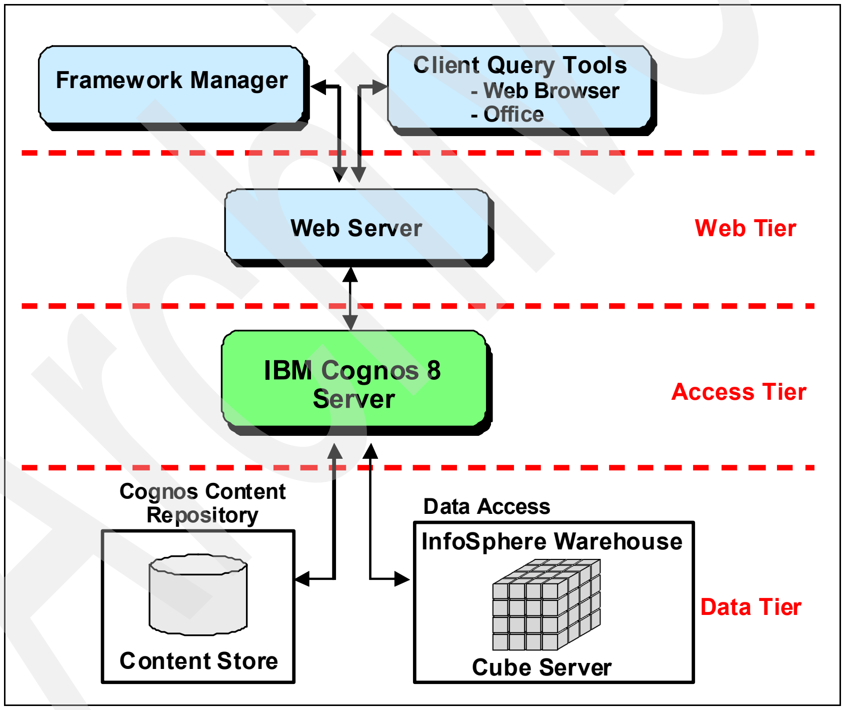


Figure 11-1 IBM Cognos 8 BI architecture

A zero-footprint browser interface provides users with the ability to create reports and access published content from the Content Store database repository. This is presented to users in the default IBM Cognos Connection portal interface. This portal also allows for administration and configuration of the Cognos 8 server properties.

IBM Cognos Framework Manager is the modeling and development tool used to generate the metadata model for user reporting. This is a full client application that is installed on the computer of the metadata modeler. Communication is still directed through the common tiered architecture when publishing packages to, or retrieving information from, the IBM Cognos 8 server.

11.1.2 Cognos metadata

Cognos 8 supports a direct connection to Cubing Services. This means that the metadata for the published package can be obtained directly from the cube server at run-time instead of requiring full metadata import into Framework Manager. You still need to publish a package from Framework Manager to enable access to the cube but there are no required changes to the cube properties in this scenario.

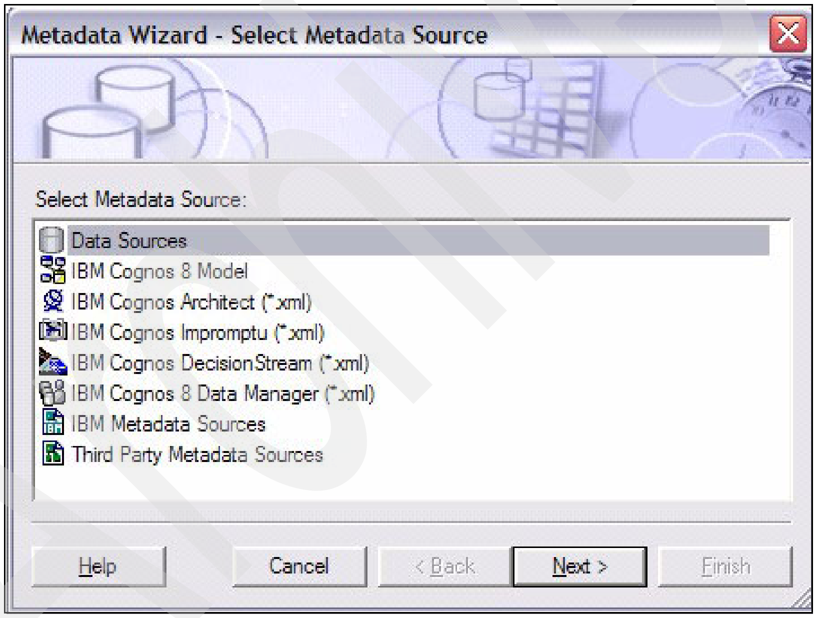
To connect to Cubing Services you need to define a data source connection to the Cube Server in Cognos 8 and import the cube into Framework Manager. In this instance the cube is a stub object that is used to reference the cube from the Cubing Services Cube Server. The full set of metadata for the dimensions, hierarchies, and levels remains in Cubing Services.

11.1.3 Importing into Framework Manager

To import into Framework Manager, perform the following steps:

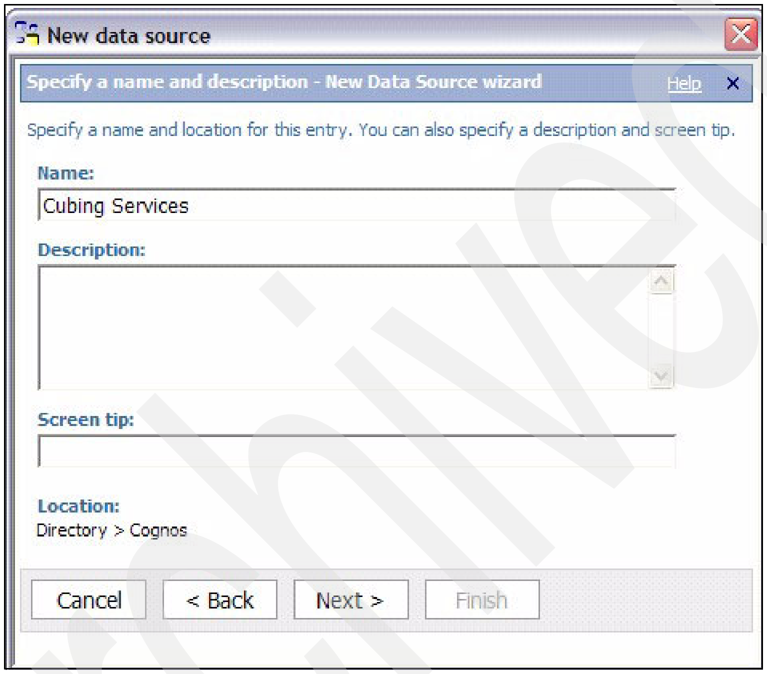
Note: When creating a new model in Framework Manager you are shown a Metadata Import Wizard. We start this importation from the first window of the wizard.

1. If you are using an existing model, start the Import Wizard by selecting any namespace or folder object in your model, and use the **Actions** → **Run Metadata Wizard**.
2. Because Cognos 8 can use Cubing Services as a data source, choose **Data Sources** as the source for our metadata import. Choose this option in the Select Metadata Source window of the Import Wizard (Figure 11-2). Click **Next**.


Metadata Wizard - Select Metadata Source
Select Metadata Source:
Data Sources
IBM Cognos 8 Model
IBM Cognos Architect (*.xml)
IBM Cognos Impromptu (*.xml)
IBM Cognos DecisionStream (*.xml)
IBM Cognos 8 Data Manager (*.xml)
IBM Metadata Sources
Third Party Metadata Sources
Help
Cancel
< Back
Next >
Finish
Figure 11-2 Choosing the metadata source

3. If you have not already created a data source connection to your Cubing Services data source in Cognos 8, you have the option to do so from the Metadata Import Wizard. To create a data source connection, perform the following steps:

   a. Click **New** to launch the New Data Source Wizard and click **Next** on the first page of the wizard to pass the welcome message. This presents you with the window shown in Figure 11-3.



Specify a name and location for this entry. You can also specify a description and screen tip.

Name:
Cubing Services

Description:

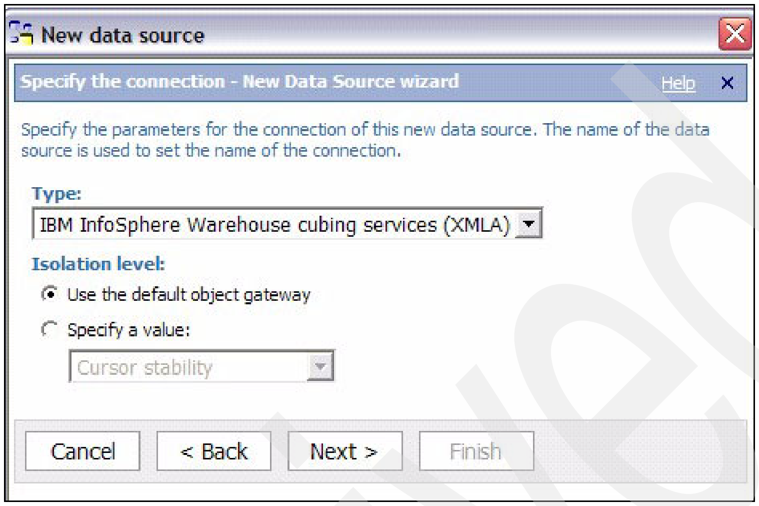
Screen tip:

Location:
Directory > Cognos

Figure 11-3   Naming the Cubing Services data source

   b. Enter a name in the Name text box. The name can be any logical name that you would like to use to identify your Cubing Services data source connection. Click **Next**.

c. Select IBM InfoSphere Warehouse Cubing Services (XMLA) from the Type menu, as shown in the New Data Source window (Figure 11-4). Click **Next**.


New data source

Specify the connection - New Data Source wizard

Specify the parameters for the connection of this new data source. The name of the data source is used to set the name of the connection.

Type: IBM InfoSphere Warehouse cubing services (XMLA)

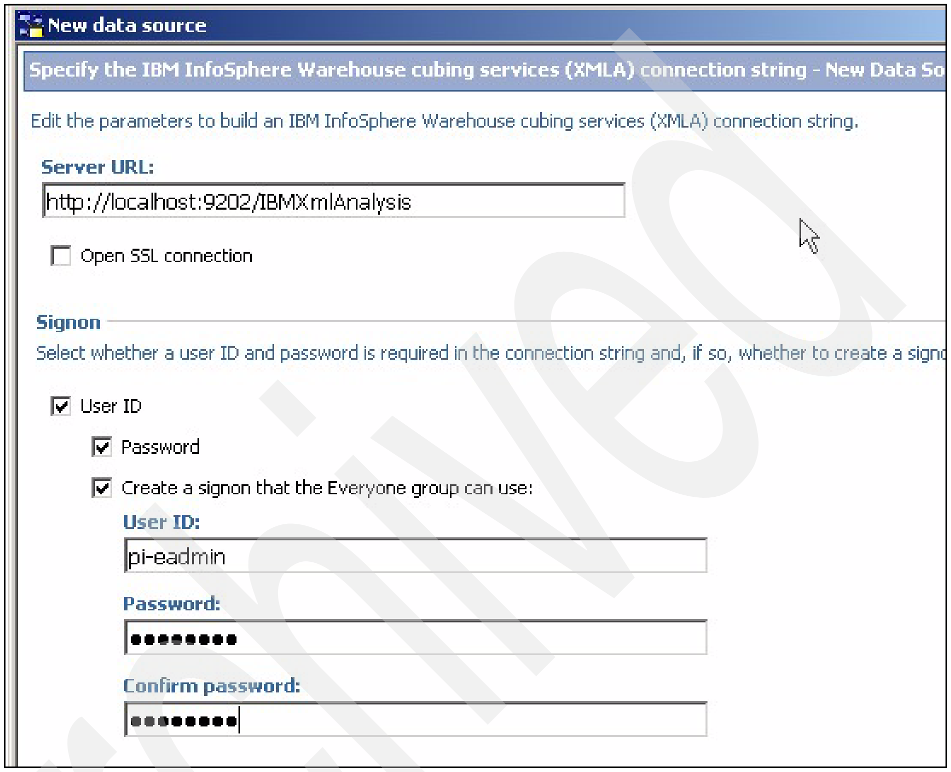
Isolation level:

(●) Use the default object gateway

( ) Specify a value: Cursor stability

Figure 11-4 Choosing the data source type

d. The Cubing Services connection is established using the XMLA protocol, so you need only supply a server URL (in the form server:port/InstanceName) and a valid Cubing Services user ID and password for the connection, as shown in Figure 11-5.


New data source

Specify the IBM InfoSphere Warehouse cubing services (XMLA) connection string - New Data So

Edit the parameters to build an IBM InfoSphere Warehouse cubing services (XMLA) connection string.

Server URL: http://localhost:9202/IBMXmlAnalysis

☐ Open SSL connection

Signon

Select whether a user ID and password is required in the connection string and, if so, whether to create a signo

☑ User ID

☑ Password

☑ Create a signon that the Everyone group can use:

User ID: pi-eadmin

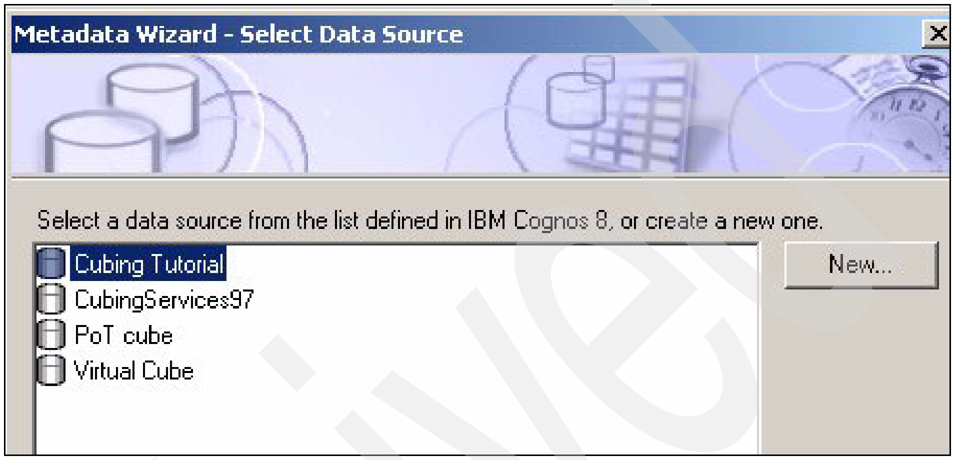
Password: ●●●●●●●●●

Confirm password: ●●●●●●●●●

Figure 11-5 Data source server and sign on window

e. Click the **Test the Connection** link in the final window of the New Data Source Wizard to test the connection from the Cognos 8 server. After the test completes successfully, click **Next** to finish creating the data source.

Figure 11-6 shows the newly created data source available as a selectable data source, in the metadata wizard window.

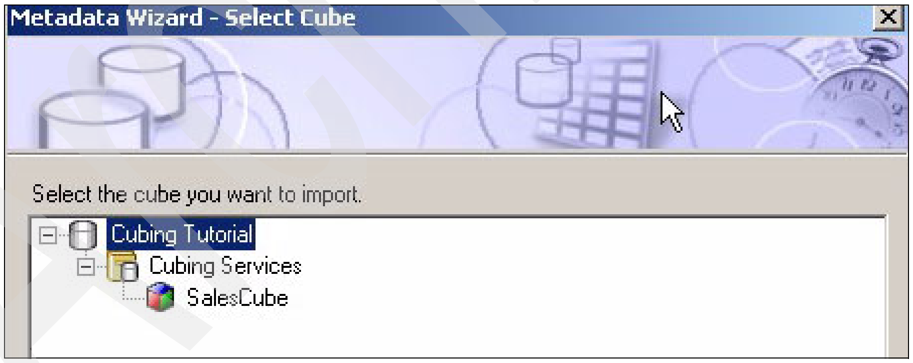


Select a data source from the list defined in IBM Cognos 8, or create a new one.

Cubing Tutorial
CubingServices97
PoT cube
Virtual Cube

Figure 11-6 Available Data Source Connections

4. Click **Next**. The metadata wizard displays a list of cubes on the Cube Server. This is shown in Figure 11-7.



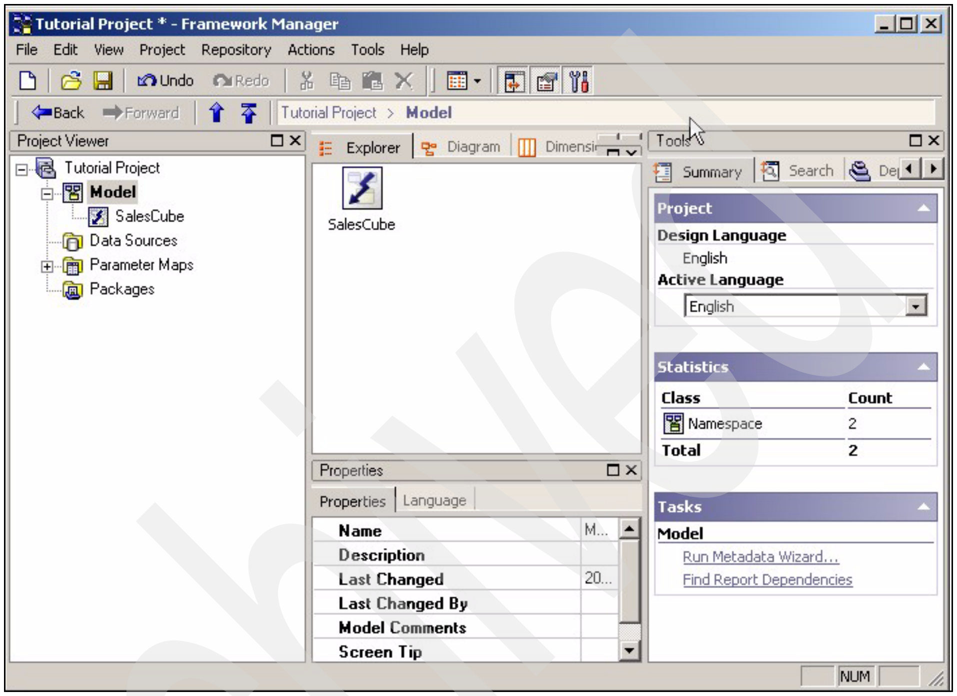
Select the cube you want to import.

Cubing Tutorial
Cubing Services
SalesCube

Figure 11-7 List of available cubes from the Cube Server

After you select the cube object desired, you are given the option to import it. When importing a cube object you are prompted at the end of the Metadata Import Wizard to create a package. Typically, this is the default action that you would want to use for cube sources. Using this option creates a package containing the single cube reference. After you have completed the Package

Wizard you are prompted to publish the package. Because there are no modeling tasks required for cube sources you can publish the new package to the IBM Cognos 8 server. The resulting model created by this import appears as shown in Figure 11-8.



Properties	
Name	M...
Description	
Last Changed	20...
Last Changed By	
Model Comments	
Screen Tip	

Project

Design Language: English

Active Language: English

Statistics

Class	Count
Namespace	2
Total	2

Tasks

Model

Figure 11-8 Framework Manager view after a cube import

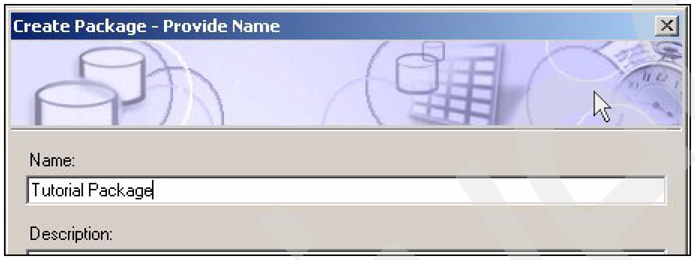
If your business needs call for multiple cubes, repeat the Metadata Import Wizard to include the additional cubes from the Cube Server in your Framework Manager model. After you have all the required cubes, you can create a package containing the cubes that you require. Having many cubes means that the metadata for each cube must be loaded when a report author is creating a report. If the business users only use a subset of the cubes that are available, it would be best to tailor your package contents to the business needs rather than include all available cubes.

11.1.4 Publishing a Framework Manager package

Once you have created your consolidated package you can publish the contents to the Cognos 8 server using either the right-click menu or the Actions menu after selecting the package in Framework Manager.

To create and publish a package, perform the following steps:

1. Right-click **Packages** in the project viewer and select **Create** → **Package**.
2. Name the package `Tutorial Package` and click **Next**, See Figure 11-9.


Create Package - Provide Name

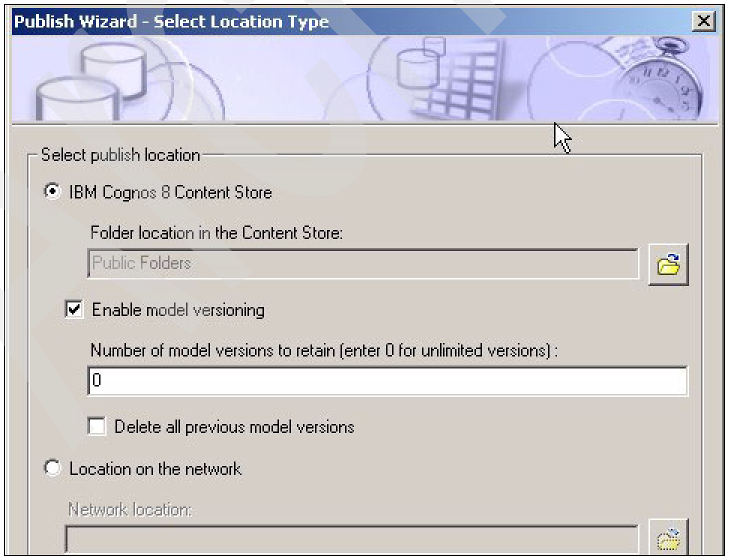
Name:

Tutorial Package

Description:

Figure 11-9 Package name

3. Define the objects that you want to include in the package.
4. Click **Finish**, and click **Yes** to publish.
5. The “Select Location Type” window allows you to specify the publish location and whether or not you would like to use model versioning, as shown in Figure 11-10.


Publish Wizard - Select Location Type

Select publish location

(●) IBM Cognos 8 Content Store

Folder location in the Content Store:

Public Folders

[✓] Enable model versioning

Number of model versions to retain (enter 0 for unlimited versions) :

0

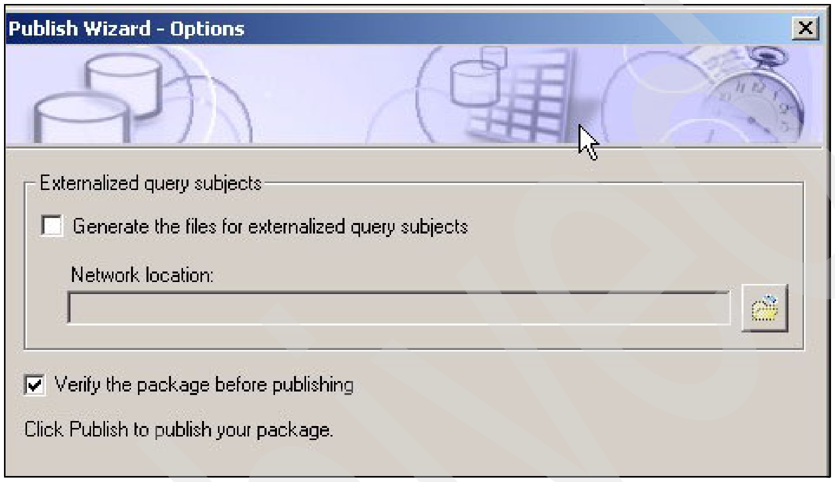
[ ] Delete all previous model versions

( ) Location on the network

Network location:

Figure 11-10 Publish location

6. Ensure that you are publishing to the Cognos Content Store and click **Next** to open the security dialog box. In this window you can set the default access permissions for the package about to be published to IBM Cognos Connection. Click **Next**.
7. Click **Publish** in the options window (Figure 11-11) to verify the package before publishing.


Publish Wizard - Options

Externalized query subjects

☐ Generate the files for externalized query subjects

Network location:

☑ Verify the package before publishing

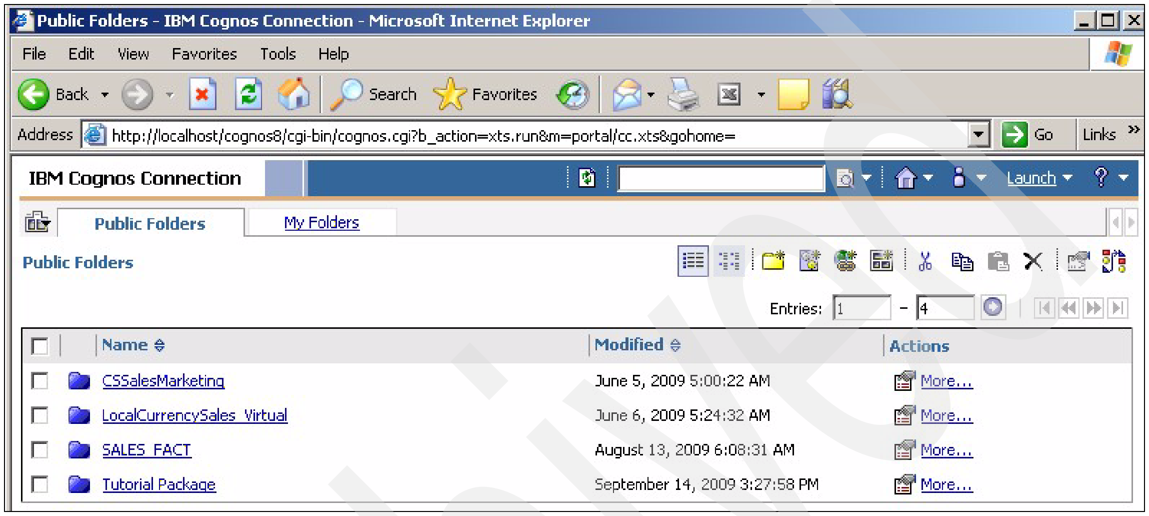
Click Publish to publish your package.

Figure 11-11 Publish wizard options

8. Click **Finish** to exit the wizard.

11.1.5 Empowering users

Once the cube model has been published to Cognos 8, the user experience starts with access to the common web portal, IBM Cognos Connection, as shown in Figure 11-12.


Public Folders - IBM Cognos Connection - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost/cognos8/cgi-bin/cognos.cgi?b_action=xts.run&m=portal/cc.xts&gohome=
Go
Links
IBM Cognos Connection
Launch
Public Folders
My Folders
Public Folders
Entries: 1 – 4

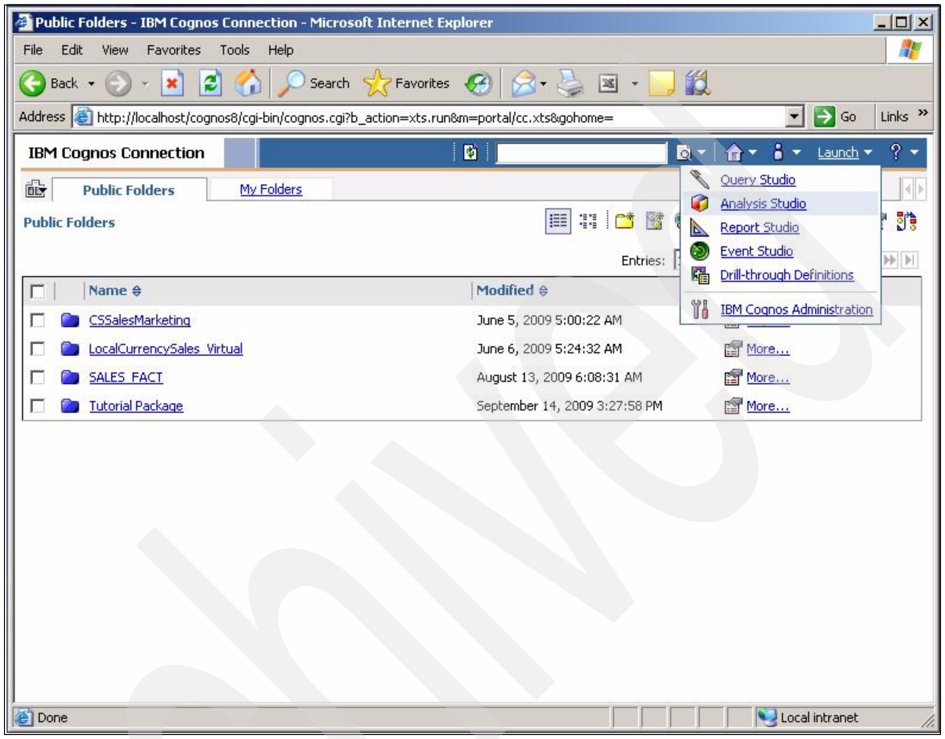
	Name	Modified	Actions
	CSSalesMarketing	June 5, 2009 5:00:22 AM	More...
	LocalCurrencySales_Virtual	June 6, 2009 5:24:32 AM	More...
	SALES_FACT	August 13, 2009 6:08:31 AM	More...
	Tutorial Package	September 14, 2009 3:27:58 PM	More...

Figure 11-12 Cognos Connection displaying published packages

Cognos Connection provides the entry-level access point to the published packages, saved reports, dashboards, and metrics.

Retrieving data using Analysis Studio

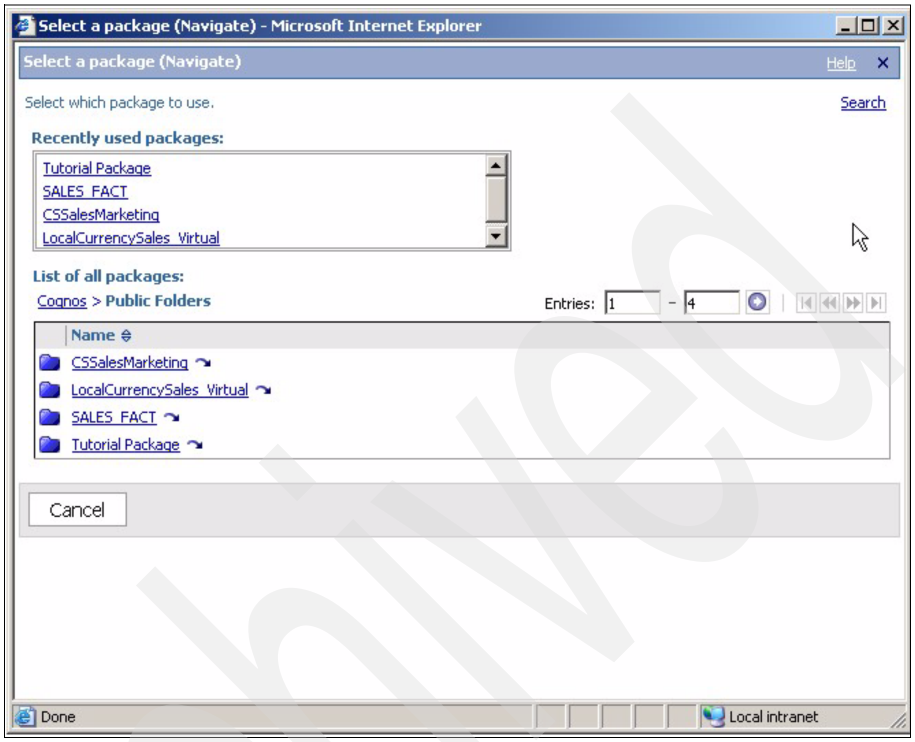
1. Click **Launch** in the Cognos Connection window and select **Analysis Studio**, as shown in Figure 11-13.


Public Folders - IBM Cognos Connection - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cognos8/cgi-bin/cognos.cgi?b_action=xts.run&m=portal/cc.xts&gohome=
Go Links
IBM Cognos Connection
Launch
Query Studio
Analysis Studio
Report Studio
Event Studio
Drill-through Definitions
IBM Cognos Administration
Public Folders
My Folders
Entries:
Done
Local intranet

Name	Modified	
CSSalesMarketing	June 5, 2009 5:00:22 AM	More...
LocalCurrencySales_Virtual	June 6, 2009 5:24:32 AM	More...
SALES_FACT	August 13, 2009 6:08:31 AM	More...
Tutorial Package	September 14, 2009 3:27:58 PM	More...

Figure 11-13   Launching Analysis Studio

2. Select the appropriate package for the analysis (in this case, Tutorial Package) as in Figure 11-14.


Select a package (Navigate) - Microsoft Internet Explorer

Select a package (Navigate)

Select which package to use.

Recently used packages:

Tutorial Package
SALES_FACT
CSSalesMarketing
LocalCurrencySales_Virtual

List of all packages:

Cognos > Public Folders

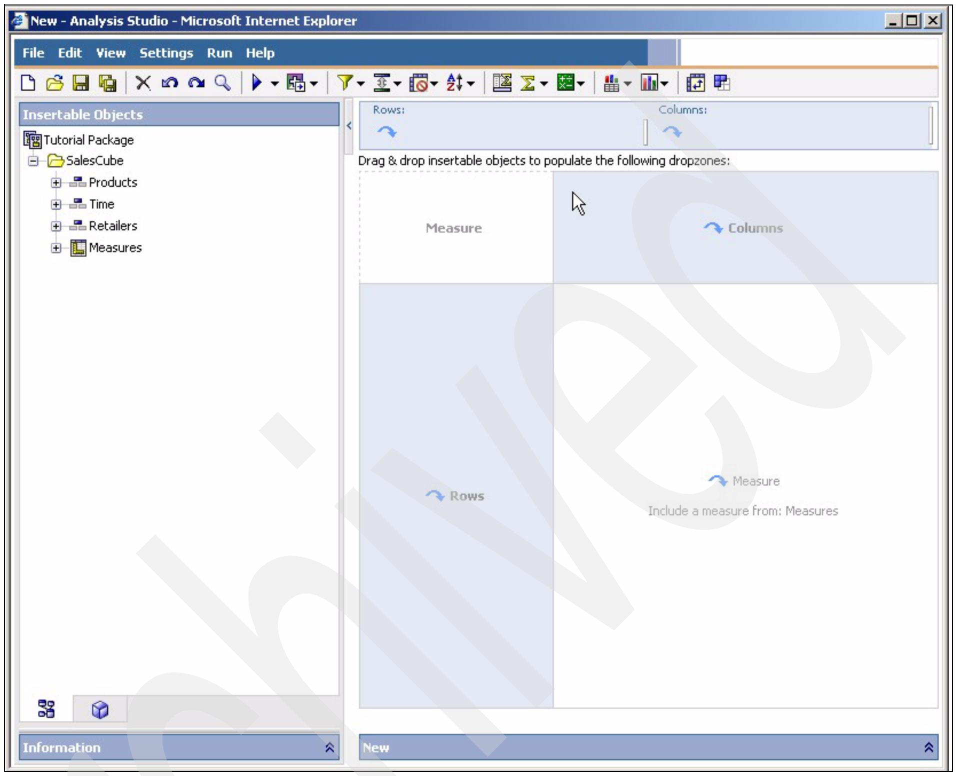
Entries: 1 - 4

Name
CSSalesMarketing
LocalCurrencySales_Virtual
SALES_FACT
Tutorial Package

Figure 11-14 Select a package

3. Select **Blank Analysis** and click **OK**.

4. Cube metadata is displayed on the left of the pane with Insertable Objects, as depicted in Figure 11-15. Objects can now be dragged into the page area for ad hoc reporting.



Drag & drop insertable objects to populate the following dropzones:

Include a measure from: Measures

Figure 11-15 Analysis Studio

11.2 Using the Alphablox client interface

Alphablox is an application development platform for rapidly building and deploying custom analytic solutions across an enterprise. Alphablox applications run as J2EE-compliant applications in an application server and are accessed using a web browser.

Alphablox works with Cubing Services to provide OLAP access to data directly from InfoSphere Warehouse. Alphablox can access and interact with Cubing Services and DB2.

In this section we show how Alphablox can be used to connect to a Cubing Services multidimensional data source and query it using the Multidimensional Expressions (MDX) query language.

11.2.1 Alphablox overview

Alphablox provides the ability to create rapidly custom, Web-based applications that fit into the corporate infrastructure and have the ability to reach a wide range of users.

Applications built with the Alphablox platform run in standard web browsers. To users, an Alphablox application appears as a collection of web pages to browse, just as with other web sites. Regardless of role or technical experience level, users find Alphablox intuitive and easy to use. Anyone comfortable using a web browser can navigate through the application and understand the available functionality.

Alphablox is built to access Cubing Services multidimensional data sources, which allows users to interact with different levels of data using analysis actions such as drilling, sorting, filtering, and pivoting to display interactively the exact view of data desired.

11.2.2 Alphablox architecture

The Alphablox Server runs in WebSphere Application Server as a set of JAR files and servlets. Alphablox takes advantage of all aspects of the application server, such as authentication, session management, clustering, and deployment management.

Alphablox provides an extensive library of modular, reusable components called Blox to help meet the analytic application design requirements for maximum usability. The Alphablox architecture is shown in Figure 11-16.

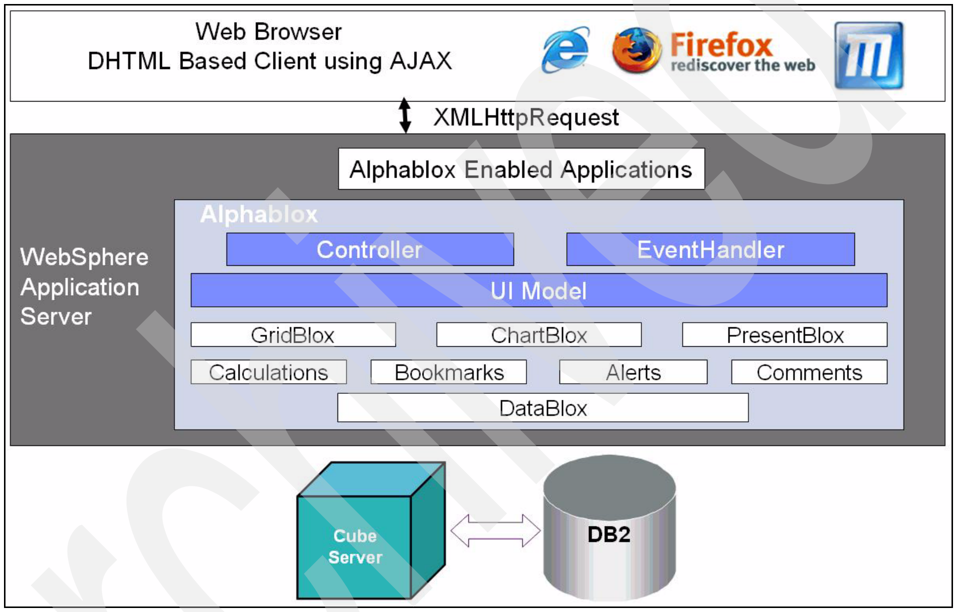


Figure 11-16 Alphablox architecture

Data access Blox, or Data Blox, handle the connections to databases, regardless of the database type. It is responsible for submitting queries and retrieving results sets from database. The syntax used for queries varies depending on the data source that is being accessed. In the case of Cubing Services, the Multidimensional Expressions (MDX) query language is used.

11.2.3 Defining a data source to Cubing Services

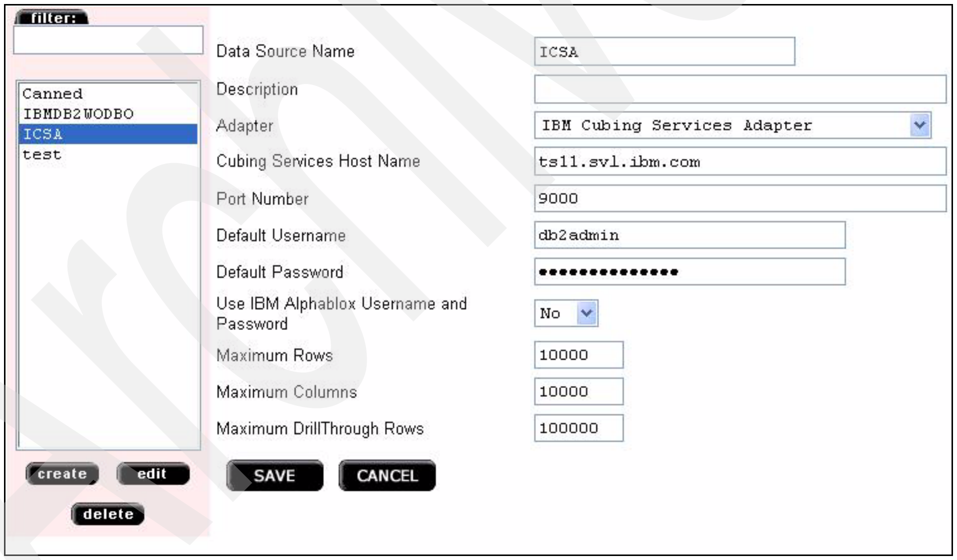
After successfully deploying and starting cubes in Cubing Services, you are now ready to define a data source in Alphablox to Cubing Services, to access and interact with the data.

To define the data source, perform the following steps:

1. Open the Alphablox Admin pages in a browser with the following URL:

   `http://<your server>:<port>/AlphabloxAdmin/home`

2. Select the **Administration** → **Data Sources**.
3. Click **Create** to create a data source connection.
4. In the Data Source Name field, enter your data source name and select **IBM Cubing Services Adapter** from the Adapter menu.
5. Specify the appropriate connection parameters to connect to your Cubing Services server, as shown in Figure 11-17. Click **Save**.
6. Click **Test Selected Data Source** to verify that you can successfully connect.

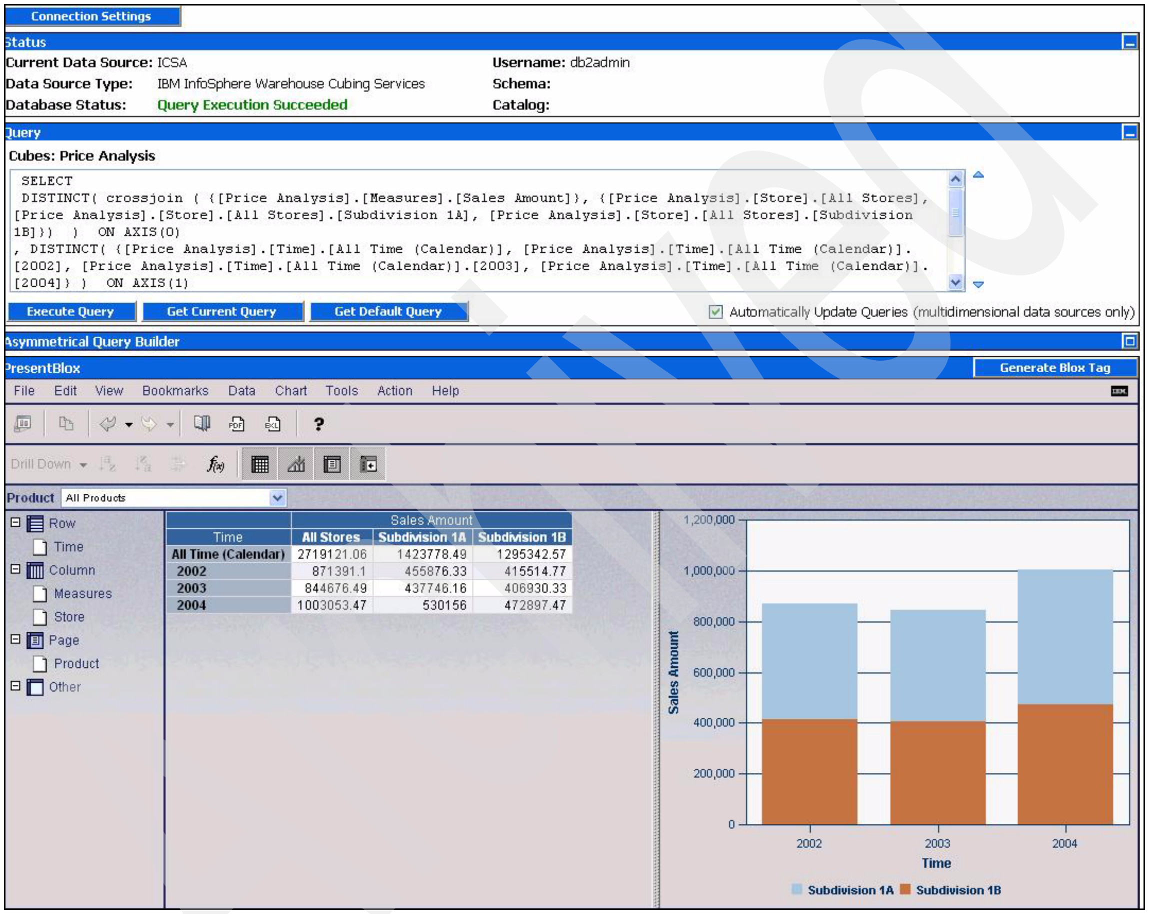


Data Source Name	ICSA
Description	
Adapter	IBM Cubing Services Adapter
Cubing Services Host Name	ts11.svl.ibm.com
Port Number	9000
Default Username	db2admin
Default Password	••••••••••••••
Use IBM Alphablox Username and Password	No
Maximum Rows	10000
Maximum Columns	10000
Maximum DrillThrough Rows	100000

Figure 11-17 Defining a data source to connect to Cubing Services

11.2.4 Accessing a Cubing Services data source using Query Builder

Now that you have defined a data source to Cubing Services, the easiest way to test access to that data source is to use Alphablox Query Builder. The Query Builder is an application that is installed by default when Alphablox is installed. It provides easy access to the data source and can help with developing and testing queries. Query Builder is shown in Figure 11-18.



Current Data Source: ICSA
Username: db2admin
Data Source Type: IBM InfoSphere Warehouse Cubing Services
Schema:
Database Status: Query Execution Succeeded
Catalog:

Cubes: Price Analysis

```
 SELECT
 DISTINCT( crossjoin ( {[Price Analysis].[Measures].[Sales Amount]}, {[Price Analysis].[Store].[All Stores],
[Price Analysis].[Store].[All Stores].[Subdivision 1A], [Price Analysis].[Store].[All Stores].[Subdivision
1B]})  )  ON AXIS(0)
, DISTINCT( {[Price Analysis].[Time].[All Time (Calendar)], [Price Analysis].[Time].[All Time (Calendar)].
[2002], [Price Analysis].[Time].[All Time (Calendar)].[2003], [Price Analysis].[Time].[All Time (Calendar)].
[2004]} )  ON AXIS(1)
```

✓ Automatically Update Queries (multidimensional data sources only)

	Sales Amount		
Time	All Stores	Subdivision 1A	Subdivision 1B
All Time (Calendar)	2719121.06	1423778.49	1295342.57
2002	871391.1	455876.33	415514.77
2003	844676.49	437746.16	406930.33
2004	1003053.47	530156	472897.47

Figure 11-18 Alphablox Query Builder

To access Query Builder, perform the following steps:

1. Open the Alphablox Admin pages in a browser, using the following URL:

   `http://<your server>:<port>/AlphabloxAdmin/home`

2. Select **Administration** → **IBM Alphablox Query Builder**.

3. Click **Connection Settings** and select the Cubing Services data source.
4. Click **Connect**. Upon connection, you see the string connected in the status frame.

   Above the Query text window you see the list of cubes running in the Cubing Services server. The default query is automatically displayed in the Query text window.
5. To view the results of the query, click **Execute Query**. Results are presented in an out-of-the-box Alphablox PresentBlox.

   You can drag dimensions from DataLayoutBlox into, or out of, GridBlox and PageBlox.
6. Right-click the grid or chart to access analysis actions such as drilling, sorting, filtering, and pivoting to display interactively the desired view of data.

Note: Every time you perform analysis actions, the MDX query can be dynamically updated in the query text box by selecting the Automatically Update Queries feature. This is a useful feature, if you are new to MDX language.

11.3 Using the Excel client interface

IBM InfoSphere Warehouse 9.5.1 delivers a Cubing Services ODBO provider (IBM OLE DB Provider for OLAP) which enables Microsoft Excel to use a standard method of interacting with the Cube Server, and thereby perform sophisticated multidimensional analysis by importing the cube data into Microsoft Excel pivot tables.

Pivot Table Services is a component of Microsoft Excel 2003 and 2007. It allows the user to connect to Cubing Services cubes and explore the dimensions in the cubes. The user can then drag dimensions and measures onto the worksheet.

In this section we describe how to perform the following activities to demonstrate the use of the Excel interface with Cubing Services:

- Connect to the cube server from Excel
- Retrieve cube metadata from the cube server
- Create reports by dragging and dropping dimensions and measures from the cube list into the report area

11.3.1 System requirements

OLE DB for OLAP is an industry standard for multidimensional data processing. You can use IBM OLE DB Provider for OLAP to retrieve data from the Cubing Services OLAP server using MS Excel, as shown in Figure 11-19.

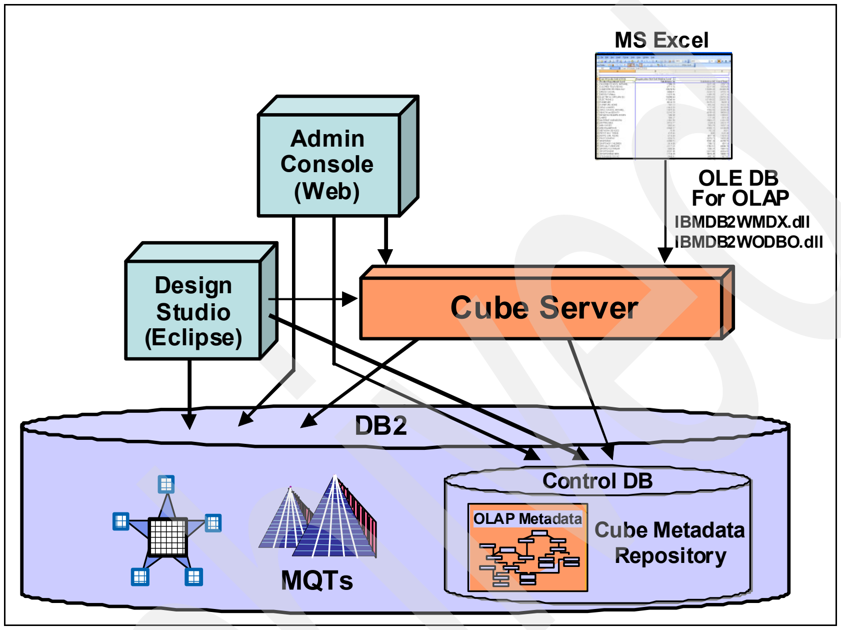


Figure 11-19 Using MS Excel as the client to your Cube Server

Supported platforms

The IBM OLE DB Provider for OLAP (ODBO) driver can be installed and used in the following client platforms:

- Windows Vista 32-bit
- Windows XP 32-bit
- Windows Server 2003 32-bit
- Windows Server 2003 64-bit

The following .NET runtime modules need to be installed in the client platforms:

- .NET 2.0
- .NET 3.0

To download .Net runtime modules visit the following web pages:

- `http://www.microsoft.com/downloads/details.aspx?FamilyID=10cc340b-f857-4a14-83f5-25634c3bf043`
- `http://www.microsoft.com/downloads/details.aspx?FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5`

Supported MS Excel versions

The IBM OLE DB Provider for OLAP driver can be used with the Pivot Table Service on the following Excel versions:

- Excel 2003 with Pivot Table Service 8.0
- Excel 2007 with Pivot Table Service 9.0

Attention: If Excel 2003 is used, a MS Office Update (KB907417) is required.

11.3.2 Connecting to the Cube Server from MS Excel

In this section we describe using MS Excel to connect with the Cube Server. This enables Microsoft Excel to use a standard method of interacting with the Cube Server, which enables the importing of the cube data into Microsoft Excel pivot tables. That allows the Excel user to connect to Cubing Services cubes and explore their dimensions.

To get this process started, you must first connect to the Cube Server from MS Excel. To do that, perform the following steps:

1. Launch MS Excel.
2. On the Data menu, click **Import External Data** and **Import Data,** as shown in Figure 11-20 on page 473.

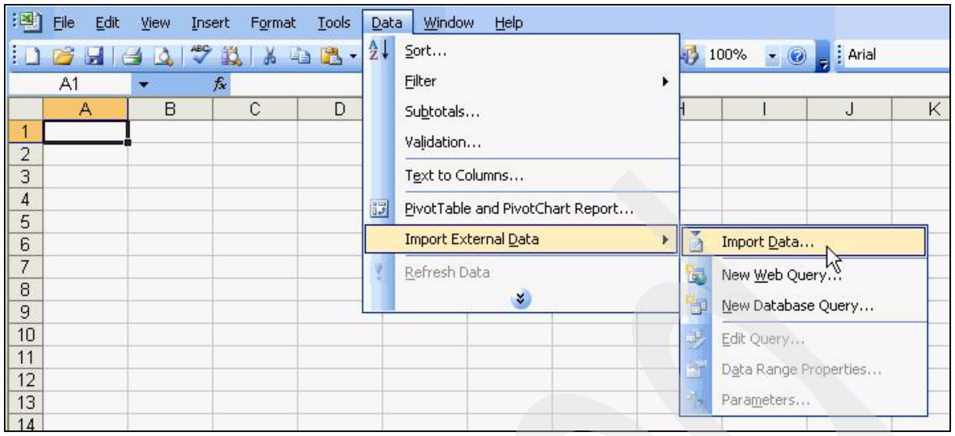
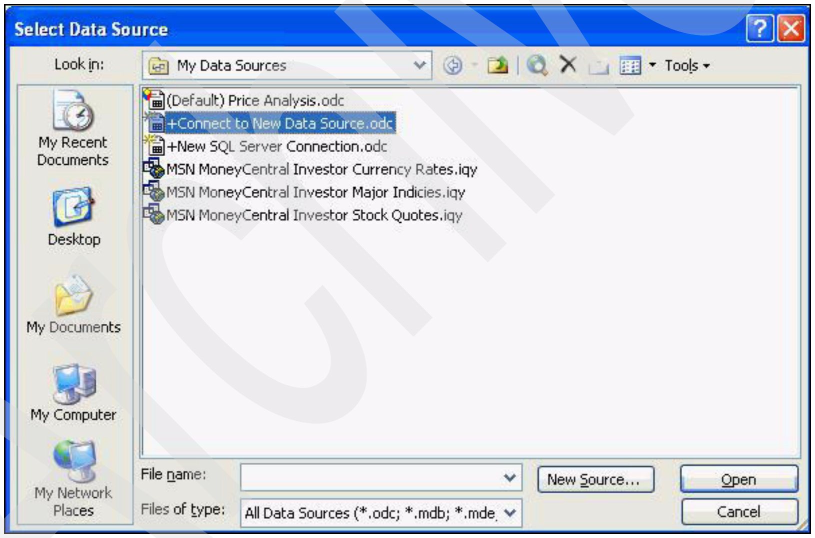


Figure 11-20 Importing data in MS Excel

3. In the “Select Data Source” window (Figure 11-21), select **Connect to New Data Source** and click **Open**.

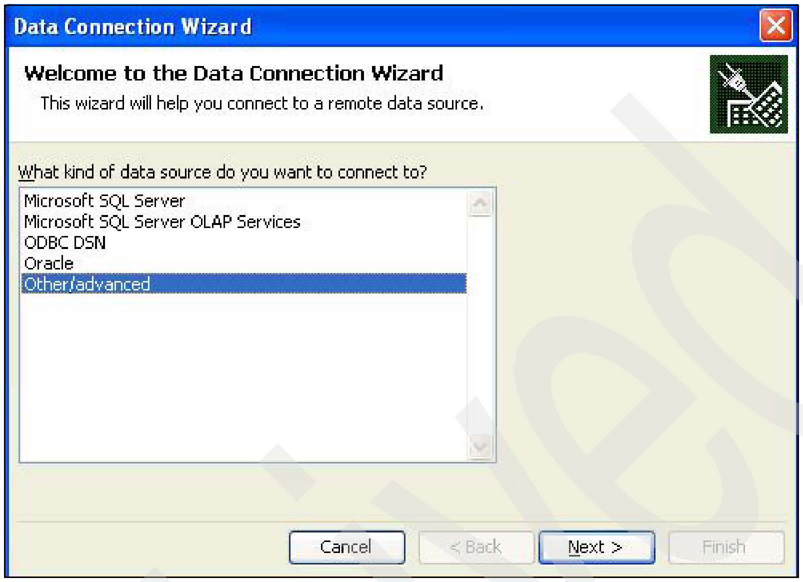


(Default) Price Analysis.odc
+Connect to New Data Source.odc
+New SQL Server Connection.odc
MSN MoneyCentral Investor Currency Rates.iqy
MSN MoneyCentral Investor Major Indicies.iqy
MSN MoneyCentral Investor Stock Quotes.iqy

File name:
Files of type: All Data Sources (*.odc; *.mdb; *.mde

Figure 11-21 Select Data Source window

4. In the “Data Connection Wizard” window (Figure 11-22), select **Other/Advanced** and click **Next**.


Data Connection Wizard

Welcome to the Data Connection Wizard

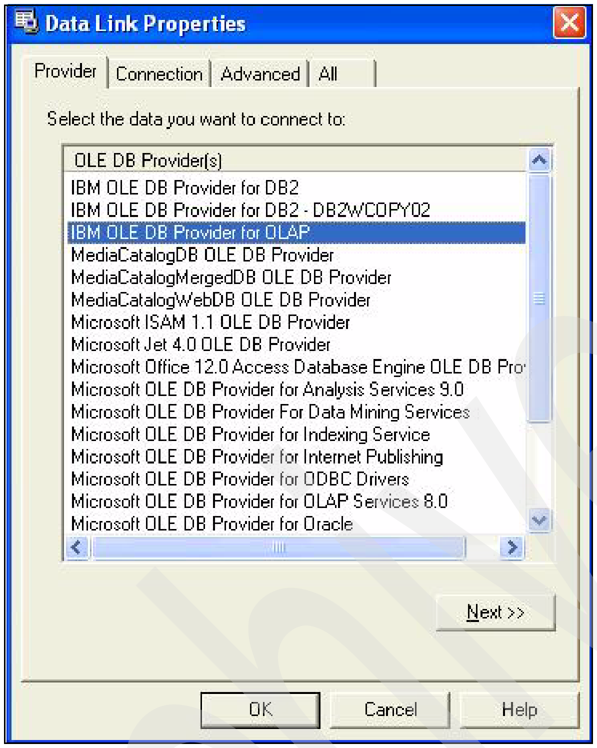
This wizard will help you connect to a remote data source.

What kind of data source do you want to connect to?

Microsoft SQL Server
Microsoft SQL Server OLAP Services
ODBC DSN
Oracle
Other/advanced

Figure 11-22 Data Connection Wizard window

5. In the “Data Link Properties” window (Figure 11-23), select **IBM OLE DB Provider for OLAP** and click **Next**.


Data Link Properties

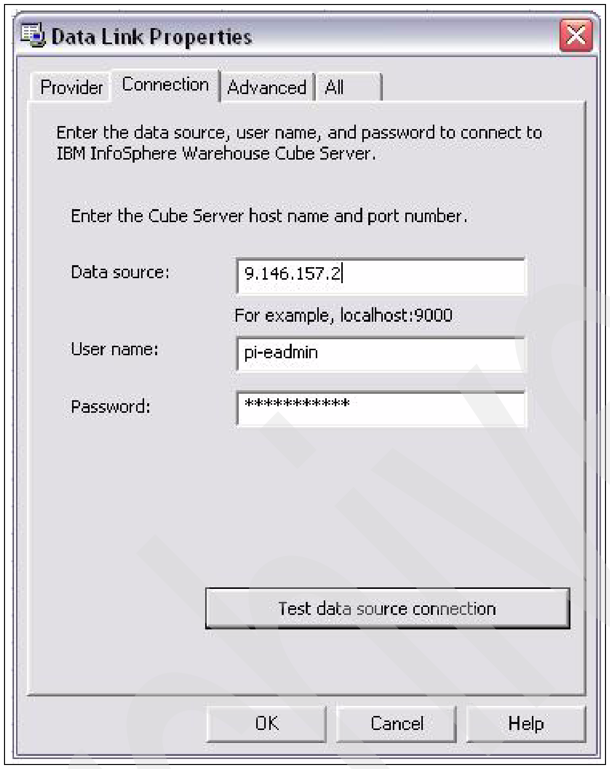
Provider | Connection | Advanced | All

Select the data you want to connect to:

OLE DB Provider(s)
IBM OLE DB Provider for DB2
IBM OLE DB Provider for DB2 - DB2WCOPY02
IBM OLE DB Provider for OLAP
MediaCatalogDB OLE DB Provider
MediaCatalogMergedDB OLE DB Provider
MediaCatalogWebDB OLE DB Provider
Microsoft ISAM 1.1 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft Office 12.0 Access Database Engine OLE DB Pro
Microsoft OLE DB Provider for Analysis Services 9.0
Microsoft OLE DB Provider For Data Mining Services
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services 8.0
Microsoft OLE DB Provider for Oracle

Figure 11-23 Data Link Provider Properties window

6. Input values for the Data Source (including the IP address and the port number of the Cube Server being connected to), User name and Password fields, as shown in Figure 11-24.



Enter the data source, user name, and password to connect to IBM InfoSphere Warehouse Cube Server.

Enter the Cube Server host name and port number.

Data source: 9.146.157.2

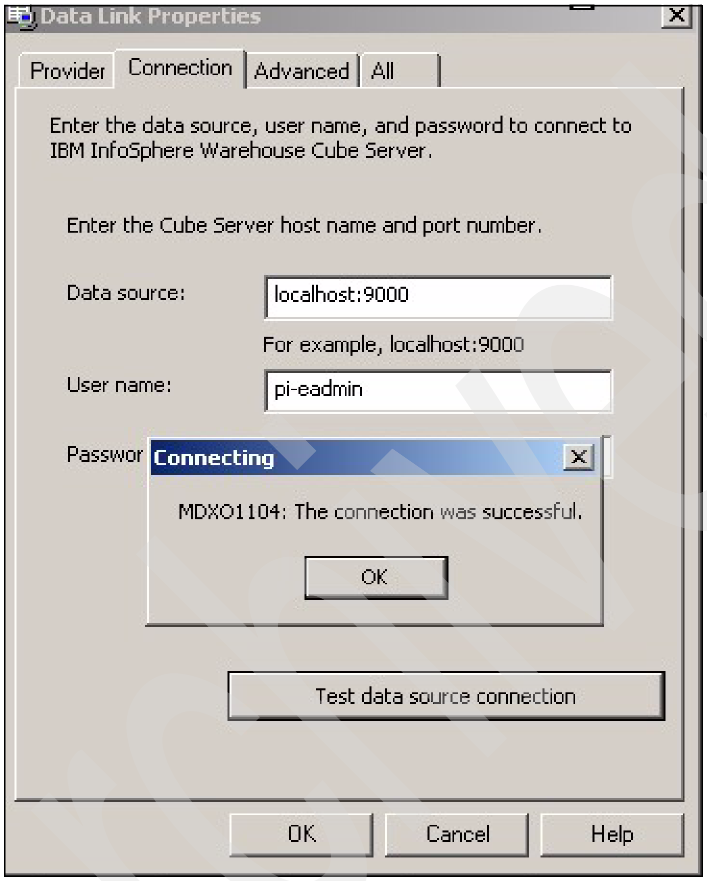
For example, localhost:9000

User name: pi-eadmin

Password: ***********

Figure 11-24 Data Link Connection Properties window

7. Test the connection to the Cube Server by clicking **Test data source connection**, then click **OK**. Close the Data Link Properties panel by clicking **OK**, as shown in Figure 11-25.



Enter the data source, user name, and password to connect to IBM InfoSphere Warehouse Cube Server.

Enter the Cube Server host name and port number.

Data source: localhost:9000

For example, localhost:9000

User name: pi-eadmin

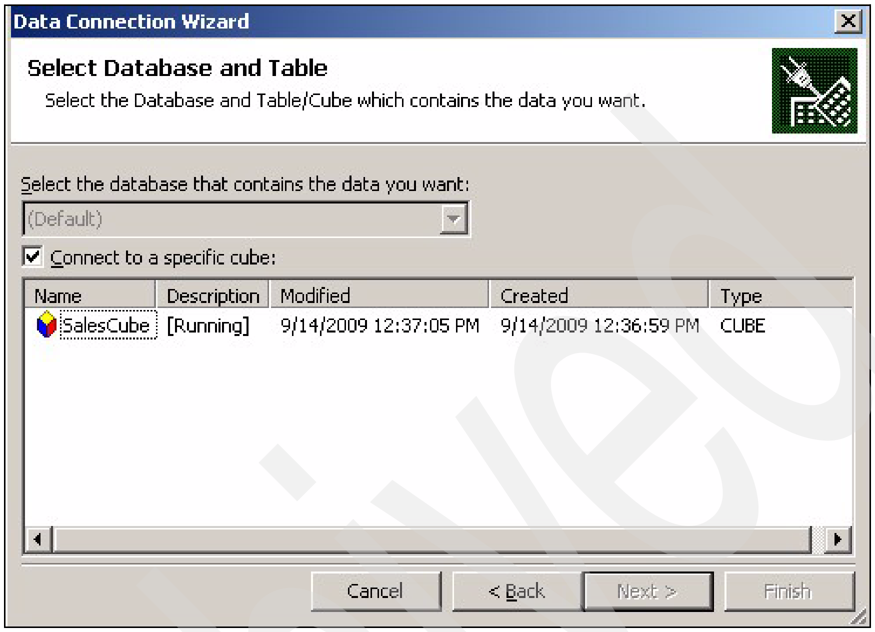
Passwor

Connecting

MDXO1104: The connection was successful.

Figure 11-25 Test connection

8. In the “Select Database and Table” panel (Figure 11-26), choose the cube to which you would like to connect and click **Next**.



Data Connection Wizard

Select Database and Table

Select the Database and Table/Cube which contains the data you want.

Select the database that contains the data you want:

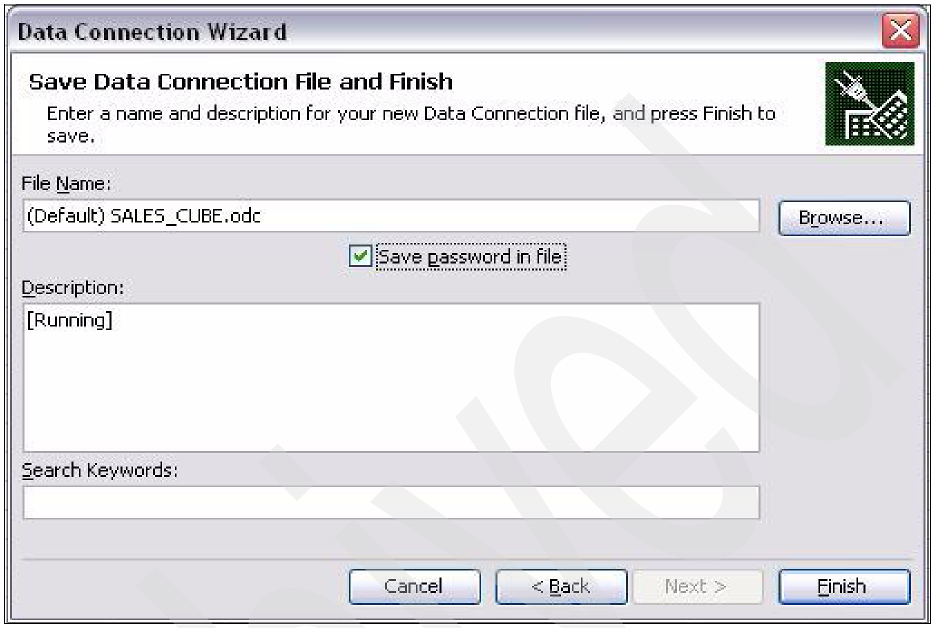
(Default)

☑ Connect to a specific cube:

Name	Description	Modified	Created	Type
SalesCube	[Running]	9/14/2009 12:37:05 PM	9/14/2009 12:36:59 PM	CUBE

Figure 11-26 Select Database and Table window

9. To reuse an external data source connection, the password must be saved in Excel. Select the **Save Password in File** check box, as in Figure 11-27.



Enter a name and description for your new Data Connection file, and press Finish to save.

File Name: (Default) SALES_CUBE.odc

✓ Save password in file

Description: [Running]

Search Keywords:

Figure 11-27 Saving a data source connection

10. Click **Finish** to create the new connection.

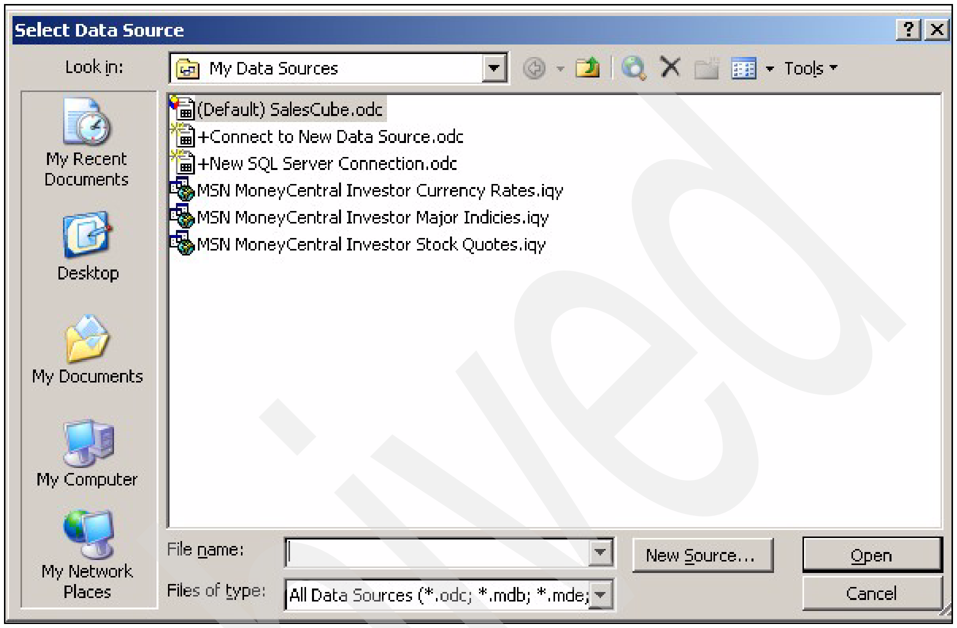
11.3.3 Retrieving data using MS Excel

After connecting to a cube, the next task is to retrieve and manipulate data to help answer your business questions. The following example shows you how to retrieve data from the selected cube in Cubing Services, and to execute MDX queries against the cube by dragging from the cube dimensions and measures list into the report area.

Note: MDX queries are transparent to your users.

To retrieve cube data in MS Excel, perform the following steps:

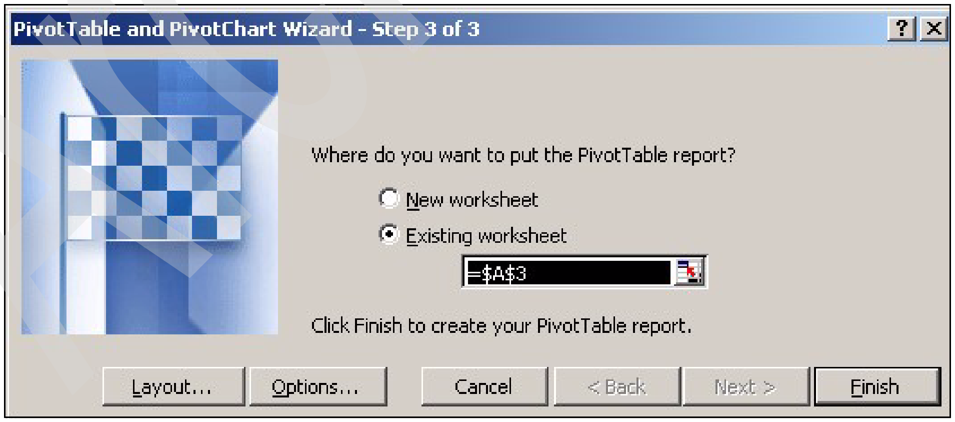
1. On the Data menu, click **Import External Data** and **Import Data**. Select your cube and click **Open**, as shown in Figure 11-28.



Select Data Source
Look in: My Data Sources
Tools
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
(Default) SalesCube.odc
+Connect to New Data Source.odc
+New SQL Server Connection.odc
MSN MoneyCentral Investor Currency Rates.iqy
MSN MoneyCentral Investor Major Indicies.iqy
MSN MoneyCentral Investor Stock Quotes.iqy
File name:
Files of type: All Data Sources (*.odc; *.mdb; *.mde;
New Source...
Open
Cancel

Figure 11-28 Connecting to a cube

2. Click **Finish** to close the Pivot Table Wizard, as shown in Figure 11-29.



PivotTable and PivotChart Wizard - Step 3 of 3
Where do you want to put the PivotTable report?
New worksheet
Existing worksheet
=A3
Click Finish to create your PivotTable report.
Layout...
Options...
Cancel
< Back
Next >
Finish

Figure 11-29 Pivot table wizard

3. To build the pivot report, drag the dimensions to either the row, column, or page area. Drag the measures into the data area of the report. See Figure 11-30.

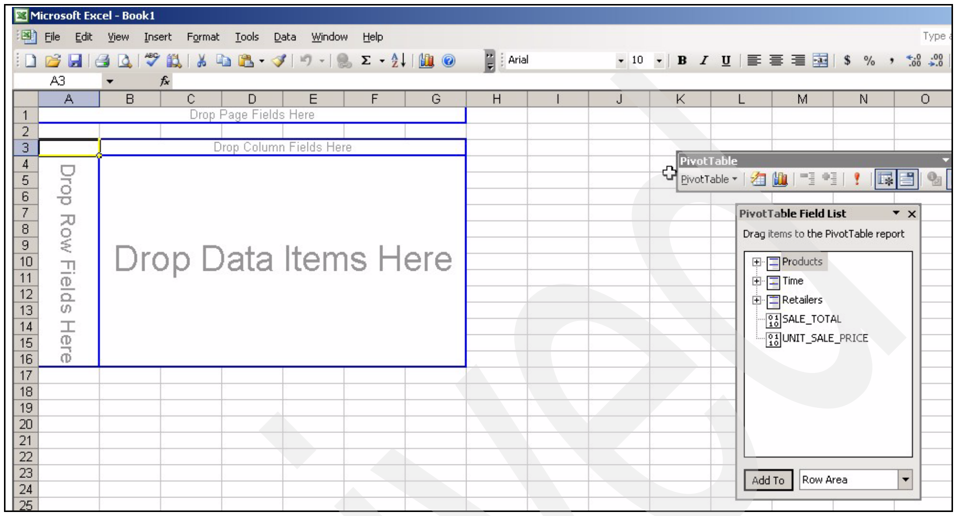


Figure 11-30 Cube metadata is available in the Pivot Table field list

12

Case Study: InfoSphere integration on System z

In this chapter we present a case study for integrating InfoSphere Q Replication and InfoSphere Warehouse on System z.

The processes needed to capture source data, transform it, and update the data warehouse, are critical elements in the day-to-day operations for updating the warehouse. As data warehouses and business intelligence applications gain importance in corporations, IT staff are increasingly asked to provide more of a continuous feed into the data warehouse.

Figure 12-1 on page 484 shows the four different stages of data before it can be used for decision making. The first stage is acquiring the data to store in the warehouse. This data is then optimized or transformed. Presentation tools such as Cognos and excel then use this data to present reports.

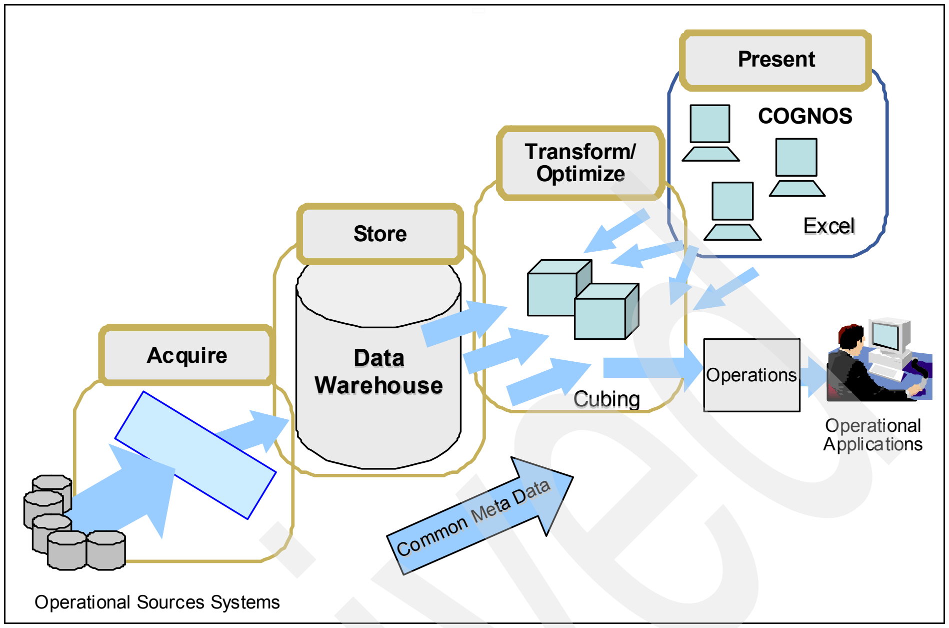


Figure 12-1 The stages of data

Historically, data in the warehouse has been refreshed based on business cycles (for example, quarterly for financial records and monthly for sales). Refreshing data monthly or quarterly was fine when the reports were being used for making strategic decisions. As data volumes increased and batch windows shrank, it became impossible to handle full extract/replace operations that at one time were the norm. Large batch operations can force CPU spikes that raise System z premiums for businesses. It can be cost effective to process data in smaller batches. Because current trends favor operational BI, these applications can benefit from data that is more current.

There are a number of methods that can be used to ensure that the data in the warehouse is current. New or changed data can be identified through queries against the source data, but this has many issues (delete operations are lost, queries can cause contention against the source data and this can also become a costly large batch process). These issues have led to greater adoption of change data capture techniques that avoid contention and capture insert, update, and delete operations against the source. These change data capture techniques are scalable and ensure that near real-time data is available for reports.

In this chapter we use a case study example to show how to integrate InfoSphere Q Replication with InfoSphere Warehouse SQL Warehousing functions to acquire source data, transform that data, and populate a data warehouse based on that data to satisfy a continuous feed to the data warehouse. InfoSphere Q Replication is a high-volume, low-latency replication solution that uses WebSphere MQ message queues to transmit transactions between source and target databases or subsystems. The SQL Warehousing tool facilitates intra-warehouse SQL-based data movement and transformation. It is discussed in detail in Chapter 5, “Data movement and transformation” on page 75.

12.1 InfoSphere Q Replication

IBM provides two different solutions that you can use to replicate data to and from relational databases:

- SQL replication
- Q replication

For a comparison of the Q Replication and SQL Replication features, see *Comparison of SQL replication and Q replication—Overview* at the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.im.iis.db.repl.intro.doc/topics/iiyrcintdif00.html?resultof=%22%63%6f%6d%70%61%72%69%73%6f%6e%22%20%22%53%51%4c%22%20%22%73%71%6c%22%20

SQL replication captures changes to sources and uses staging tables to store committed transactional data. The changes are read from the staging tables and replicated to corresponding target tables. With staging tables, data can be captured and staged once for delivery to multiple targets, in different formats, and at different delivery intervals. SQL replication can be used for a variety of purposes that require replicated data, including capacity relief, feeding data warehouses and data marts, and auditing change history

Q replication is a replication solution that can replicate large volumes of data at low levels of latency. Q replication captures changes to source tables and converts committed transactional data to messages. The data is not staged in tables. As soon as the data is committed at the source and read by Q replication, the data is sent. The messages are sent to the target location through WebSphere MQ message queues. At the target location, the messages are read from the queues and converted back into transactional data. The transactions are applied to your target tables with a highly parallelized method that preserves the integrity of your data.

Q replication can be used for a variety of purposes that require replicated data, including failover, capacity relief, geographically distributed applications, and data availability during rolling upgrades or other planned outages.

We use unidirectional Q Replication in our case study, but SQL replication can be easily plugged in to feed the warehouse data. Figure 12-2 shows a schematic of how Q Replication works.

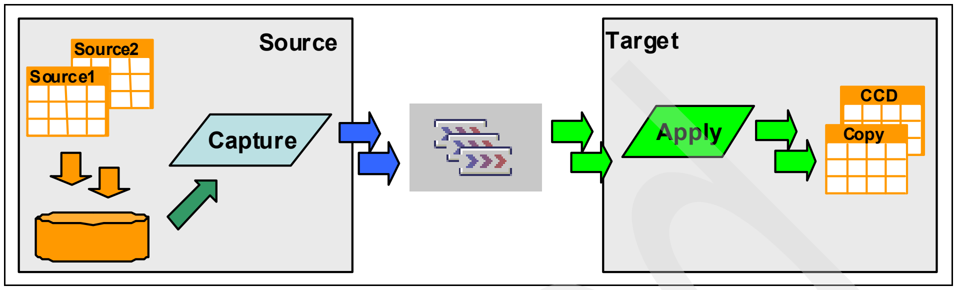


Figure 12-2 InfoSphere Q Replication schematic

To capture a change from a DB2 table, that table has to be defined as a replication source, which sets up the DB2 environment to start capturing changes to that table. To capture changes, the Q Capture component monitors the DB2 log to detect any changes that occur for the replication sources. It inserts those changes into the WebSphere MQ queues. Detecting data changes by monitoring the log has the minimum impact on the operational transaction. The Q capture program is a continuously running process which is running anytime the DB2 database is up.

When we want to apply changes to the target tables, the Q Apply process applies the changes from the queue.

We use the Replication Center to define the replication sources. The following steps are necessary to define replication:

1. Create Capture control tables.

   The infrastructure for the Q Capture program on the server where your source tables are located is a small set of tables that store information about the source and target tables. A Q Capture program uses this information to know the location of the source and target tables. The Q Capture program looks for changes to the source tables and places committed changes on queues.

2. Create Apply control tables.

   Before applying changes to the targets, the infrastructure for the Q Apply program on the server where the target tables are located needs to be set up. The infrastructure is a small set of tables that store information about the target and source tables. A Q Apply program uses this information as it reads transactions off of queues and applies them to the target tables.

3. Create Q Subscriptions.

   A Q subscription is created to pair a source table with a target table and to specify the queues to use for sending and receiving changes. When a source table is identified in a Q subscription, the columns that are of interest are specified. A WHERE clause can also be specified to filter rows. If a transaction makes changes to these columns and rows and commits those changes, the Q Capture program places messages containing the changes in a send queue. The Q Apply program reads the messages from a receive queue and apply the changes to the target table.

Figure 12-3 shows the set up of unidirectional Q Replication.

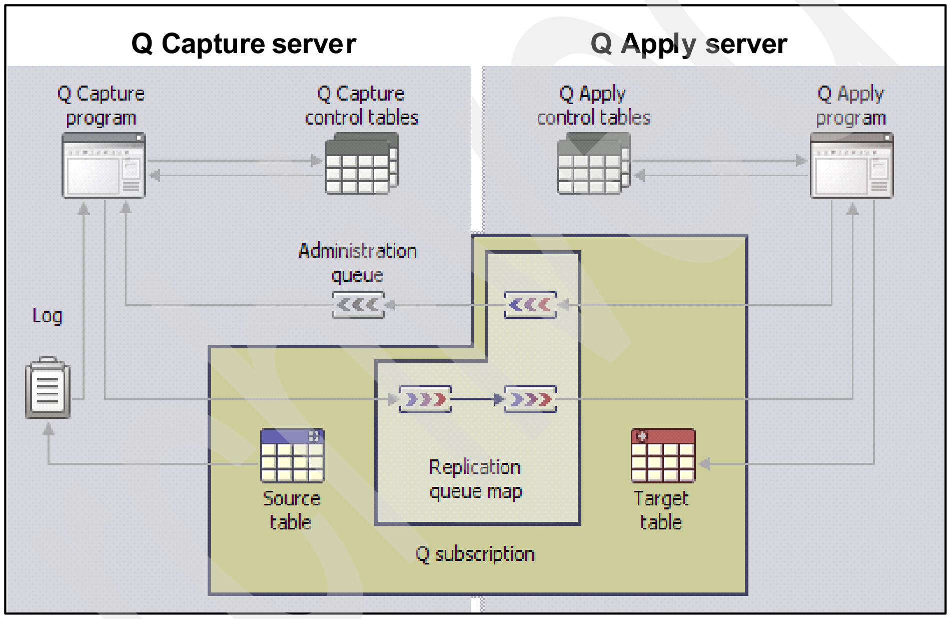


Figure 12-3 Unidirectional Q Replication setup

For more information about subscriptions and how to create them, see the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.swg.im.iis.repl.qrepl.doc/topics/iiyrqsubtunicrts.html

12.2 Case study overview

The primary purpose of any population process is to extract data from a number of sources, move the data to the destination system, transform the data in various ways, and finally update the data warehouse. Sounds simple? However, the tasks involved are usually more complex:

- We might have a number of source structures across many systems.
- We do many more functions in preparing the data for the warehouse. Not only must we worry about getting the data into a new structure, we must also enhance the data by converting codes into something meaningful, calculating new data, adding data from external sources, geo-coding the data, and so forth. We also have to make sure the data is clean and reconciled across the different source systems.
- We must maintain history. The population process must be repeatable, as we add data to the warehouse at periodic intervals.

In addition to all these requirements, we have to maintain this application as new sources are added, as new business rules must be applied, and as new cleansing requirements arise.

In this case study, we use the example of a fictional company named JK Superstore. JK Superstore is a retail enterprise with a chain of department stores. Their product lines include clothing, shoes, beauty products, home furnishings, and electronics. The company has grown steadily for the last several years, plans to extend business into new markets, and wants to ensure that it maintains its profitability during the expansion. The JK Superstore data warehousing team is committed to consolidating the company data into a DB2 database, and a new warehouse that provides a consistent data source for analysis and reporting. The IT department has created schemas for the transactional and aggregated data for JK Superstore retail chain for pricing and sales analysis.

JK Superstore has historically used a batch update to their data warehouse. Figure 12-4 on page 489 shows the left side and Figure 12-5 on page 489 shows the right side of a data flow that shows the fact table batch job. The purpose of the data flow is to load two similar input files that contain fact data for different time periods. The files are checked for duplicates, and merged with an SQL UNION expression and stored in a staging table. The data in the staging table is then fed into a key lookup operation, which matches the fact records with key values in two referenced tables before loading the fact table.

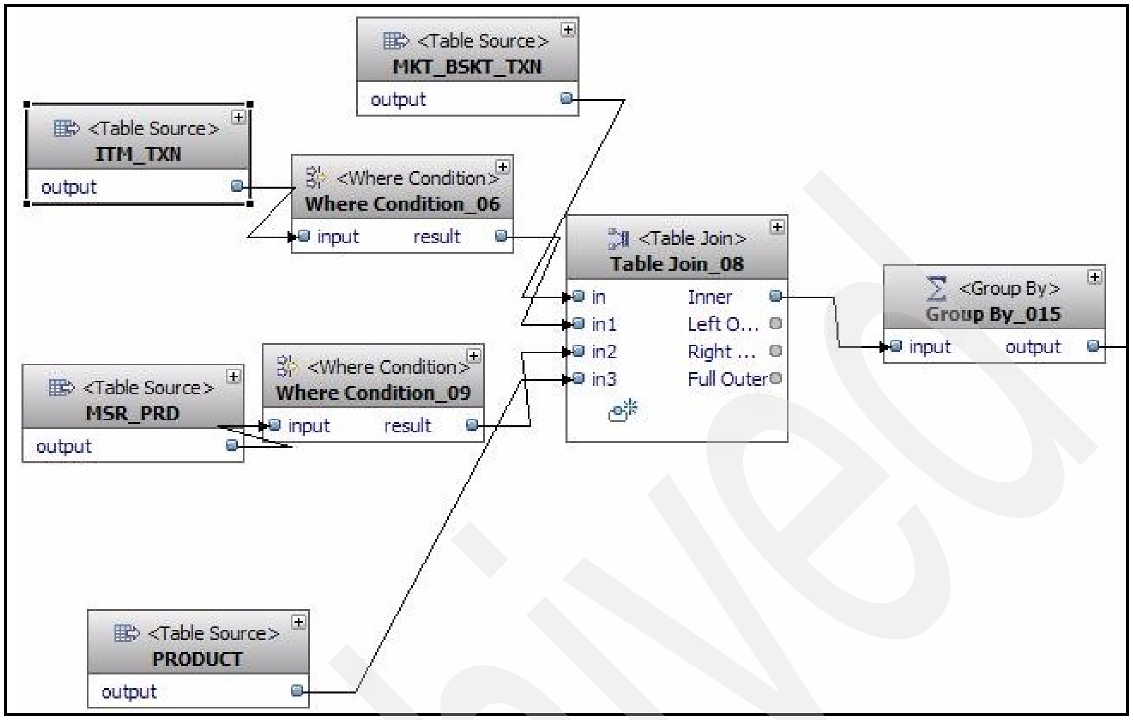


Figure 12-4 Batch job for fact table: Left side

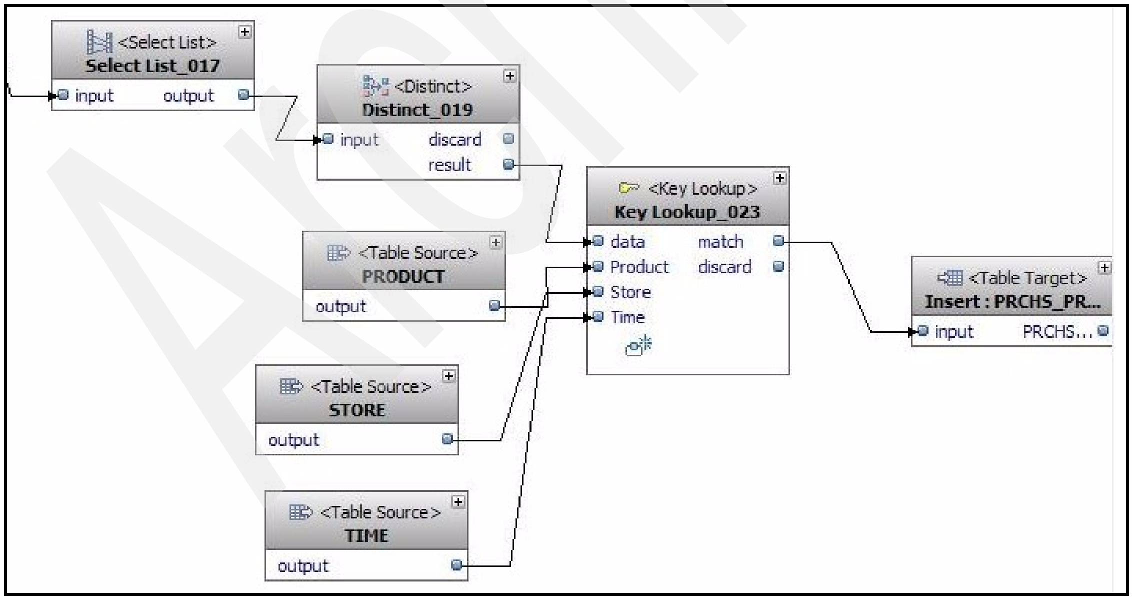


Figure 12-5 Batch job for fact table: Right side

The batch data flow for the product dimension table is lengthy and detailed. As such, it cannot be clearly represented in a single figure. Therefore, it has been segmented into three figures:

- Figure 12-6
- Figure 12-7 on page 491
- Figure 12-8 on page 492

The data in the product table comes from warehouse data that represents the product information in terms of different levels or hierarchies. For simplicity and efficiency in reporting and analytical applications, this data is flattened or denormalized before being loaded into the dimension tables. To transform the data, the data flow selects rows from the warehouse tables and joins them recursively. Each new join receives information at the next level. When all of these intermediate results are joined, the final result is a set of rows that contain all of the information about a particular product.

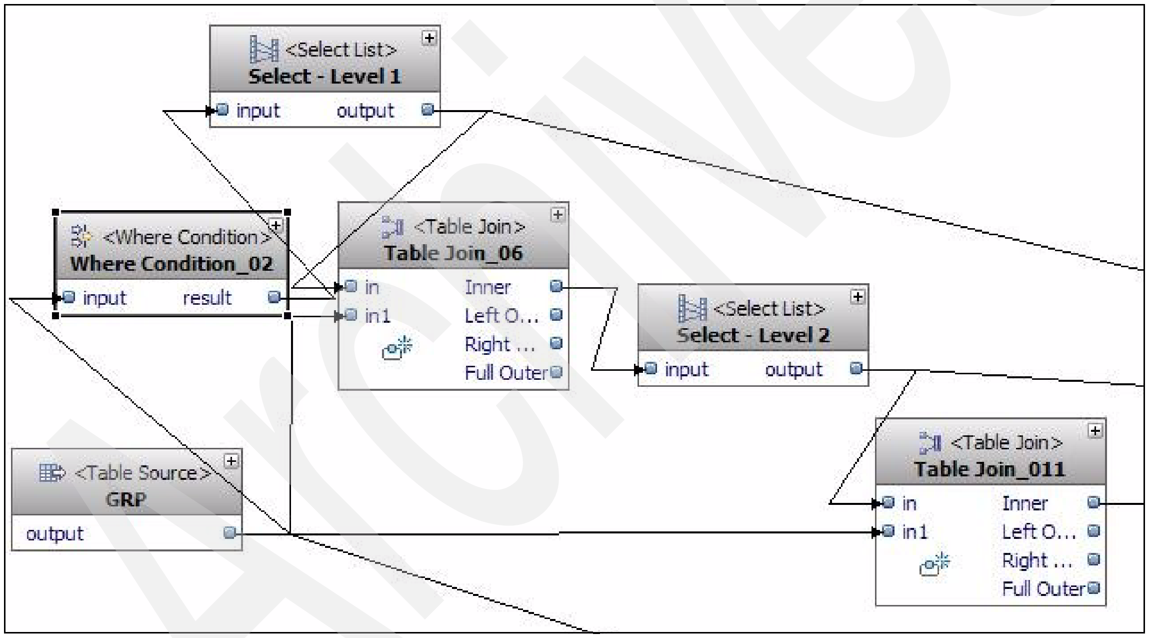


Figure 12-6 Batch data flow for the product dimension: Part 1

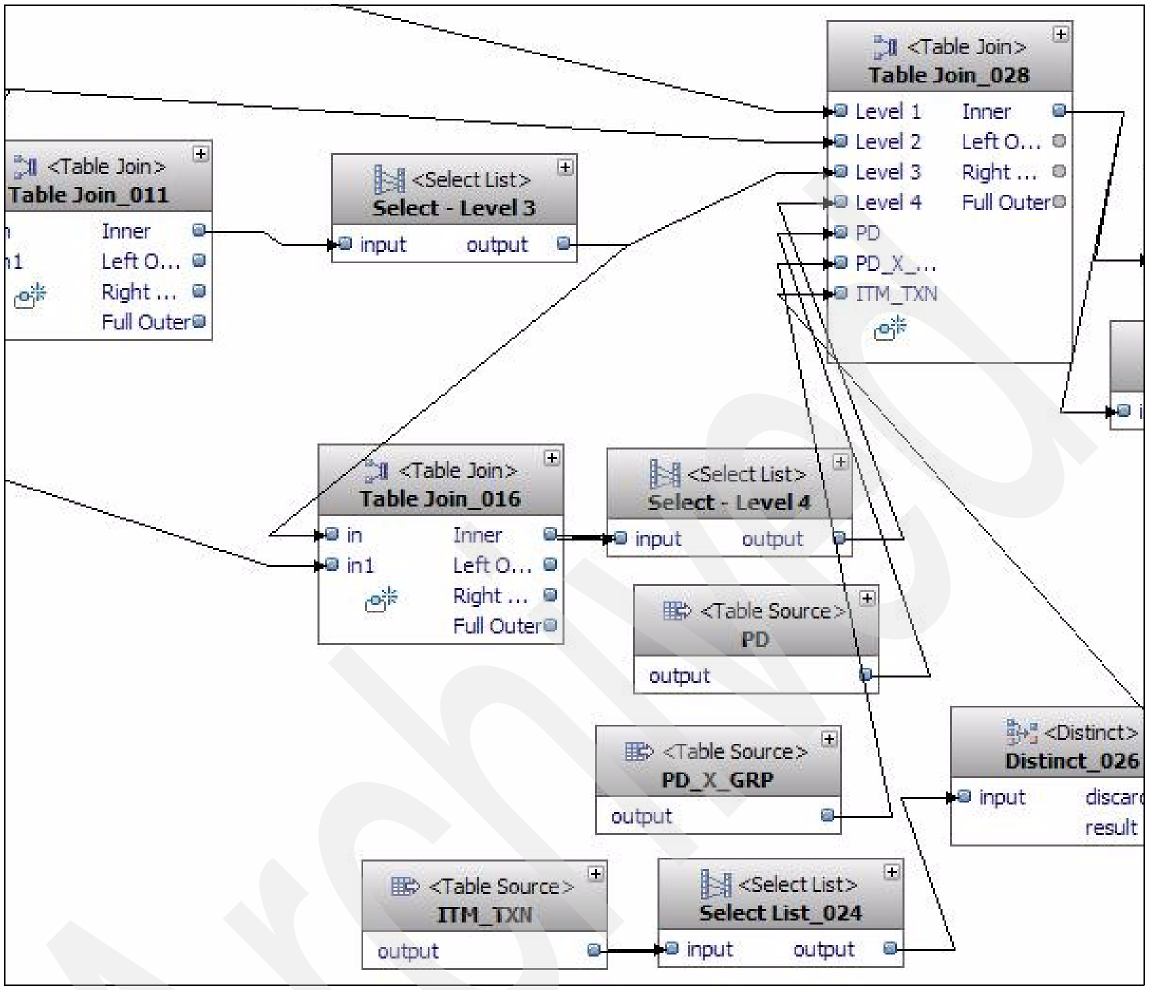


Figure 12-7 Batch data flow for the product dimension: Part 2

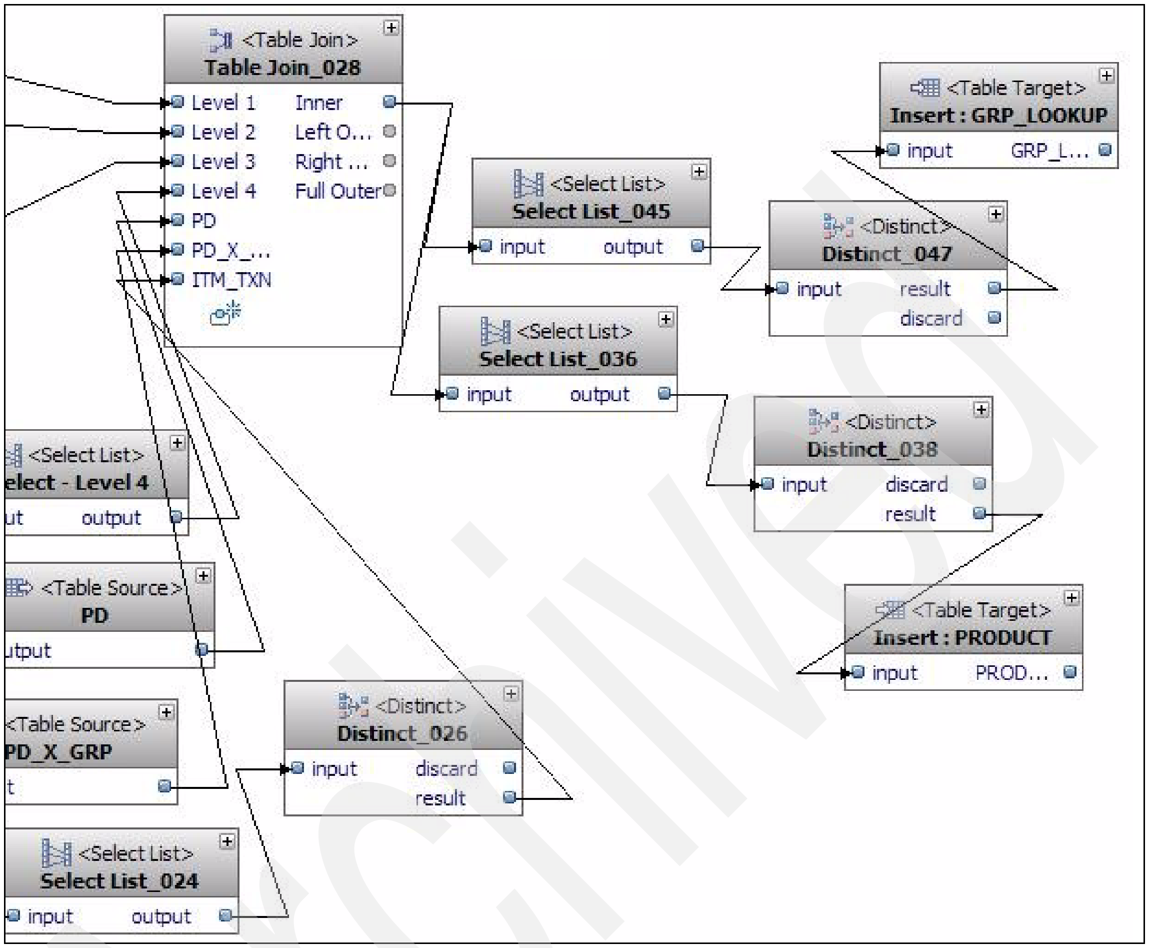


Figure 12-8 Batch data flow for the product dimension: Part 3

JK Superstore used monthly batch jobs, which helped in strategic decision making. They moved to weekly batch jobs and nightly jobs to create the reports that were necessary for decision making. Batch updates have the advantage of economies of scale, but the data is only available at the end of the batch job cycle. However, to remain competitive, business requirements require that the sales transactions of JK Superstore are loaded into the data warehouse much more frequently during the course of the business day (at least every half hour).

The IT department has to provide a way to capture those sales transactions and associated data such as store, product and time information as they happen, bring the data into the data warehouse environment, transform it and update the data warehouse tables appropriately. There are a number of ways to provide near real time data refreshes, but we discuss only one of them in the case study.

12.3 Case study architecture and design

JK Superstore has experience in developing data movement applications and have chosen to use a defined architecture as a framework for designing and implementing the warehouse population subsystem, rather than create ad-hoc processes in the interests of reusability, and interoperability. In this section, we describe the architecture JK Superstore uses to design the population subsystem.

12.3.1 Architecture

An architecture describes a system in terms of its components, relationships between those components, patterns that guide their composition, and constraints over those patterns. This consistency allows us to make intelligent decisions regarding the sometimes conflicting requirements of build-time requirements, adaptability, reusability, and testability, as well as the run-time requirements of efficiency, reliability, interoperability, and security.

Figure 12-9 shows the architecture for a data warehouse population subsystem. If you want to learn more about architecting a population subsystem, IBM offers courses on this and other related topics. Search for information on currently available courses at the following web page:

http://www-304.ibm.com/jct03001c/services/learning/ites.wss/zz/en?pageType=tp_search

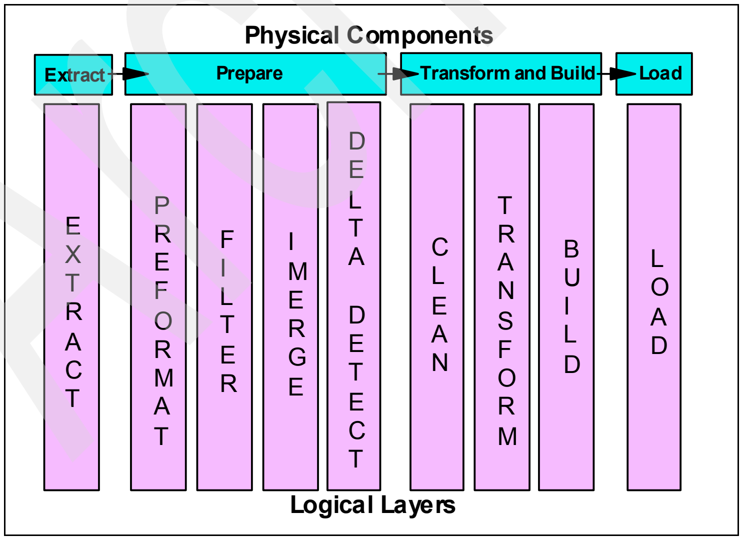


Figure 12-9 SQW processing architecture

This architecture splits the processing into layers such that each layer performs a specific function that does not overlap the other layers. The processes in each layer are encapsulated such that what happens in one layer is independent of the other layers. This approach allows us to implement a more modular approach, allows the layers to be modified independently, and allows layers to be moved to other platforms with minimal impact to the other layers. This architecture is described in detail in IBM Redbooks publication *Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager*, SG24-6513.

This architecture has physical components, each providing a set of services based on its logical layers. Each physical component is likely one or more technical functions implemented through user programs, database or file utility, services of an Extract-Transform-Load/Extract-Load-Transform tool or, most likely, a combination of all of these.

Depending on the complexity of the overall flow design and the capabilities of the available tools, there might not always be a clear delineation between all of the components and layers of the architecture. For example, IBM's Q Replication Server can handle both the Extract and the Prepare component. Or, in a simple scenario, all of the layers might be implemented in one tool.

12.3.2 High level design

In this section, we consider how we can implement the population subsystem to capture delta changes from tables that feed into the dimension and fact tables so we can consolidate them into the warehouse. The setup used for the case study is described in Figure 12-10.

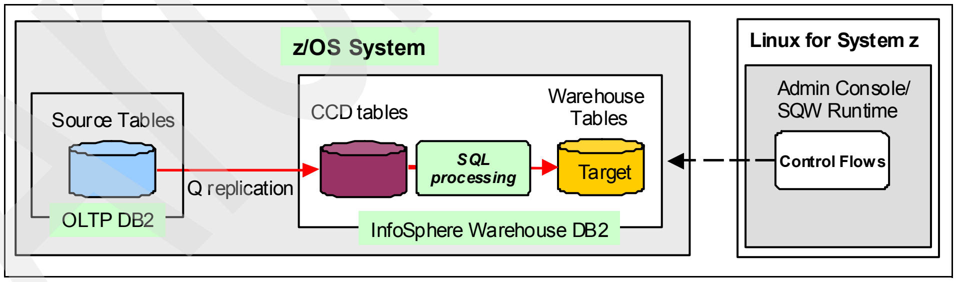


Figure 12-10 The Replication—InfoSphere Warehouse setup

Figure 12-11 depicts a high-level view of the data warehouse population subsystem. InfoSphere Q Replication is used to capture data changes in the On Line Transaction Processing (OLTP) database and write them to the intermediate tables in the Warehouse database. The InfoSphere Warehouse on System z data movement tool, SQW, is used to transform and load the data into the warehouse.

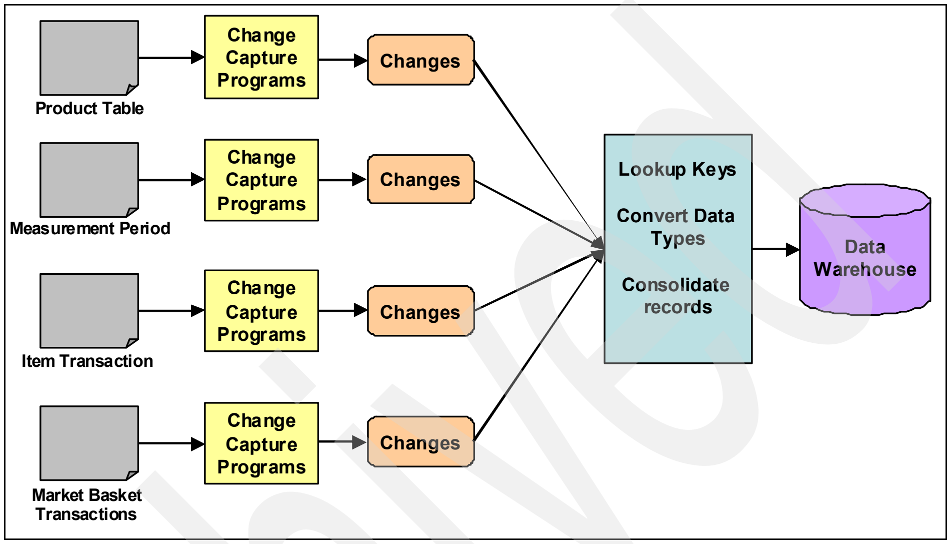


Figure 12-11 Population subsystem

Extract component

The Extract component implements the Extract layer and ensures that the data changes at the source are available to the population subsystem.

The objective of our Extract component (shown in Figure 12-12 on page 496) would be to capture every insert, update, or delete in all the source tables and make these changes available to the population subsystem. To do this, the InfoSphere Q Replication monitors the DB2 log for changes in the source tables. When a change is detected, whether an Insert, Update, or Delete, it is captured and a change record is put on an WebSphere MQ queue called a receive queue. WebSphere MQ is the transport mechanism that guarantees delivery of this message to a local queue on our Warehouse system. For more information about InfoSphere Q Replication, see the IBM documentation:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.im.iis.db.prod.repl.nav.doc/dochome/iiyrcnav_dochome.html

The data changes in the following tables are of interest to us:

- Product table

  The product table identifies goods and services that can be offered or sold by JK Superstore.

- Measurement period

  The measurement period table records the intervals of time at which measurements are captured in the warehouse.

- Item transaction table

  The item transaction table contains individual transactions that record the transfer of a single product item (or multiple identical product items) from JK Superstore to a customer, or vice versa. For example, one transaction can represent a single barcode scan at a checkout or a refund for a returned product.

- Market basket transaction table

  The market basket transaction table groups the individual transactions that occur together as one event. For example, one market basket transaction can represent the collection of items purchased by a single customer at one time.

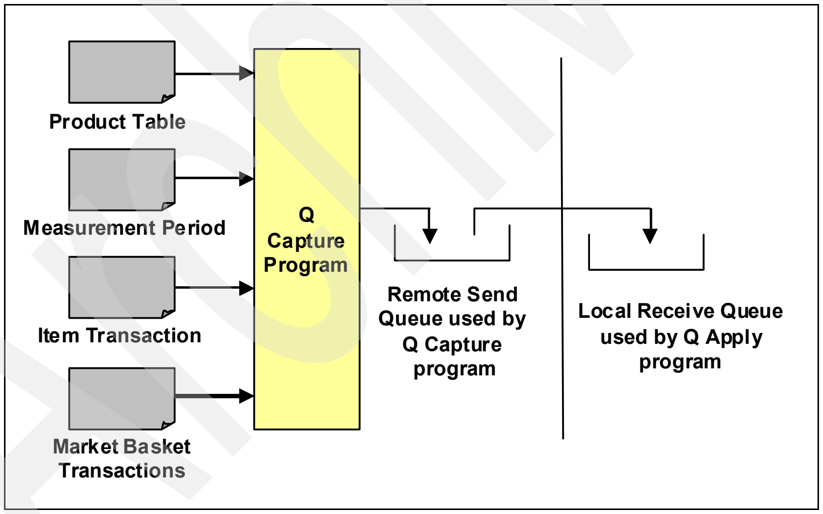

Product Table
Measurement Period
Item Transaction
Market Basket Transactions
Q Capture Program
Remote Send Queue used by Q Capture program
Local Receive Queue used by Q Apply program

Figure 12-12 Extract component

Prepare component

The Prepare component implements the Preformat, Filter, Intelligent Merge, and Delta Detection logical layers. The objective of this component is to bring the extracted data into a common storage mechanism, filter out records that are not needed, and intelligently merge data files together if necessary.

In our case study, because we are using Q Replication to capture changes directly from the source, we have automatic delta detection. Figure 12-13 describes the Prepare component. In this instance, we are just storing the data from the source system in DB2 for z relational tables. However, it is possible that filtering or record merging might have to be done. If we have a relatively long Q Apply interval, we could conceivably have more that one change record for any given key. In this case, we might decide to merge these separate change records into one. Fortunately, InfoSphere replication technology can do this for us, but we might have to provide this function for other change capture methods.

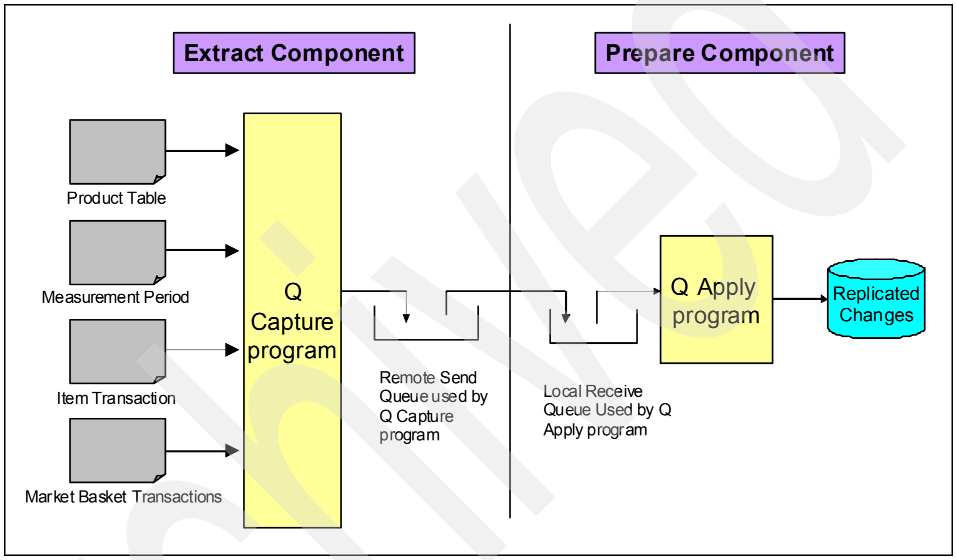


Figure 12-13 Prepare component

We have added the InfoSphere Replication Q Apply (Q Apply) program to the design. Q Apply takes the change records from the source, puts them into a standard record format, and then places them in a table in the warehouse database.

Transform and build component

The transform and build component implements the clean, transform and build layers. This component is the most complex of the components and might involve several steps and, perhaps, specialized tools. The output of this component is finalized change records with cleansed, transformed data to be input into the load component.

The following typical transformations that might occur:

- Assignment or lookup of surrogate keys
- Text case conversions
- US measurements to metric
- Converting cryptic codes into meaningful descriptions
- Separating/concatenating fields
- Normalizing or de-normalizing data
- Adding derived data

The following is a summary of the transformations that we need to perform for this case study:

- If a new record, assign a new surrogate key, otherwise, look up the surrogate key
- Map the data fields from the input format to the output format
- Do data type conversions where necessary
- For the measurement period table, ignore deletes and process only inserts and updates.
- For the product table, process updates, inserts and deletes.
- For the item transaction, and market basket transaction tables, the number of items in the market basket transaction table has to match the number of rows for a basket in the item transaction table.
- For the item transaction, and market basket transaction tables, process only inserts.

We build data flows using the InfoSphere SQW component to perform the above transformations. Figure 12-14 describes the transform component.

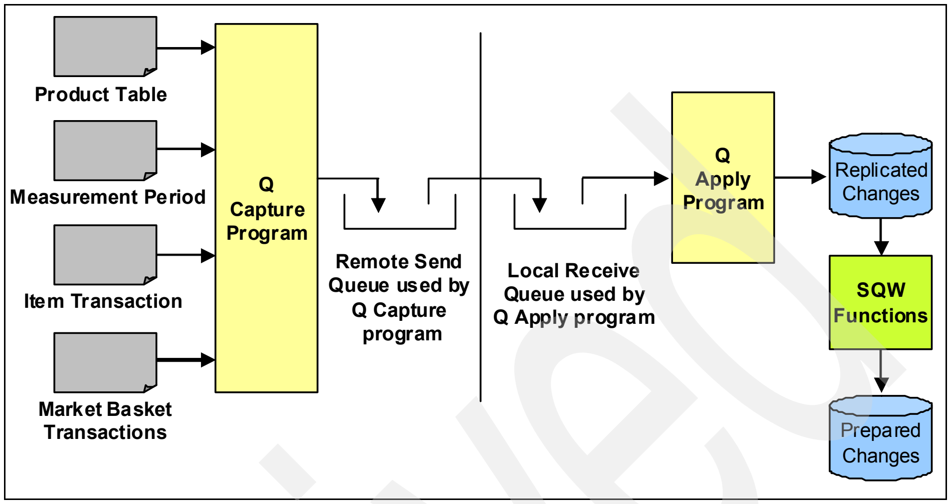


Figure 12-14 Transform component

Load component

The Load component implements the load layer and applies the prepared change records to the warehouse as defined by the business requirements.

The load component only has to do is to read the changes and apply the appropriate insert/update/delete action to the database with the minimal amount of processing. This component is responsible for committing changes to the database frequently enough not to hold locks too long. However, balancing the COMMIT frequency and associated COMMIT overhead with the overhead of processing those commits.

Figure 12-15 depicts the architecture of the Load component. This architecture is only a framework. Keep in mind that business requirements, simplicity of certain processes and technical capabilities might allow one or more architectural layers to be combined.

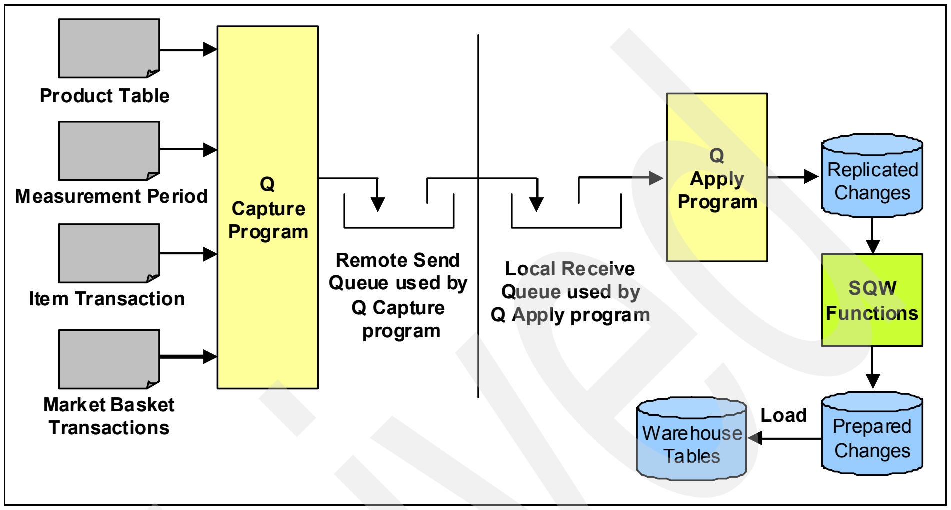


Figure 12-15 Load component

12.4 Implementation details

In this section, we describe the implementation details of the integration between InfoSphere Q Replication and InfoSphere Warehouse SQW functions.

12.4.1 Implementing the Replication Component

We are interested in the following source tables:

- Product table
- Measurement period
- Item Transaction table
- Market basket transaction table

To capture changes to these source tables using InfoSphere Q Replication, we need to create capture and apply control tables, and subscribe the source tables.

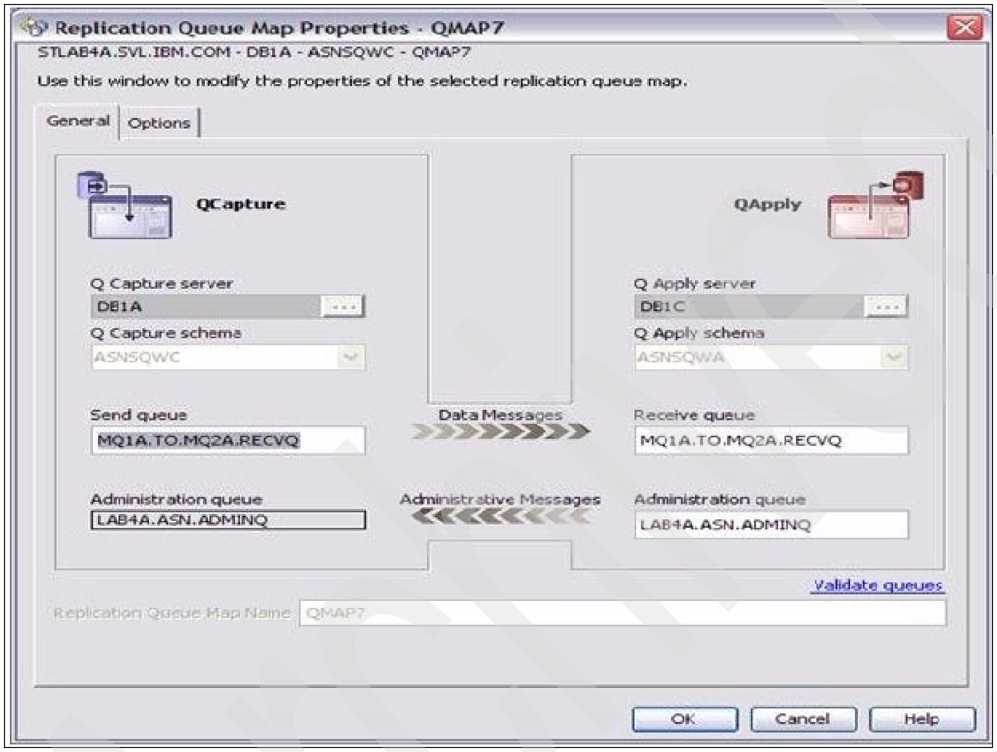
Table 12-1 shows the database and MQ setup on the source and target z systems.

Table 12-1 The database and MQ setup at source and target systems

	Source System	**Target System**
DB2 for System z database	DB1A	DB1C
Queue Manager	MQ1A	MQ2A
Remote send queue used by Q capture	MQ1A.TO.MQ2A.RECVQ	-
Local receive queue used by Q apply	-	MQ1A.TO.MQ2A.RECVQ
Administration queue used by Q capture	LAB4A.ASN.ADMINQ	-

Figure 12-16 shows the queue map properties of our subscription, which also displays the WebSphere MQ queues that are used to transmit the data. For an overview of the WebSphere MQ queues and channels needed to use Q Replication in this configuration, see the following web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.swg.im.iis.repl.qrepl.doc/topics/iiyrqwmqcobjunir.html


Replication Queue Map Properties - QMAP7

STLAB4A.SVL.IBM.COM - DB1A - ASNSQWC - QMAP7

Use this window to modify the properties of the selected replication queue map.

Q Capture server: DB1A

Q Capture schema: ASNSQWC

Send queue: MQ1A.TO.MQ2A.RECVQ

Administration queue: LAB4A.ASN.ADMINQ

Q Apply server: DB1C

Q Apply schema: ASNSQWA

Receive queue: MQ1A.TO.MQ2A.RECVQ

Administration queue: LAB4A.ASN.ADMINQ

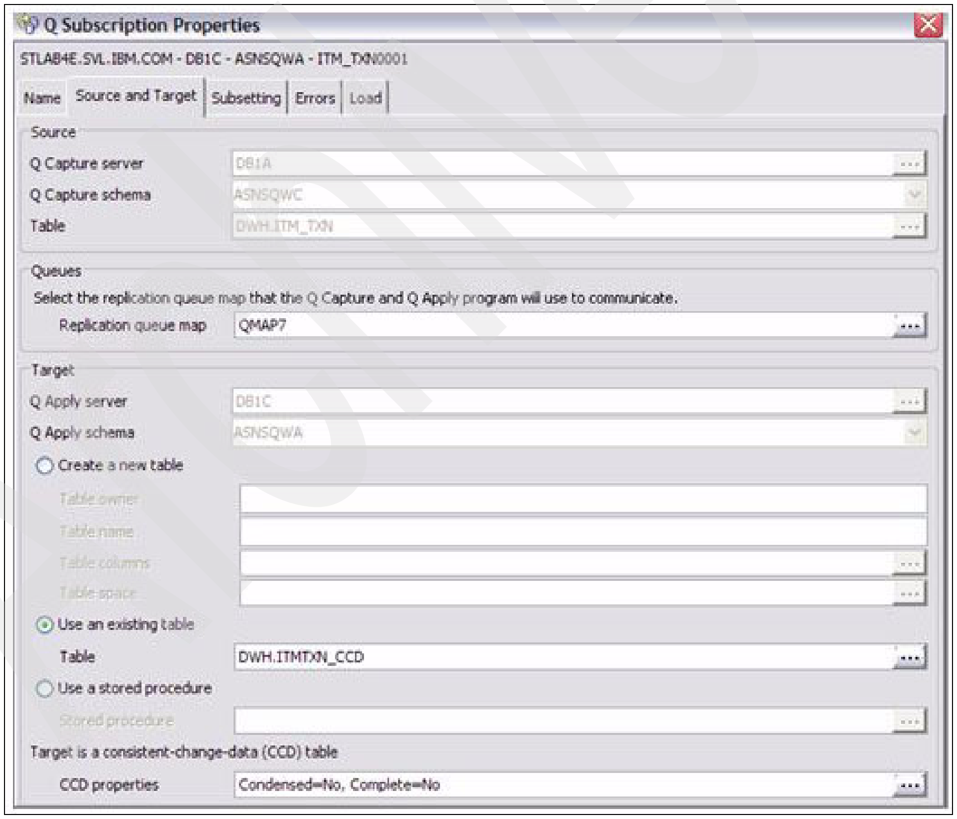
Replication Queue Map Name: QMAP7

Figure 12-16 Replication Q map properties

Figure 12-17 shows the properties of the target table. All the target tables are defined as noncomplete and noncondensed Change Capture Data (CCDs). A CCD is a type of auditing table built in to Infosphere Replication Server. A noncomplete CCD table contains only captured changes to the source table and starts with no data. A noncondensed CCD table contains multiple rows with the same key value, one row for every UPDATE, INSERT, or DELETE operation at the source table. When added to the CCD table, the operations are held intact as insert, update, or delete and the whole row is inserted into the CCD table. These CCD tables are inputs to the data flows.

There is a description of the business rationale for using noncomplete and noncondensed CCD tables in the Information Center topic CCD tables as targets for Q replication at the following web page:

```
http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.
im.iis.repl.qrepl.doc/topics/iiyrqsubcccdtgts.html?resultof=%22%43%43%4
4%22%20%22%63%63%64%22%20%22%74%61%62%6c%65%22%20%22%74%61%62%6c%22%20
```


Q Subscription Properties

STLAB4E.SVL.IBM.COM - DB1C - ASNSQWA - ITM_TXN0001

Source

Q Capture server	DB1A
Q Capture schema	ASNSQWC
Table	DWH.ITM_TXN

Queues

Select the replication queue map that the Q Capture and Q Apply program will use to communicate.

Replication queue map: QMAP7

Target

Q Apply server	DB1C
Q Apply schema	ASNSQWA

Create a new table

Table owner

Table name

Table columns

Table space

[✓ Use an existing table]

Table: DWH.ITMTXN_CCD

Use a stored procedure

Stored procedure

Target is a consistent-change-data (CCD) table

CCD properties: Condensed=No, Complete=No

Figure 12-17 Q subscription Properties window

For our application, we want to keep a history of changes to the source tables, so we want updates and deletes to be recorded as inserts. We need to decide how to deal with updates and deletes. When a source record is updated, do we update the dimension or the fact table? Or do we insert a new record with the updated values, but keep the original record as a history? In this case, we kept the original and insert a new row. Similarly, for deletes, kept the original row but updated the end date to mark it as no longer current.

12.4.2 Implementing and automating the Data and Control Flows

The next step is to define SQW flows using InfoSphere Warehouse Design Studio to extract, prepare, transform and load the delta data. These data and control flows are built using the above architecture and are repeatable. Data flows are graphical flow models that represent, at a higher level the logical operations that are needed to move and transform data. From the flow model, optimized SQL is generated that executes at the designated DB2 for z/OS database.

The following general requirements apply to all flows:

- Updates to the source tables should be reflected in the target table in 30 minutes. Over time, the Service Level Agreement (SLA) decreases this to five minutes.
- The solution should minimize impact on the production tables in terms of availability and response time.
- The solution should minimize impact to the target table in terms of availability for query.
- The solution should allow for independent scheduling of extracting changes in the Sales system and the updating of the fact table.

The source tables for all of these data flows are the CCD tables that the Q apply process writes to by pulling the changes from the WebSphere MQ queue where the Q capture process puts them. These are the records from the OLTP system with additional control information such as an indicator of the type of change, insert, update or delete. The apply process is invoked in another kind of flow called a control flow.

Time dimension data flow

The data flow in Figure 12-18 shows the delta processing for the time dimension table. This flow is based on the previously discussed architecture, but many layers have been collapsed as the flow is fairly simple.

The following are the business requirements for the time dimension delta flow:

- Pick up the delta changes from the measurement CCD table.
- By definition, there will be no delete records coming from the OLTP system.
- Ignore the deletes and process only inserts and updates.
- Pick only transactions that match MSR_PRD_TP_ID = 3200, 3201 or 3202.
- Perform datatype conversion.
- Map columns to match the target table.

We read the change records from the measurement period change table (MSR_PRDCCD in Figure 12-18). We use the WHERE operator to select just the transactions that match the record types specified in the list above (3200, 3201, and 3202). The select list is used to perform datatype conversions and column mappings that are required. The splitter is used to separate the inserts and updates, which are processed appropriately against the TIME dimension table.

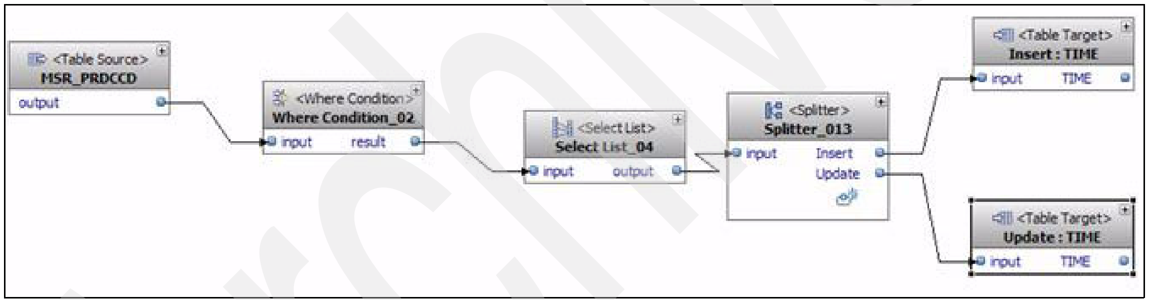


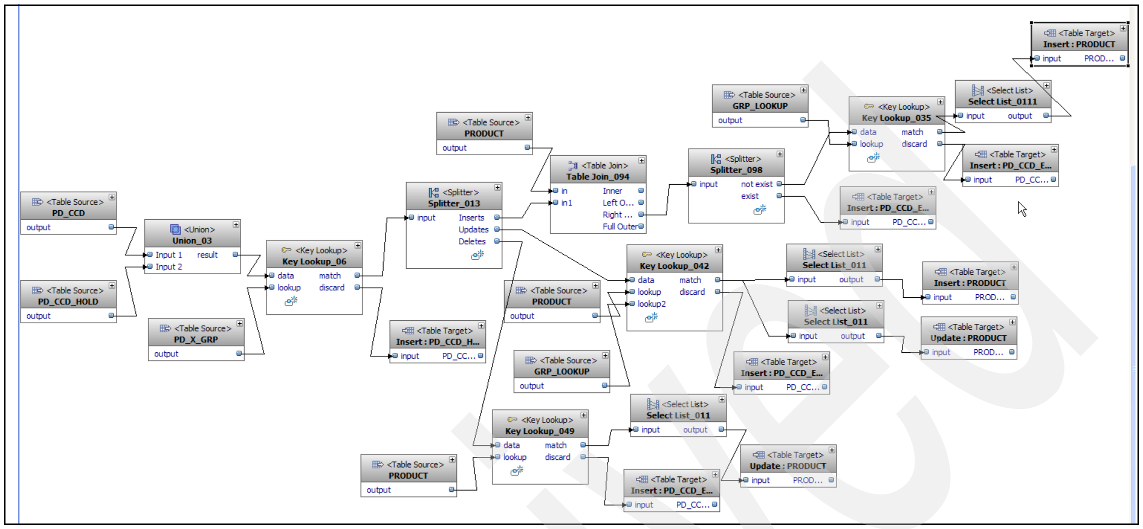
Figure 12-18 Time dimension: Delta data flow

Product Dimension data flow

The following business requirements are for the product dimension:

- Process all inserts, updates and delete change records.
- A product change record must have a matching grouping record in the PD_X_GRP table. If not, hold the record for later processing.
- Maintain the PRODUCT table as a type 2 slowly changing dimension. The type 2 slowly changing dimension tracks historical data by creating multiple records in the dimensional tables with separate keys, creating a continuous history preservation as a new record is inserted each time a change is made.

Figure 12-19 shows the product dimension data flow. As you can see, it is large and complex. As such, you are not expected to read the contents of that data flow, it is provided to give you an overall perspective of the complete data flow.



<Table Source> PD_CCD
<Table Source> PD_CCD_HOLD
<Union> Union_03
<Table Source> PD_X_GRP
<Key Lookup> Key Lookup_06
<Splitter> Splitter_013
<Table Target> Insert : PD_CCD_H...
<Table Source> PRODUCT
<Table Join> Table Join_094
<Table Source> PRODUCT
<Table Source> GRP_LOOKUP
<Key Lookup> Key Lookup_049
<Table Source> PRODUCT
<Key Lookup> Key Lookup_042
<Splitter> Splitter_098
<Table Source> GRP_LOOKUP
<Key Lookup> Key Lookup_035
<Select List> Select List_0111
<Table Target> Insert : PRODUCT
<Table Target> Insert : PD_CCD_E...
<Table Target> Insert : PD_CCD_E...
<Select List> Select List_011
<Select List> Select List_011
<Table Target> Insert : PRODUCT
<Table Target> Update : PRODUCT
<Table Target> Insert : PD_CCD_E...
<Select List> Select List_011
<Table Target> Update : PRODUCT
<Table Target> Insert : PD_CCD_E...

Figure 12-19 Product dimension: Delta data flow

However, for the purposes of this example you can want to follow the steps in the flow. To help you do that, we have segmented the flow into four logical sub-processes:

1. Preliminary processing

   Read the changes from the product CCD tables and split the processing into separate streams for inserts, updates and deletes.

2. Process inserts

   Add a new current product row being inserted into the PRODUCT table.

3. Process updates

   Update the current product row by updating the effective timestamp and insert a new current product row.

4. Process deletes

   Update the current product row by updating the effective timestamp.

Figure 12-20 shows how the delta changes are picked up from the PD_CCD table. We hold the rows that do not have a match in the group hierarchy table, PD_X_GRP in another table which we UNION with the PD_CCD table at the next run. Finally, we split the change records based on the type of change, insert, update, or delete as they have different processing requirements.

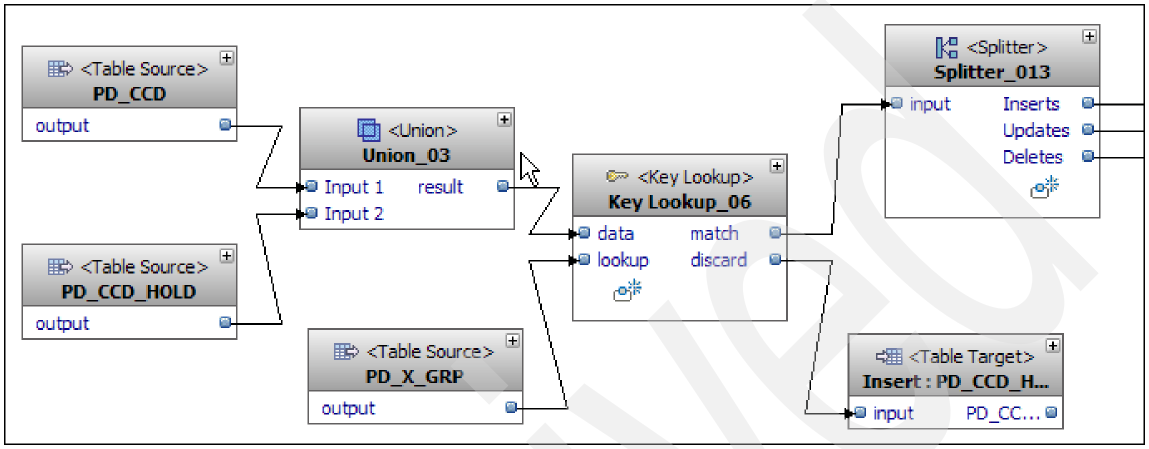


Figure 12-20 Product dimension: Preliminary processing

Figure 12-21 and Figure 12-22 show the insert portion of the product data flow, which has been segmented into two figures to make it more legible. Because this is an insert, we expect that the row does not already exist in the target. We use a right outer join and a splitter to identify if an incoming change record has a corresponding PRODUCT row. If the change record has an entry in the PRODUCT table, the row is written to the error table. If it does not, the product hierarchy fields are obtained from the GRP_LOOKUP table, perform column mappings and datatype conversions and finally insert a new record to the target table.

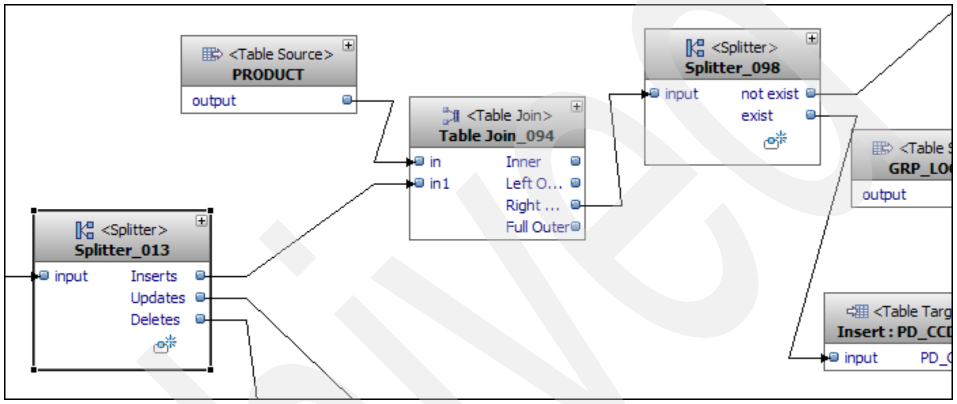


Figure 12-21 Product dimension: Insert delta data flow: Part 1

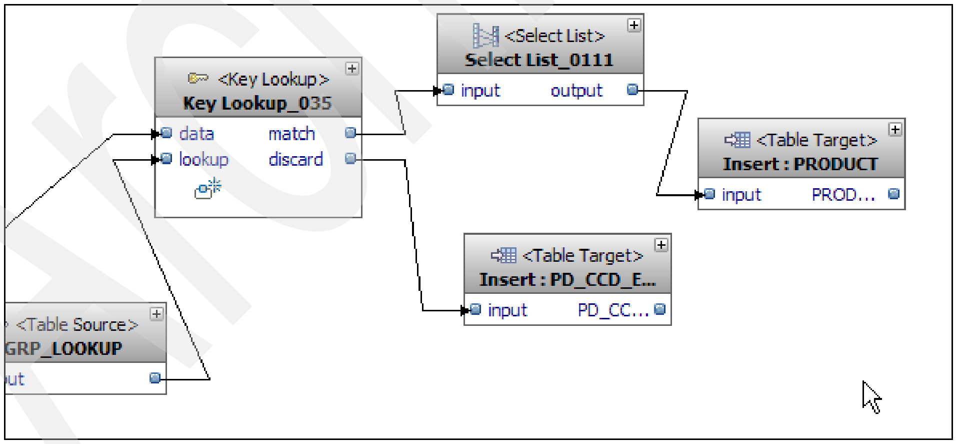


Figure 12-22 Product dimension: Insert delta data flow: Part 2

Figure 12-23 shows the update part of the product dimension data flow. In this case, a current record must already exist in the PRODUCT table as well as a corresponding entry in the GRP_LOOKUP table. New current product rows are inserted into the PRODUCT table with the END TIMESTAMP of NULL to show that the data in this row is current. The current PRODUCT record is updated to set the END TIMESTAMP to the current timestamp to mark this row as obsolete. If the row does not exist, we write the update to an error table.

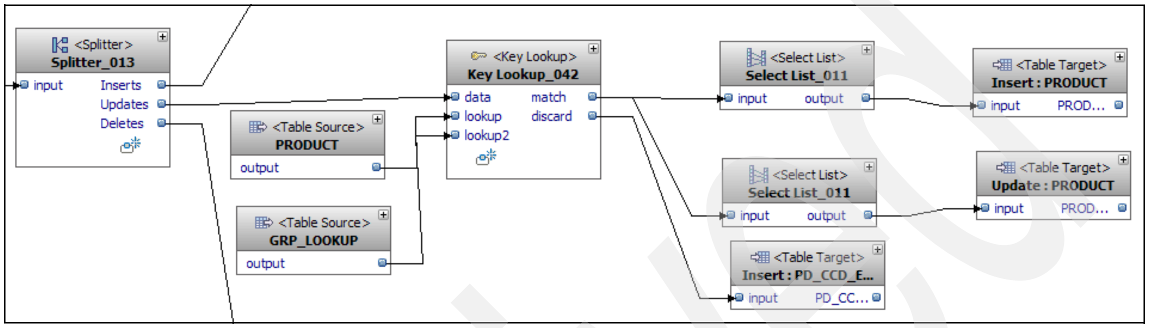


Figure 12-23 Product dimension: Update delta data flow

Figure 12-24 shows the delete part of the product dimension data flow. To process deletes a current product row must exist. A logical delete is performed by updating the END TIMESTAMP in the row to the current timestamp to mark the row obsolete. If a current product row does not exist in the PRODUCT table, we write to the error table.

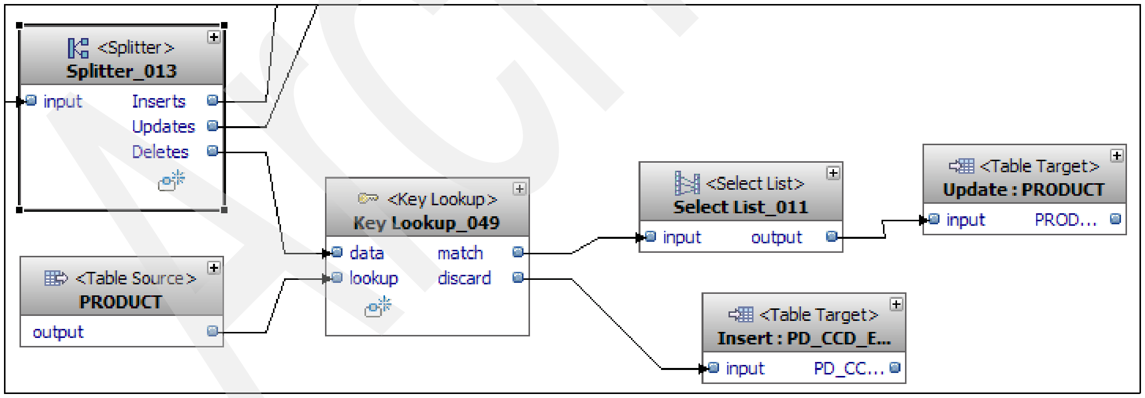


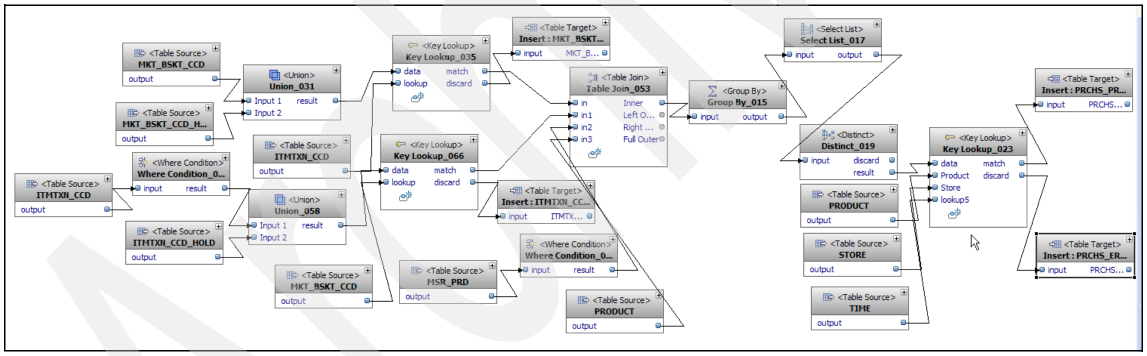
Figure 12-24 Product dimension: Delete delta data flow

Fact Table data flow

The following business requirements are for the fact table:

- The rows in the target table, PRCHS_PRFL_ANLYSIS are sourced from two different tables, ITM_TXN and MKT_BKST_TXN and must be processed together:
  - There must be one market basket record in the MKT_BSKT_TXN
  - The item table, ITM_TXN, must have at least one row for each market basket in the MKT_BSKT_TXN table identified by the MKT_BSKT_TXN_ID column
  - The number of items in ITM_TXN for a market basket, MKT_BKST_TXN, must equal the value in MKT_BKST_TXN.NBR_ITM.
- Only inserts are processed from the CCD table.
- Only ITM_TXN record type 69 (ITM_TXN_TP_ID) will be processed.
- Only Products with product type 3200 (MSR_PRD_TP_ID) will be processed.

Figure 12-25 shows the complete data flow that we used to capture deltas for the fact tables. This is the complete flow with numerous nodes, and thus is not legible even in this larger figure. We included it here to give you the full perspective.



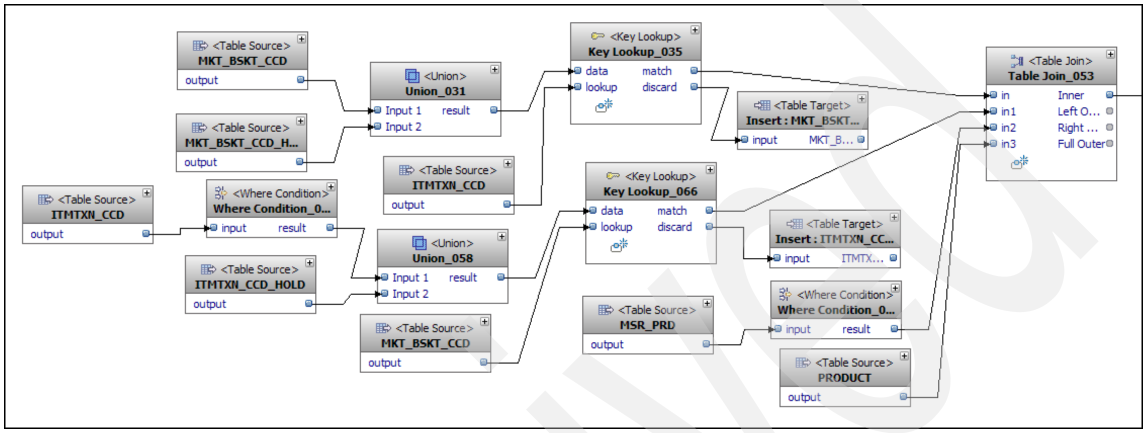
<Table Source> MKT_BSKT_CCD output
<Table Source> MKT_BSKT_CCD_H... output
<Table Source> ITMTXN_CCD output
<Where Condition> Where Condition_0... input result
<Table Source> ITMTXN_CCD_HOLD output
<Union> Union_031 Input 1 Input 2 result
<Table Source> ITMTXN_CCD output
<Union> Union_058 Input 1 Input 2 result
<Table Source> MKT_BSKT_CCD output
<Key Lookup> Key Lookup_035 data lookup match discard
<Key Lookup> Key Lookup_066 data lookup match discard
<Table Source> MSR_PRD output
<Table Target> Insert : MKT_BSKT... input MKT_B...
<Table Target> Insert : ITMTXN_CC... input ITMTX...
<Where Condition> Where Condition_0... input result
<Table Source> PRODUCT output
<Table Join> Table Join_053 in in1 in2 in3 Inner Left O... Right ... Full Outer
<Group By> Group By_015 input output
<Select List> Select List_017 input output
<Distinct> Distinct_019 input discard result
<Table Source> PRODUCT output
<Table Source> STORE output
<Table Source> TIME output
<Key Lookup> Key Lookup_023 data Product Store lookup5 match discard
<Table Target> Insert : PRCHS_PR... input PRCHS...
<Table Target> Insert : PRCHS_ER... input PRCHS...

Figure 12-25 The Fact table delta data flow

To make Figure 12-25 more legible, we present it in two parts:

- Figure 12-26 on page 511
- Figure 12-27 on page 511

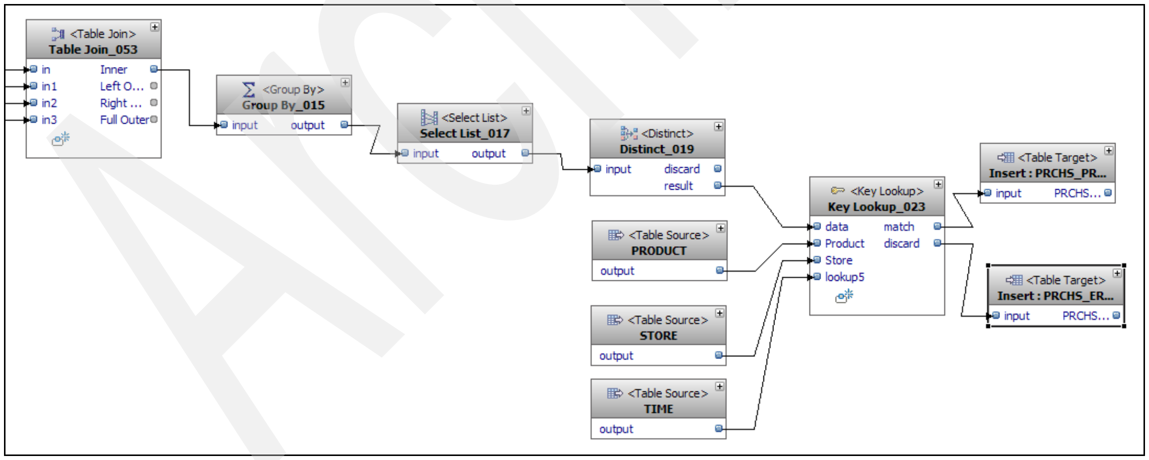
Figure 12-26 shows the first part of the data flow. We pick up the delta changes from the ITMTXN_CCD and MKT_BSKT_CCD tables. Because we want to process the transactions together, any non-matching rows are placed in the hold tables and picked up at the next run. In this part of the flow, we also filter out the rows that we need, ITM_TXN record type 69 and products with product type 3200. We join these tables with the master period and product tables.



<Table Source> MKT_BSKT_CCD output
<Table Source> MKT_BSKT_CCD_H... output
<Table Source> ITMTXN_CCD output
<Where Condition> Where Condition_0... input result
<Table Source> ITMTXN_CCD_HOLD output
<Union> Union_031 Input 1 Input 2 result
<Table Source> ITMTXN_CCD output
<Union> Union_058 Input 1 Input 2 result
<Table Source> MKT_BSKT_CCD output
<Key Lookup> Key Lookup_035 data lookup match discard
<Key Lookup> Key Lookup_066 data lookup match discard
<Table Source> MSR_PRD output
<Table Target> Insert : MKT_BSKT... input MKT_B...
<Table Target> Insert : ITMTXN_CC... input ITMTX...
<Where Condition> Where Condition_0... input result
<Table Source> PRODUCT output
<Table Join> Table Join_053 in in1 in2 in3 Inner Left O... Right ... Full Outer

Figure 12-26 First part of the fact table delta data flow

Figure 12-27 shows the second part of the fact delta flow. In this part of the flow, ensure that the rows exist in all dimensions before inserting into the fact table.



<Table Join> Table Join_053 in in1 in2 in3 Inner Left O... Right ... Full Outer
<Group By> Group By_015 input output
<Select List> Select List_017 input output
<Distinct> Distinct_019 input discard result
<Table Source> PRODUCT output
<Table Source> STORE output
<Table Source> TIME output
<Key Lookup> Key Lookup_023 data Product Store lookup5 match discard
<Table Target> Insert : PRCHS_PR... input PRCHS...
<Table Target> Insert : PRCHS_ER... input PRCHS...

Figure 12-27 Second part of the fact table delta data flow

The control flow

While building the data flows shown on the previous page, we test them by executing them as we build them. In practice, we might want to schedule data flows to run concurrently while others can start only when previous loads have finished. We can also add processing rules to ensure that the success or failure of an activity or process is handled appropriately.

A control flow consists of a set of operators that defines this logic, including the ability to specify activities that can be run in parallel. In our control flow, we run a JCL script to start Q Apply first so that the CCD tables are populated with the changes from the source tables. We run Q Apply with the AUTOSTOP parameter so that changes from the last time Q Apply was run till the current time are applied and Q Apply stops. The AUTOSTOP Q Apply parameter is a feature that allows us to control the scheduling of data refresh so that we can schedule the control flow to run when we need it to.

We run the delta data flows in parallel to propagate the changes to the warehouse dimension tables. We could, conceivably, run these data flows in series (for example, if we wanted the time deltas every day, but the product delta's every week). Because the fact tables cannot be loaded till the dimension tables are loaded, run the fact delta flow if the data flows populating the dimension tables are successful. And if the fact data flow is successful, run an SQL script to delete from the CCD tables. If any of the operations fail, an email is sent using the email operator. Figure 12-28 shows the control flow.

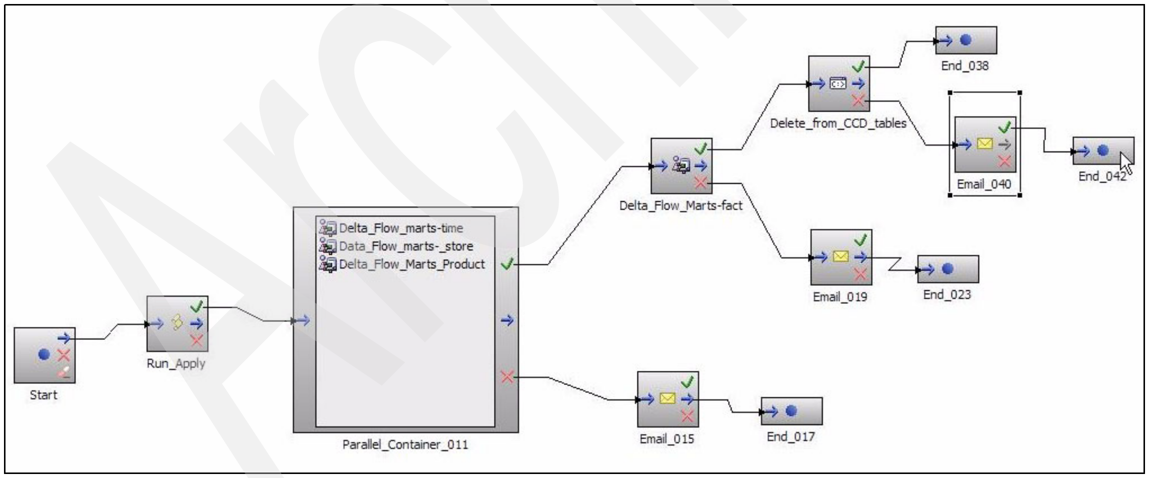
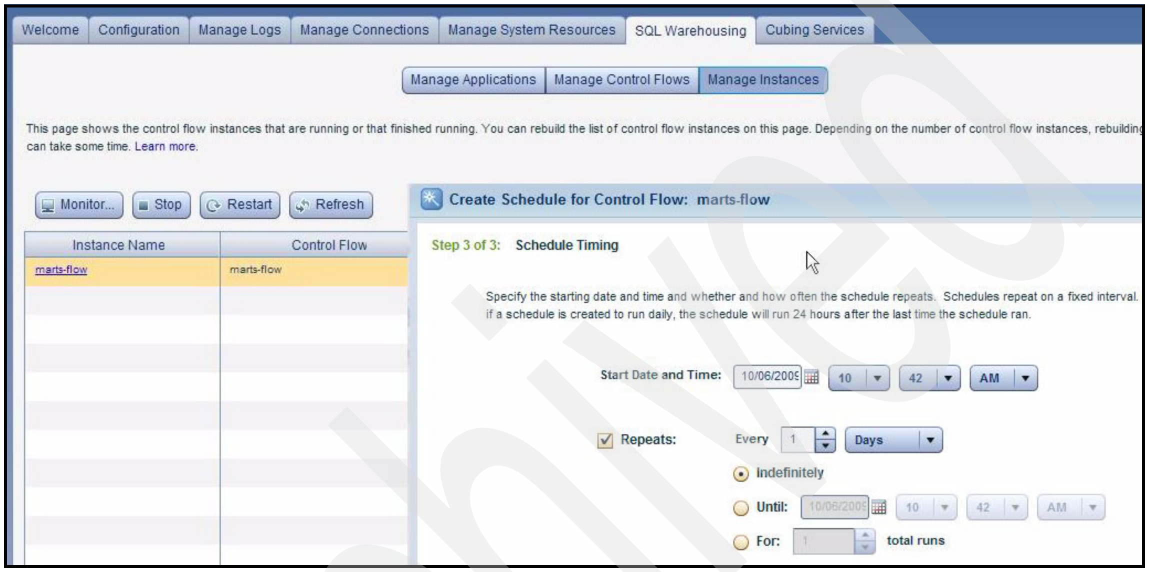


Figure 12-28 Control flow using Q Replication to feed the warehouse

This control flow can be deployed to the administration console and scheduled to run at specific intervals. Depending on the business requirements for the currency of data, the control flow can be scheduled to run every five minutes, every hour, or even daily or weekly. The control flow can also be monitored and the status of the run displayed. Log files can also be viewed using the administration console. Figure 12-29 shows the administration console where the control flow can be scheduled and monitored.



This page shows the control flow instances that are running or that finished running. You can rebuild the list of control flow instances on this page. Depending on the number of control flow instances, rebuilding can take some time. Learn more.

Instance Name	Control Flow
marts-flow	marts-flow

Create Schedule for Control Flow: marts-flow

Step 3 of 3: Schedule Timing

Specify the starting date and time and whether and how often the schedule repeats. Schedules repeat on a fixed interval. if a schedule is created to run daily, the schedule will run 24 hours after the last time the schedule ran.

Start Date and Time: 10/06/2009 10 42 AM

[✓] Repeats: Every 1 Days

[✓ Indefinitely]

Until: 10/06/2009 10 42 AM

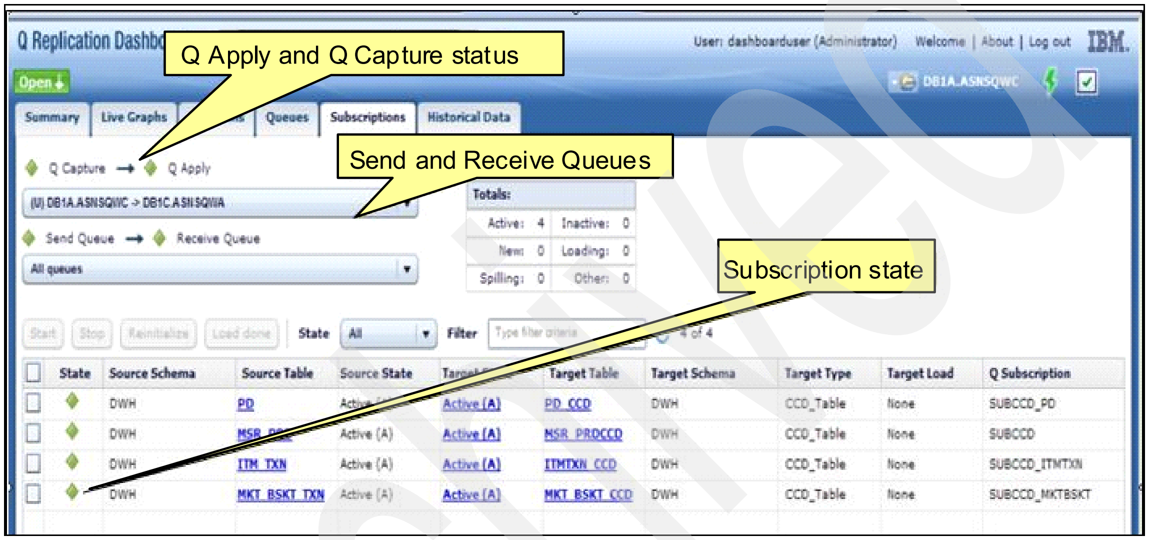
For: 1 total runs

Figure 12-29 InfoSphere administration console

Monitoring the InfoSphere Replication Process

The Q Replication Dashboard provides health and performance monitoring for InfoSphere Q Replication. The dashboard summarizes information to help you identify and troubleshoot problems. It allows drill down into details in addition to at-a-glance status of programs, queues, Q subscriptions, and other objects.

We used the Q Replication Dashboard to view our subscriptions and to see Q Apply and Q Capture's health. Figure 12-30 shows the dashboard with the subscriptions and the queues we are using.



Totals:	
Active: 4	Inactive: 0
New: 0	Loading: 0
Spilling: 0	Other: 0

	State	Source Schema	Source Table	Source State	Target State	Target Table	Target Schema	Target Type	Target Load	Q Subscription
☐	◆	DWH	PD	Active (A)	Active (A)	PD_CCD	DWH	CCD_Table	None	SUBCCD_PD
☐	◆	DWH	MSR_PRD	Active (A)	Active (A)	MSR_PRDCCD	DWH	CCD_Table	None	SUBCCD
☐	◆	DWH	ITM_TXN	Active (A)	Active (A)	ITMTXN_CCD	DWH	CCD_Table	None	SUBCCD_ITMTXN
☐	◆	DWH	MKT_BSKT_TXN	Active (A)	Active (A)	MKT_BSKT_CCD	DWH	CCD_Table	None	SUBCCD_MKTBSKT

Figure 12-30 The Q Replication Dashboard showing the subscriptions

Figure 12-31 shows the Q Capture and Q Apply activity.

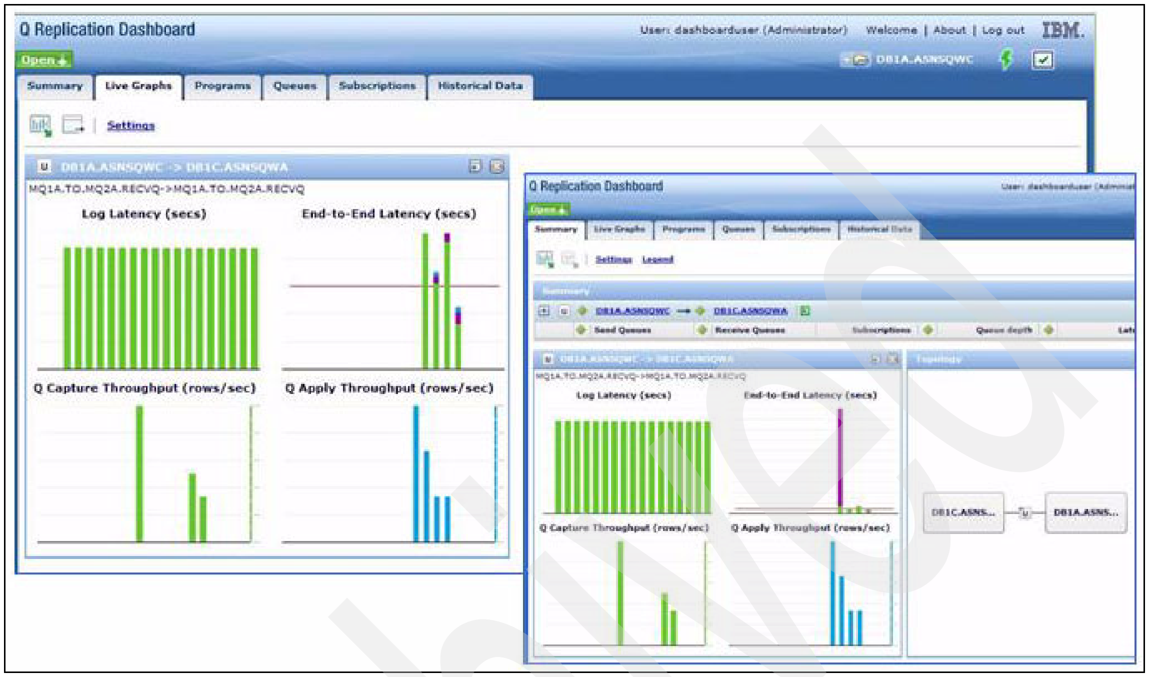

Q Replication Dashboard
User: dashboarduser (Administrator) Welcome | About | Log out
Open
DB1A.ASNSQWC
Summary
Live Graphs
Programs
Queues
Subscriptions
Historical Data
Settings
DB1A.ASNSQWC -> DB1C.ASNSQWA
MQ1A.TO.MQ2A.RECVQ->MQ1A.TO.MQ2A.RECVQ
Log Latency (secs)
End-to-End Latency (secs)
Q Capture Throughput (rows/sec)
Q Apply Throughput (rows/sec)
Legend
Send Queues
Receive Queues
Queue depth
Topology
DB1C.ASNS...
DB1A.ASNS...

Figure 12-31 The Q Replication Dashboard showing Q capture and apply activity

12.5 Summary

In this chapter we have provided an architecture and high level design for a population subsystem. We used this architecture to create data flows and control flows to propagate delta changes to the warehouse tables. To do this, we used InfoSphere Q Replication to get near-real time updates and SQW to create the data and control flows.

A

DB2 ESE for LUW: Support for data warehousing

InfoSphere Warehouse is built on DB2 Enterprise Server Edition (ESE). In this appendix we describe key data warehouse features in DB2 ESE for Linux, UNIX and Microsoft Windows (LUW) that should be taken into consideration as part of any large-scale data warehouse implementation. New capabilities for data warehousing introduced in DB2 ESE for LUW V9.7 are also included.

Introduction

DB2 Enterprise Server Edition (ESE) is a full-function premier web-enabled client/server RDBMS. It is available on UNIX operating environments (including AIX, Solaris, and HP-UX), Linux, and Windows.

DB2 ESE is positioned for large and mid-sized departmental servers. It has the ability to partition data in a single server or across multiple servers with the database partitioning feature (DPF).

Note: InfoSphere Warehouse Enterprise Edition is now required to exploit the capabilities of the DPF.

Architecture

DB2 has a shared-nothing architecture, as depicted in Figure 12-32, as opposed to a shared-disk or shared-memory architecture, which is used by other commercial database products. The advantage of the shared-nothing architecture is the ability to spread all the data across multiple servers in a near linear fashion. This architecture is a proven way to spread a query workload evenly over all available system hardware (disks, processors, switches, and memory).

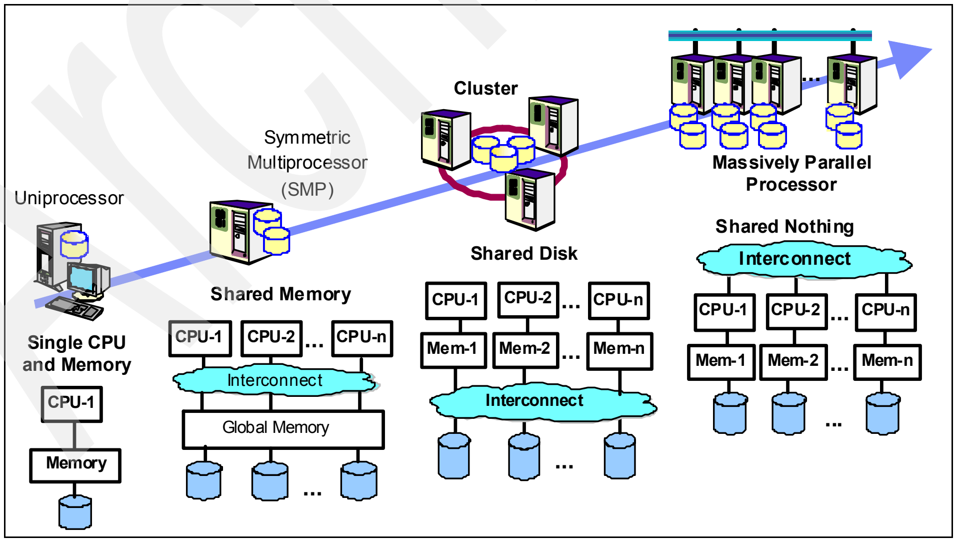


Figure 12-32 Software and hardware architectures

A shared nothing architecture supports environments in the following fashion:

- Uniprocessor

  A single processor accessing individual memory and disk storage
- Symmetric multiprocessor (SMP)

  Multiple processors accessing the same memory and all the available disk storage
- Cluster

  Distributed memory processors capable of accessing all the available disk storage
- Massively parallel processor (MPP)

  Multiple loosely coupled processors linked by a high-speed interconnect

DB2 runs on a variety of hardware topologies including a large number of clustered small or large SMP servers. For DB2 to run in a shared-nothing architecture, a high-speed interconnect between servers is required. The interconnect connects the various servers in the cluster such that a coordinated query can be effectively executed over a shared-nothing architecture. Switches connecting the cluster are also diverse, and DB2 has supported the Intel® Virtual Interface Architecture and the Sun Cluster Interface since 1998. NUMA-Q® Fabric support was added in 1999, and Logic Infiniband in 2001. The DB2 shared-nothing implementation on UNIX and Windows has been proven to scale to over 50 servers on audited TPC-H, TPC-C, and TPC-D benchmarks, and over two hundred servers in client environments.

Clustering for high availability is a task that can be done by the operating system for any data service, such as a database. On the other end of the spectrum, clustering for scalability requires intelligence in the database engine.

Choosing shared-nothing for scaling on UNIX systems, or Intel SMPs clustered with Windows or Linux, offers two advantages:

- Linear scaling

  Shared-nothing is modular and adaptive. As the system grows, it is easy to add more hardware to the cluster. Data redistribution can be done incrementally if it has to fit in any maintenance windows, or as a background task.
- High availability (HA)

  HA is provided by using redundant machines, even commodity hardware, that can assume the workload of a machine that realizes a hardware failure. Workload ownership (including an IP address) and data ownership can move quickly from one machine to another or from one machine to many to ensure workload balancing during a hardware failure. Ownership of the data does not have to be arbitrated by a separate software layer, because ownership is associated with the IP address of the node.

InfoSphere Balanced Warehouse

IBM has years of experience building large data warehousing infrastructures which can be a complex combination of server and storage hardware together with all the necessary systems software. It is important that the infrastructure is stable and manageable and provides predictable performance.

To meet these goals the InfoSphere Balanced Warehouse was developed using industry-standard components for the following advantages:

- Ease of installation and implementation.
- Balanced performance for delivering scalability.
- Fault tolerance for high availability.

The end result is a solution that provides faster time to market, lower total cost of ownership, and high reliability. The InfoSphere Balanced Warehouse is an appliance-like (complete and packaged) solution that uses IBM commodity components and is delivered as a database-ready solution.

The InfoSphere Balanced Warehouse is a complete solution for building a data warehousing infrastructure using a modular, building-block design with a focus on balanced performance throughout the hardware and software configuration. It consists of a combination of processors, memory, I/O, storage, DB2 database partitions, and DB2 configuration parameters under a single operating system representing a scalable ratio of disk I/O to memory to CPU to network.

InfoSphere Balanced Warehouse offerings range from the C-Class based on the Linux operating system and Intel-based servers intended for small and medium sized organizations up to the highly scalable E-Class intended for large scale enterprise warehouses and based on IBM POWER6® processors running the AIX operating system.

The key concept of the InfoSphere Balanced warehouse is the modular design consisting of Balanced Configuration Units (BCUs) as shown in Figure A-1.

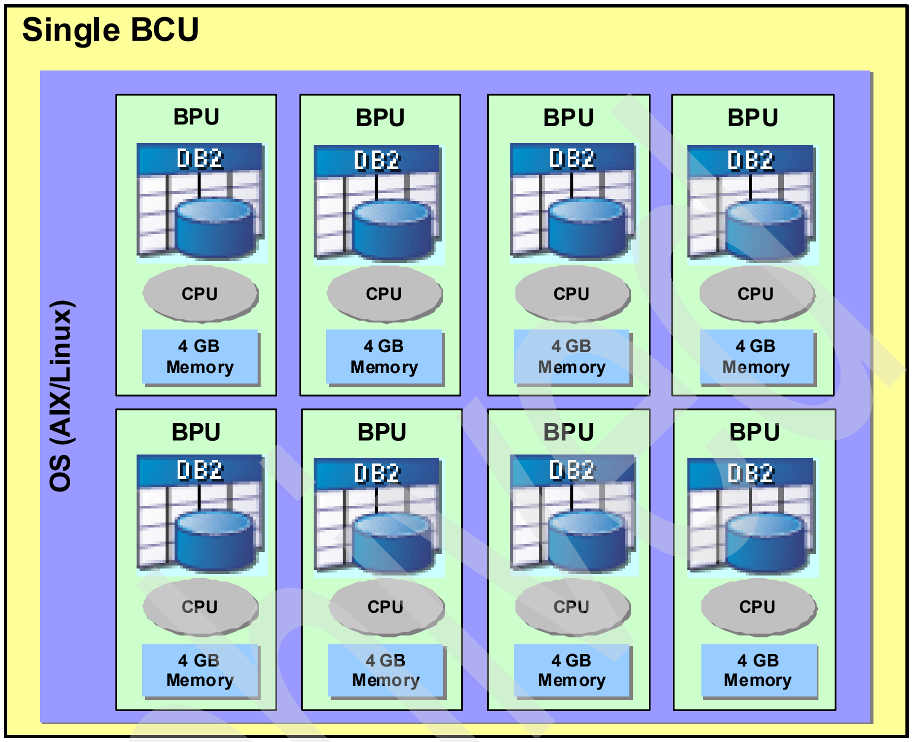


Figure A-1 Single BCU

The building blocks start with a virtual entity, the Balanced Partition Unit (BPU). The BPU is one DB2 partition and its related components. Components see the processes and hardware resources that service each BPU (which consists of a DB2 partition, CPU, memory, and disk).

Balanced collaboration is the key to success, whereby all of the components are chosen to fit with each other for a mixture of compatibility, performance, and reliability reasons. The idea of the BCU is to choose a combination of processors, memory, I/O storage, DB2 partitions, and a single operating system, which make up a scalable building block.

Larger systems are built by combining numerous BCU building blocks, as shown in Figure A-2.

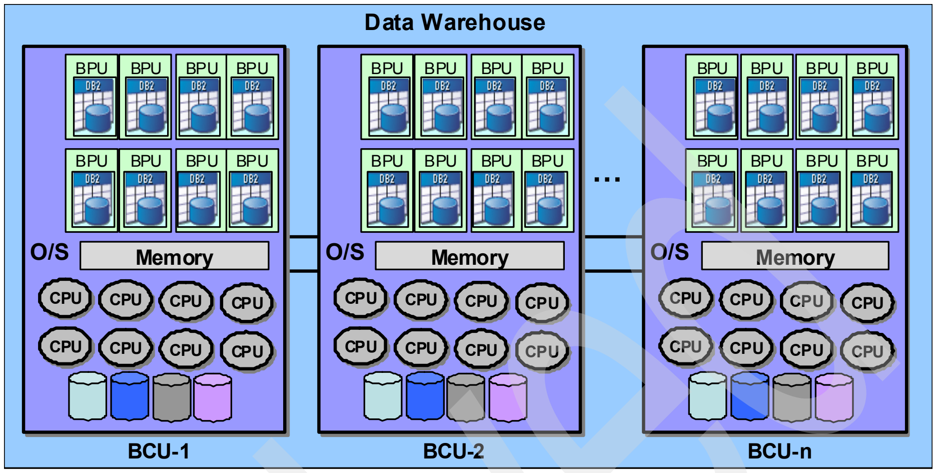


Figure A-2 BCU building blocks

A large data warehouse uses the shared-nothing DB2 architecture and consists of a number of virtually identical scalable BCU building blocks. These blocks are combined into a single DB2 system image configuration to support the required data and query volumes.

The BCU is composed of specific software levels and hardware that IBM has integrated and tested as a pre-configured building block for data warehousing systems. The BCU reduces the complexity, cost, and risk of designing, implementing, growing, and maintaining a data warehouse and business intelligence (BI) infrastructure.

Figure A-3 on page 523 shows a complete E-Class InfoSphere Balanced Warehouse example including the following features:

- BCUs for storing the data.
- Administrative BCUs for storing the DB2 catalog and user connectivity
- InfoSphere Warehouse application servers
- Extract, Transform and Load (ETL) servers for processing the input data
- Tivoli® Storage Manager (TSM) for controlling backups
- Remote Support Manager (RSM) to assist with storage server problem resolution
- DS4000® series storage servers

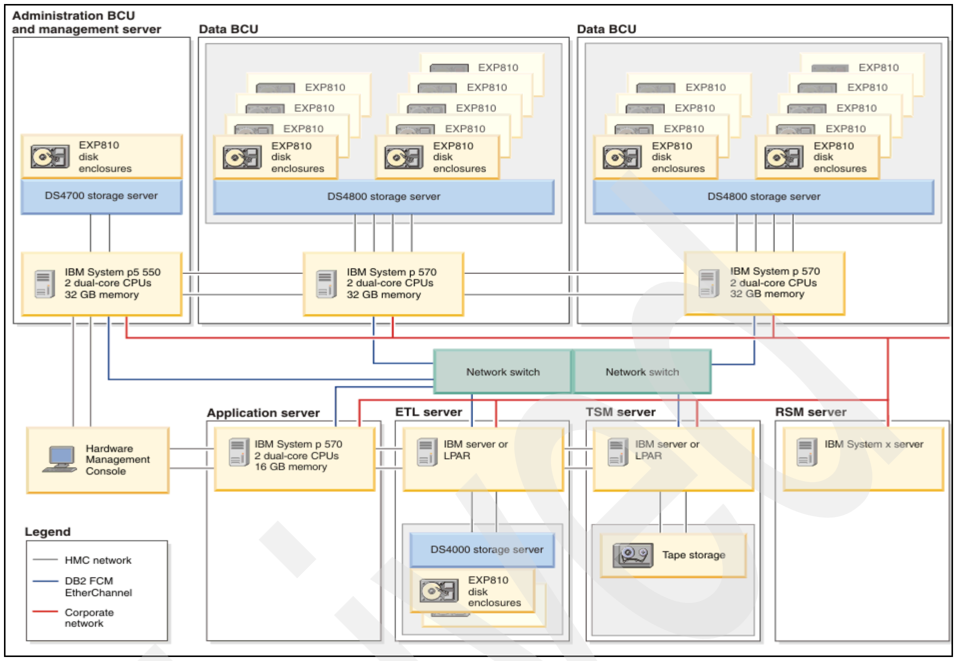


Figure A-3 Typical E-Class InfoSphere Balanced Warehouse

Attention: More information about the InfoSphere Balanced Warehouse solution and methodology can be obtained by engaging the IBM BI Best Practices team. Contact the team through your local IBM representative for current recommendations.

Partitioning

Partitioning a database is often done to provide increased scalability, improve query performance, or ease database maintenance. Partitioning typically involves logically or physically dividing up a table into smaller units to increase database, CPU, and I/O parallelism. DB2 offers several methods to partition a database. One or all methods can be used for data warehousing. In all cases, the partitioning scheme is handled through the data definition language (DDL).

Database partitioning

DB2 UDB with DPF uses database or hash partitioning to support large databases and exploit large hardware environments efficiently. *Database partitioning* refers to splitting a database into separate physical partitions, which can be distributed across or in SMP and clustered servers, while still appearing to the user and database administrator as a single-image database.

DB2 UDB with DPF divides the database into partitions. By searching these database partitions in parallel, elapsed times for queries can be dramatically reduced. Data is distributed across multiple database partitions, thereby enabling database scaling into the terabytes. A crucial factor in attaining the performance advantages of a parallel environment is being able to distribute intelligently the data across the various database partitions. DB2 UDB with DPF supports intelligent database partitioning in that it evenly distributes and remembers the location of each data row.

SQL operations are performed in parallel across all partitions and therefore operate much faster. DB2 UDB with DPF provides parallel everything such that all transactions (selects, updates, deletes, inserts), as well as utility operations, all operate in parallel across all partitions.

The architecture design divides and executes the workload to enhance scalability through enabling partitioned workloads to have their own private data and computing resources. In this way, near-linear scalability is achieved as resources across data partitions are not shared, and there is no central locking of resources.

A database partition is a part of a database that consists of its own data, indexes, configuration files, and transaction logs. A database partition is sometimes called a node or a database node. A single-partition database is a database having only one database partition, and all data in the database is stored in that partition. In this particular case, database partition groups, while present, provide no additional capability.

A partitioned database is a database with two or more database partitions. Tables can be located in one or more database partitions. When a table is in a database partition group consisting of multiple partitions, rows are stored in one partition, and other rows are stored in other partitions. Usually, a single database partition exists on each physical node, and the processors on each system are used by the database manager at each database partition to manage its part of the total data in the database.

Because data is divided across database partitions, you can use the power of multiple processors on multiple physical nodes to satisfy requests for information. Data retrieval and update requests are decomposed automatically into sub-requests, and executed in parallel among the applicable database partitions. The fact that databases are split across database partitions is transparent to users issuing SQL statements.

User interaction occurs through one database partition, known as the coordinator node for that user. The coordinator runs on database partition to which the application is connected. Any database partition can be used as a coordinator node.

Partition groups

A partition group is a logical layer that allows the grouping of one or more partitions to perform operations on all the partitions in the group. A database partition can belong to more than one partition group. When a database is created, DB2 creates three default partition groups that cannot be dropped:

- IBMDEFAULTGROUP

  This is the default partition group for any table you create. It consists of all database partitions as defined in the `db2nodes.cfg` file. This partition group cannot be modified. Tablespace USERSPACE1 is created in this partition group.

- IBMTEMPGROUP

  This partition group is used by all system temporary tables. It also consists of all database partitions as defined in the db2nodes.cfg file. Tablespace TEMPSPACE1 is created in this partition.

- IBMCATGROUP

  This partition group contains the catalog tables (tablespace SYSCATSPACE), and thus it only includes the database catalog partition. This partition group cannot be modified.

To create new database partition groups you can use the create database partition group statement. This statement creates the database partition group in the database, assigns database partitions that you specified to the partition group, and records the partition group definition in the database system catalog tables.

Example A-1 shows the creation of a pgrpall partition group on all partitions specified in the db2nodes.cfg file.

Example A-1 Partition group pgrpall

```
create database partition group pgrpall on all dbpartitionnums
```

To create a database partition group pg0123 consisting only of partitions 0, 1, 2 and 3, issue the command in Example A-2.

Example A-2 Partition group pg0123

```
create database partition group pg0123 on dbpartitionnums (0,1,2,3)
```

A typical scenario for a data warehouse is to separate the catalog, small code tables and all other tables into separate partition groups. The partition groups for the catalog and the code tables each consist of a single partition. The remaining data tables are then stored in a partition group covering all remaining partitions.

Tablespaces in a DPF environment

A tablespace can be created in specific partitions by associating it with a partition group. The CREATE TABLESPACE statement with the IN DATABASE PARTITION GROUP clause can be used for this purpose. This allows users to have flexibility as to which partitions actually store their tables. Example A-3 shows a tablespace mytbls, which spans partitions 0, 1, 2, and 3.

Example A-3 Tablespaces

```
create regular tablespace mytbls in database partition group pg0123
   managed by automatic storage
```

Partitioning keys

A partitioning key is a column (or group of columns) used to determine the partition in which a particular row of data is stored. A partitioning key is defined on a table using the CREATE TABLE statement, as depicted in Example A-4.

Example A-4 Partitioning key

```
create table mytable (col1 int, col2 int, col3 int, col4 char(10))
            in mytbls1
            partitioning key (col1, col2, col3)
```

Partition maps

When a database partition group is created or modified, a partitioning map is automatically created by DB2. A partitioning map, in conjunction with a partitioning key and a hashing algorithm, is used by DB2 to determine which database partition in the database partition group stores a given row of data.

Where a new row is inserted is determined by hashing the partitioning key value of the rows to an entry in the partitioning map, which contains the partition number to use, as shown in Figure A-4.

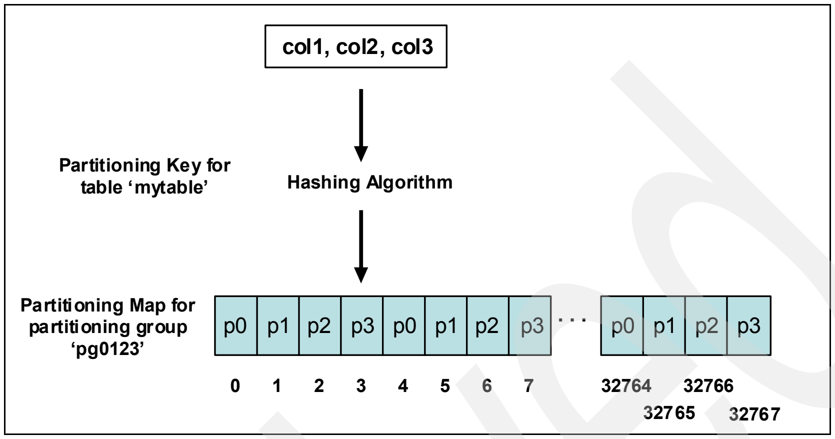

col1, col2, col3
Partitioning Key for table 'mytable'
Hashing Algorithm
Partitioning Map for partitioning group 'pg0123'
p0 p1 p2 p3 p0 p1 p2 p3 ... p0 p1 p2 p3
0 1 2 3 4 5 6 7
32764 32765 32766 32767

Figure A-4 Hash partitioning map

Partition group pg0123 has been defined on partitions 0, 1, 2, and 3. An associated partitioning map is automatically created. This is an array with 32768 entries containing the values 0, 1, 2, 3, 0, 1, 2, 3.... (Partition numbers are stored in round-robin fashion by default, although this can be changed.) Table mytable has been created with a partitioning key consisting of columns col1, col2, and col3. For each row, the partitioning key column values are passed to the hashing algorithm, which returns an output number from 0 to 32767. This number corresponds to one of the entries in the partitioning map array that contains the value of the partition number where the row is to be stored. If the hashing algorithm had returned an output value of 7, the row would have been stored in partition p3.

The partitioning map is an internally generated array containing either 32768 entries for multiple-partition database partition groups, or a single entry for single-partition database partition groups. The partition numbers of the database partition group are specified in a round-robin fashion.

Prior to DB2 v9.7 the number of entries in the partitioning map was limited to 4096 entries. New databases or databases migrated from an earlier release automatically take the larger partitioning map if the DB2_PMAP_COMPATIBILITY registry variable is set to OFF.

Although it is possible to change the partitioning map and correct data skew or make allowances for clustered servers of different CPU performance in practice a large data warehouse should be based on balanced configuration principles and have large number of tables partitioned across many identical balanced partition units. It is likely to be impractical to correct for data skew for a subset of tables and likely increase skew for other tables with different partitioning keys in the same partition group.

Choosing partitioning keys

Given a particular DPF workload and configuration, one of the bigger performance factors is the choice of partitioning keys for the tables. There might not, however, be one absolute and obvious best set of them, as one set might optimize queries and another set optimize other queries, so you must look at the overall workload and know what is most important to optimize. Try various alternatives before getting the best results.

The following lists detail a few quick tips for partitioning key selection. For more assistance see IBM Redbooks publication *Up and Running with DB2 UDB ESE: Partitioning for Performance in an e-Business Intelligence World*, SG24-6917.

Do:

- Include frequently used join columns.
- Spread data evenly across partitions.
- Have a broad range domain.
- Use integer, which is more efficient than character and is more efficient than decimal.
- Use DB2 Design Advisor.

Do not:

- Use LOB or LONG fields.
- Have a unique index or primary key unless it is a super-set of the PK.
- Allow ALTERations of the PK.
- Update columns.

Important: DPF partitioning is based on hashing, with the intent to spread rows evenly over partitions and the available hardware resources to maximize performance. It is not wise to choose partitioning keys in an attempt to implement range partitioning, which should be accomplished using table partitioning (TP).

The following list details other considerations involved in choosing partitioning keys:

- Once a table is created as partitioned, you cannot directly change its partitioning key. Try one of these approaches:
  - Unload the table to a file, drop and re-create the table with the new partitioning key, and reload the table.
  - Create a new version of the table with a temporary name and the new partitioning key, load the new table from the old one (the fastest way is: `declare mycursor for select * from t1`, followed by `load from mycursor of cursor replace into t1_new`), drop the old table, rename the new table to the real name, and recreate indexes and re-do other steps as when the table was originally created.
- With ALTER TABLE you can add or drop a partitioning key, but only for a non-partitioned table.
- The columns in any unique or primary key constraint defined on the table must be a super-set of the partitioning key.
- Avoid unbalanced distribution of rows across partitions by choosing partitioning key columns with high cardinality. If this cannot be achieved, try to use columns with uniformly distributed values. After a table is populated you can check how many of its rows are in each partition, and how many of its rows map to each entry in the partitioning map (for possible redistribution purposes) running queries to analyze the table contents. Unless you have extremely large tables, differences of a few percent in cardinalities per partition can be ignored.
- Partitioning keys should not include columns that are updated frequently. Whenever a partitioning key value is updated, DB2 might need to drop the row and re-insert it into a different partition, as determined by hashing the new partitioning key value.
- Unless a table is not critical or you have no idea what a good partitioning key choice would be, you should not let the partitioning key be chosen by default. For the record, the default partitioning key is the first column of the primary key, and if there is none, the first column that has an eligible data type.

- Make sure that you understand collocation and the different join types. For information, see the DB2 Partitioning and Clustering Guide, SC27-2453.

  Collocated tables must fulfill the following requirements:

  – Be in the same database partition group, one that is not being redistributed. (During redistribution, tables in the database partition group might be using different partitioning maps. They are not collocated.)

  – Have partitioning keys with the same number of columns.

  – Have the corresponding columns of the partitioning key be partition compatible.

  – Be in a single partition database partition group, if not in the same database partition group, defined on the same partition.

- Partition compatibility is defined between the base data types of corresponding columns of partitioning keys. Partition-compatible data types have the property that two variables, one of each type, with the same value, are mapped to the same partitioning map index by the same partitioning function. Partition compatibility has the following characteristics:

  – Internal formats are used for DATE, TIME, and TIMESTAMP. They are not compatible with each other, and none is compatible with CHAR or VARCHAR.

  – Partition compatibility is not affected by columns with NOT NULL or FOR BIT DATA definitions.

  – NULL values of compatible data types are treated identically. Different results might be produced for NULL values of non-compatible data types.

  – The base data type of a UDT is used to analyze partition compatibility.

  – Decimals of the same value in the partitioning key are treated identically, even if their scale and precision differ.

  – Trailing blanks in character strings (CHAR, VARCHAR, GRAPHIC, or VARGRAPHIC) are ignored by the system-provided hashing function.

  – CHAR or VARCHAR of different lengths are compatible data types.

  – REAL or DOUBLE values that are equal are treated identically even though their precision differs.

Multidimensional data clustering (MDC)

Multidimensional data clustering is a method of organizing and storing data along multiple dimensions. Organizing data in this manner can greatly improve the performance of certain query types and reduce reorg and index maintenance.

The MDC feature, introduced with DB2 UDB Version 8, provides a method for automatic clustering of data along multiple dimensions. Prior to Version 8, DB2 UDB only supported single-dimensional clustering of data, through clustering indexes. Using a clustering index, DB2 maintains the physical order of data on pages in the key order of the index, as records are inserted and updated in the table. Clustering indexes greatly improves the performance of range queries that have predicates containing one or more keys of the clustering index. With good clustering, only a portion of the table needs to be accessed. In addition, when the pages are sequential, more efficient prefetching can be performed. Secondary indexes can be created to access the tables when the primary key index is not applicable. Unfortunately, secondary indexes perform many random I/O accesses against the table for a simple operation, such as a range query.

The MDC feature addresses this important deficiency in database systems. MDC enables a table to be physically clustered on more than one key (or dimension) simultaneously. MDC tables are organized physically by associating records with similar values for the dimension attributes in an extent (or block). DB2 maintains this physical layout efficiently and provides methods of processing database operations that provide significant performance improvements.

Figure A-5 presents a conceptual diagram of multi-dimensional clustering along three dimensions: region, year, and color.

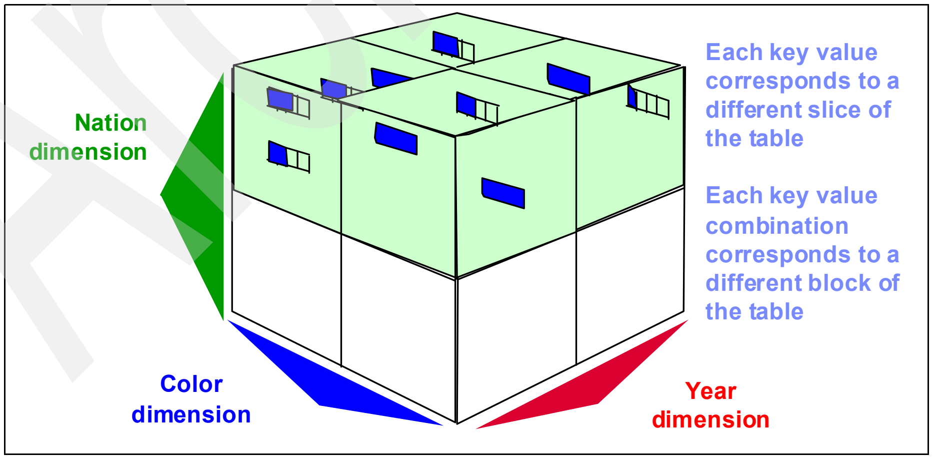

Nation dimension
Color dimension
Year dimension
Each key value corresponds to a different slice of the table
Each key value combination corresponds to a different block of the table

Figure A-5 MDC conceptual diagram

Each block contains only rows that have the same unique combination of dimension values. The set of blocks that have the same unique combination of dimension values is called a cell. A cell might consist of one or more blocks in the MDC table. As depicted in Figure A-5 on page 531, a cell can be described as a combination of unique values of year, nation, and color, such as 1997, Canada, and Blue. All records that have 1997 as year, Canada as nation, and blue as color are stored in the same extents of that table.

With MDC tables, clustering is guaranteed. If an existing block satisfies the unique combination of dimension values, the row is inserted into that block, assuming there is sufficient space. If there is insufficient space in the existing blocks, or if no block exists with the unique combination of dimension values, a new block is created.

Example A-5 shows the DDL for creating an MDC table. The ORGANIZE keyword is used to define an MDC table.

Example A-5 MDC example

```
CREATE TABLE sales (
    cust_id INTEGER,
    year    CHAR(4),
    nation  CHAR(15),
    color   CHAR (12),
    amount  INTEGER)
PARTITIONING KEY(cust_id)
ORGANIZE BY (nation,color,year)
```

MDC introduced a new type of index, called a block index. When you create an MDC table, the following two kinds of block indexes are created automatically:

- Dimension block index

  A dimension block index per dimension contains pointers to each occupied block for that dimension.

- Composite block index

  A composite block index contains all columns involved in all dimensions specified for the table, as depicted in Figure A-6 on page 533. The composite block index is used to maintain clustering during insert and update activities. It can also be used for query processing.

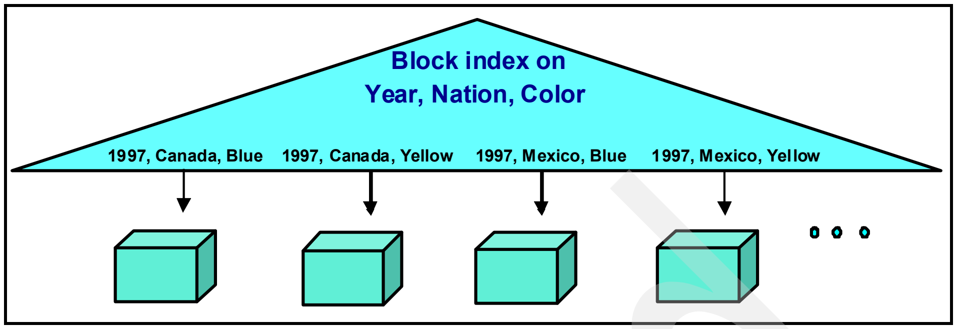


Figure A-6 Block index

With MDC, data is organized on disk based on dimensions. Queries can skip parts of the tablespace that the optimizer has determined do not apply. When data is inserted, it is automatically put in the proper place so you no longer need to reorganize the data. In addition, because one index entry represents the entire data page (versus having one index entry per row with traditional indexes), MDCs reduce the overall size of the space required for the indexes. This reduces disk requirements and produces faster queries because of the reduced amount of I/O needed for a query. MDCs also improve delete performance because DB2 now only has to drop a few data pages. Inserts are also faster because DB2 rarely has to update an index page, only the data page.

DB2 manages MDC tables by blocks according to dimensions instead of row IDs (RIDs) as implemented in clustering indexes. This is depicted in Figure A-7.

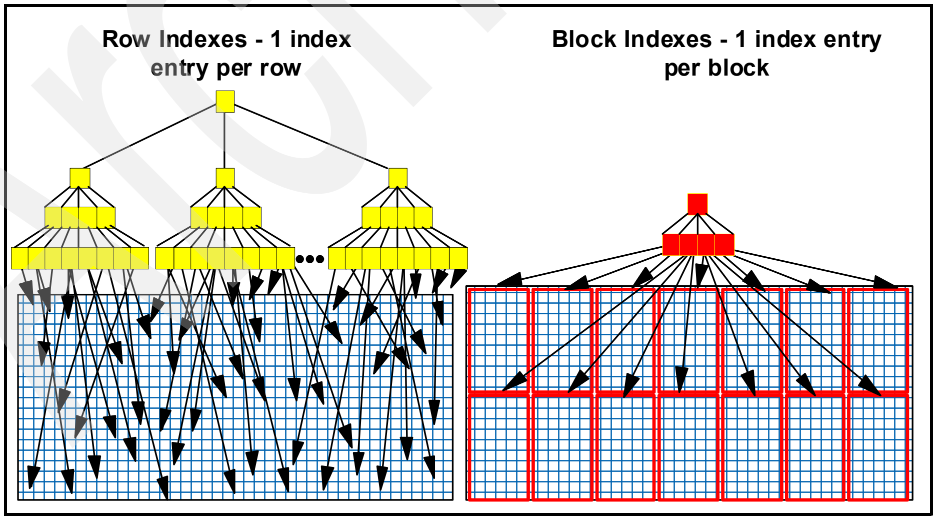


Figure A-7 MDC page and block retrieval comparison

RID indexes require one RID per data record. Block indexes (BIDs) only have one index entry pointing to the block. Consequently, the size of the BID index is dramatically smaller than the RID index. This means the probability that a BID index page needed is in memory cache when needed is much higher than with RID indexes. You can have multiple block indexes on the same fact table because these indexes are small, so there is little disk space required to have many indexes.

DB2 has the ability to perform fast rollouts of data from in MDCs. For a rollout deletion, the deleted records are not logged. Instead, the pages that contain the records are made to look empty by reformatting parts of the pages. The changes to the reformatted parts are logged, but the records themselves are not logged. There is no change in the logging of index updates, so the performance improvement depends on how many RID indexes there are. The fewer RID indexes, the better the improvement is as a percentage of the total time and log space.

Multidimensional clustering is a unique capability that is targeted for large database environments, providing an elegant method for flexible, continuous, and automatic clustering of data along multiple dimensions. The result is significant improvement in the performance of queries, as well as a significant reduction in the overhead of data maintenance operations (such as reorganization) and index maintenance operations during insert, update, and delete operations.

Performance considerations

The performance of an MDC table is dependent upon the proper choice of dimensions and the block (extent) size of the tablespace for the given data and application workload. A poor choice of dimensions and extent size can result in low disk storage use and poor query access and load utility performance.

Choosing dimensions

In choosing dimensions, you must identify the queries in the existing or planned workloads that can benefit from multidimensional clustering. For existing applications, the workload might be captured from DB2 Query Patroller or the dynamic SQL snapshot and the SQL statement Event Monitor. This workload can be used as input into the DB2 Design Advisor to analyze the workload and recommend dimensions for the MDC tables.

For a new table or database, you need a good understanding of the expected workload. The following dimension candidates are typical:

- Columns in range or equality or IN-list predicates
- Columns with coarse granularity
- Roll-in and roll-out of data
- Columns referenced in GROUP BY and ORDER BY clauses
- Foreign key columns or join clauses in fact table of a star schema database
- Combinations of the above

Usually there are several candidates. Rank them and evaluate the impacts on the workload and cell density. When there are no good low or medium cardinality dimensions, or the chosen dimension has high cardinality, consider using generated columns. The generated expressions used should be monotonic (increasing/decreasing) for those columns used in the range queries.

Choosing the extent size

Extent size is related to the concept of cell density, which is the percentage of space occupied by rows in a cell. Because an extent only contains rows with the same unique combination of dimension values, significant disk space could be wasted if dimension cardinalities are high (for example, when there is a dimension with unique values that would result in an extent per row).

The ideal MDC table is one where every cell has enough rows to fill one extent exactly, but this can be difficult to achieve. The objective of this section is to detail the ideal MDC table.

Note: The extent size is associated with a tablespace, and applies to all of the dimension block indexes, as well as the composite block index. This makes the goal of high cell density for every dimension block index and the composite block index difficult to achieve.

Defining small extent sizes can increase cell density, but increasing the number of extents per cell can result in more I/O operations, and potentially poorer performance when retrieving rows from this cell. However, unless the number of extents per cell is excessive, performance should be acceptable. If every cell occupies more than one extent, it can be considered excessive.

At times, due to data skew, cells occupy a large number of extents while others occupy a small percentage of the extent. Such cases signal a need for a better choice of dimension keys. Currently, the only way to determine the number of extents per cell requires the DBA to issue appropriate SQL queries or use db2dart.

Performance might be improved if the number of blocks can be reduced by consolidation. Unless the number of extents per cell is excessive, this situation is not considered an issue.

Note: The challenge is to find the right balance between sparse blocks/extents and minimizing the average number of extents per cell as the table grows to meet future requirements.

Range partitioning

Table partitioning (TP), often known as range partitioning, was introduced in DB2 UDB V9. TP is a data organization scheme where table data is divided across multiple storage objects, called *data partitions* (not to be confused with database partitions (DPF), according to values in one or more table columns. These storage objects can be in different tablespaces, in the same tablespace, or in a combination of both.

DB2 UDB supports data partitions or data ranges based on a variety of attributes. Commonly used partitioning schemes are to cluster together data partitions by date, such as by year or month, or have numeric attributes for partitioning. For instance, records with IDs from one to one million are stored in one data partition, IDs from one million to two million in another data partition, and so on. Or, for example, records for clients with names starting with A–C could be in one data partition, D–M in the second data partition, N–Q in a third data partition, and R–Z in the last data partition.

Although you have the option of referring to data partitions by names or numbers (useful for data partition operations), they can be completely transparent to applications. That is, applications can continue to access data by specifying column and table names, and do not need to worry about where the data partitions reside.

DB2 provides flexibility for creating partitioned tables. For example, if there is one year of data, it can be partitioned by date so that each quarter resides in a separate data partition. The create table syntax in Example A-6 shows how this can easily be done. The graphical DB2 Control Center can also be used for creating and managing data partitions.

Example A-6 Create table syntax for partitioning

```
CREATE TABLE orders(id INT, shipdate DATE, …)
  PARTITION BY RANGE(shipdate)
    (
    STARTING '1/1/2006' ENDING '12/31/2006'
      EVERY 3 MONTHS)
```

This results in a table being created with four data partitions, each with three months of data, as shown in Figure A-8.

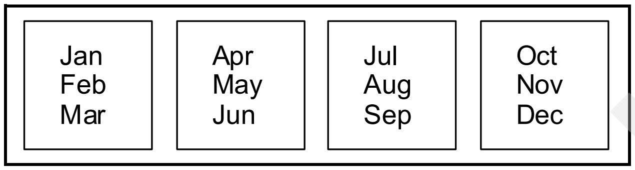


Figure A-8 Table partitions

Table partitioning provides the following benefits:

- Improved manageability

  DB2 UDB allows the various data partitions to be administered independently. For example, you can choose to back up and restore individual data partitions instead of entire tables. This enables time-consuming maintenance operations to be segmented into a series of smaller operations.

- Increased query performance

  The DB2 Optimizer is data partition aware. Therefore, during query execution, only the relevant data partitions are scanned. This eliminates the need to scan data partitions that are not impacted by the query, and can result in improved performance. This is illustrated in Figure A-9.

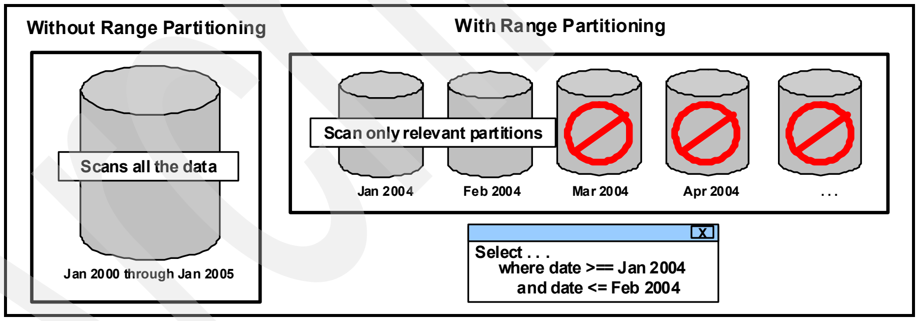


Figure A-9 Scanning only relevant partitions

- Fast roll-in/roll-out

  DB2 9 allows data partitions to be easily added or removed from the table without having to take the database offline. This ability can be useful in a data warehousing environment where there is the need to load or delete data to run decision-support queries. For example, a typical insurance data warehouse might have three years of claims history. As each month is loaded and rolled in to the data warehouse, the oldest month can be archived and removed (rolled out) from the active table. This method of rolling out data partitions is also more efficient, as it does not need to log delete operations, which is the case when deleting specific data ranges.

- Better optimization of storage costs

  Table partitioning in DB2 UDB enables better integration with hierarchical storage models. By only using the fastest and most expensive storage hardware for only the most active data partitions, DB2 UDB allows optimization of the overall storage costs and improves performance. If most queries only run against the last three months of data, there is an option to assign slower and less expensive storage hardware to the older data.

- Larger table capacity

  Without partitioning, there are limits on the maximum amount of data a storage object, and hence a table, can hold. However, by dividing the contents of the table into multiple storage objects or data partitions, each capable of supporting as much data as in a non-partitioned table, you can effectively create databases that are virtually unlimited in size.

Starting with DB2 V9.7 there are indexes that see rows of data across all partitions in a partitioned table (known as nonpartitioned indexes), or you can have the index itself partitioned such that each data partition has an associated index partition.

Using partitioned indexes that match the underlying partitioned tables in a large data warehouse environment can improve the overheads involved in attaching or detaching partitions.

When a partition is attached to a table, a **SET INTEGRITY** command must be issued before the data becomes visible. Among other tasks the **SET INTEGRITY** command updates any existing non-partitioned indexes that could be a lengthy and use a considerable amount of log space.

When the new partition is attached to a table with partitioned indexes and indexes are already built that match these indexes, the SET INEGRITY statement does not incur the performance and logging overhead associated with index maintenance.

Table partitioning is similar to MDC in that it enables rows with similar values to be stored together. However, the following TP characteristics distinguish it from MDC:

- TP supports partitioning a table into data partitions along a single dimension. A common design is to create a data partition for each month. MDC supports defining multiple dimensions.
- With TP, the user can manually define each data partition, including the range of values to include in that data partition. MDC automatically defines a cell (and creates blocks to store data for that cell) for each unique combination of MDC dimension values.
- Each TP partition is a separate database object (unlike other tables, which are a single database object). Consequently, TP supports attaching and detaching a data partition from the TP table. A detached partition becomes a regular table. Also, each data partition can be placed in its own tablespace, if desired.
- Essentially, the distinct TP benefit is related to adding or removing large numbers of rows from a table. That is, roll in and roll out. For readers familiar with the use of Union All View (UAV) to partition history tables on date, TP can be considered an analogous, but superior, solution.

Using all partition schemes together

Each partitioning scheme can be used in isolation or in combination with other data organization schemes. Each clause of the CREATE TABLE statement includes an algorithm to indicate how the data should be organized. The following three clauses demonstrate the levels of data organization that can be used together in any combination:

- DISTRIBUTE BY

  DISTRIBUTE BY spreads data evenly across database partitions. Use this clause to enable intraquery parallelism and distribute the workload across each database partition. This concept is known as database partitioning and is enabled using DPF in DB2.
- PARTITION BY

  PARTITION BY groups rows with similar values of a single dimension in the same data partition. This concept is known as table partitioning.
- ORGANIZE BY

  ORGANIZE BY groups rows with similar values on multiple dimensions in the same table extent. This is known as MDC.

This syntax allows for consistency between the clauses as well as allowing for future algorithms for data organization. Combining the DISTRIBUTE BY and PARTITION BY clauses of the CREATE TABLE statement allows data to be spread across database partitions spanning multiple tablespaces.

DB2 is the first data server to support all three methods of grouping data at the same time. This is a major innovation in improving data management and information availability. Figure A-10 illustrates all three DB2 data organization schemes being used in conjunction with each other.

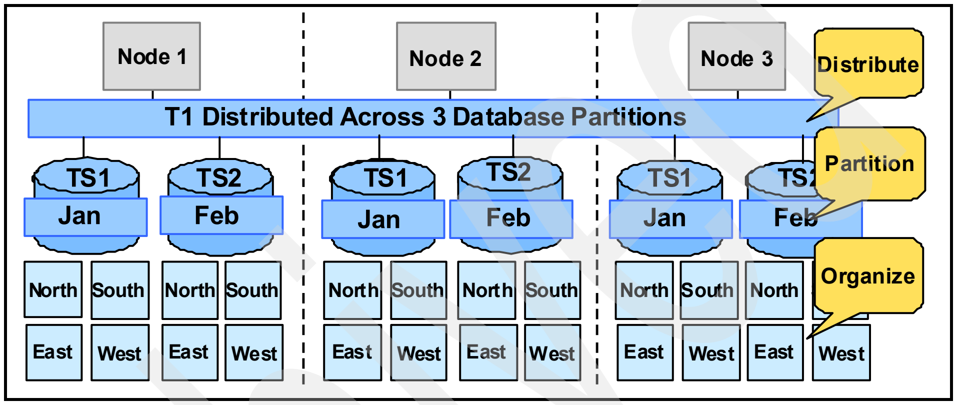


Figure A-10 Using partitioning schemes together

DB2 partitioning options summary

The CREATE table statement now provides three complementary ways to group data in a database table, and is shown in Table A-1.

Table A-1 Create table options

CREATE TABLE statement	DB2 feature
DISTRIBUTE BY HASH	DPF
ORGANIZE BY DIMENSION	MDC
PARTITION BY RANGE	TP

These clauses can be used in any combination to achieve the desired effect. Table A-2 summarizes the feature terminology.

Table A-2 DB2 feature terminology

DB2 feature name	Name of part	Column used to partition the data	Other terms
DPF	Database partition	Distribution key	Also known as the partitioning key
MDC	Cells consisting of blocks	Dimensions	Block indexes
TP	Data partition	Table partitioning key	

Table A-3 provides a feature comparison.

Table A-3 DB2 feature comparison summary

Feature	How the feature organizes data	Benefits
DPF	Distributes rows across database partitions	Scalability. Computer resources in the form of database partitions are added as the database grows
MDC	Groups rows with similar values on multiple dimensions in the same physical location in the table (the block)	Query performance. Data organized for faster retrieval especially for queries that involve ranges of multiple predicates
TP	Groups all rows in a specified range of a single dimension in the same data partition	Data movement. Large volumes of data might be added or removed by adding or removing data partitions.

Table design

When designing a table, the use of each feature can be considered independently. As examples:

- Determining the distribution key for DPF is not affected by whether MDC and TP are also being used.
- Whether a column should be used as table partitioning key for TP is not impacted by whether that column is also used by an MDC dimension and visa versa. Each decision can be made independently.

The way each feature works, with regard to indexing is not changed with the introduction of new partitioning features. For example, when MDC was introduced, its index aspects did not change DPF index aspects. Also, when TP was introduced, it did not change indexing related to DPF or MDC. Keeping this in mind when learning about these features can help avoid confusion.

For example, suppose that you are learning about TP and you come across the statement that TP has global indexes. You should not infer that there is a change in how indexes are handled in DPF. In statements such as this, the term *global* merely means that indexes are global across TP data partitions.

Fact (or history) tables in data warehouses make excellent candidates for use with each partitioning feature, as shown in Table A-4.

Table A-4 Fact table characteristics

Feature	Suitable table characteristic	Characteristic of fact table
DPF	Large tables. Larger than can be handled by a single set of CPUs and I/O channels.	Fact tables might contains millions or billions of rows.
MDC	Queries whose result sets consist of rows with similar values along multiple dimensions	These are typical data warehouse queries.
TP	Tables (such as snapshot tables) where large number of rows are appended and later archived.	The most recent snapshot is added on a regular basis and the oldest snapshots are periodically archived

Rules of thumb

In this section we provide guidelines that help to understand the nature of design decisions.

For DPF, the top priority when selecting a distribution key is to find one that distributes the rows evenly across the database partitions. When this situation is not achieved, the result is data skew. This means that database partitions are assigned a disproportionate number of table rows, which can create a performance bottleneck. Columns with many distinct values are good candidates. Other considerations include choosing a column that maximizes the performance of joins.

Another DPF design decision concerns the number of database partitions. The number of database partitions is not primarily a table design consideration. Rather, it is an overall system design consideration based on the anticipated raw data size of the entire database and capacity of the server hardware. Many systems need fewer than 20 database partitions. However, the largest systems might have many more. The common range can be expected to increase due to the trend toward larger data warehouses.

For MDC, a key decision is which columns will serve as MDC dimensions. The design challenge is to find the best set of dimensions and granularity to maximize grouping but minimize storage requirements. This requires knowledge of the pattern of queries to be run. Good candidates are columns that have any or all of the following characteristics:

- Used for range, equality, or IN-list predicates
- Used to roll-in, roll-out, or other large-scale deletes of rows
- Referenced in GROUP BY or ORDER by clauses
- Foreign key columns
- Columns in the join clauses in fact table of star schema database
- Coarse granularity, that is, few distinct values

A typical design includes an MDC dimension for a column representing date plus dimensions on 0–3 other columns, such as region and product type.

For TP, design decisions include selecting the column to use as the table partitioning key and number of partitions. Usually the table partitioning key is a time-based column. Each partition corresponds to a time range of data that is rolled out at the same time. For example, a table that has monthly rollout has a partition for each month of data. One design consideration unique to TP is the need to handle rows that fall outside the ranges of values defined in the CREATE table statement.

A typical design for a database with monthly roll-out based on sale_date is to use sale_date as the table partitioning key and create a separate partition for each month.

Generally, one of the MDC dimensions are a time-based column so the same column can be used for both MDC and TP. The MDC granularity can be finer-grain than the TP data partition size.

These points are summarized in Table 2.

Table 2 Design rule of thumb summary

Partitioning feature design decision	**Rule of thumb**
DPF Column to use as the distribution key	A column with many distinct values, preferably used to join with other large tables.
MDC Columns to use as dimension	Typically a combination of date and other low cardinality dimension keys
TP Column used as the table partitioning key and number of partitions	A column representing a time period.

DB2 performance features

Many features that improve performance are built into DB2. However, there are several design features that are highly recommended for all data warehouse implementations.

MQTs

Sometimes a simple change in physical database structure dramatically improves query performance. In addition to indexes, DB2 UDB provides MQTs, which often are more efficient than indexes.

Think of an MQT as a type of materialized view. Both views and MQTs are defined on the basis of a query. The query on which a view is based is run whenever the view is referenced. However, an MQT actually stores the query results as data, and you can work with the data that is in the MQT instead of the data that is in the underlying tables.

MQTs contain the results of queries, as illustrated in Figure A-11. In fact, the DDL, or SQL code, used to create an MQT contains a query. MQTs are based on queries of base tables. (A base table is a regular table on which an MQT is based.) The data contained in an MQT is usually an aggregation or summary of the data in the underlying base tables.

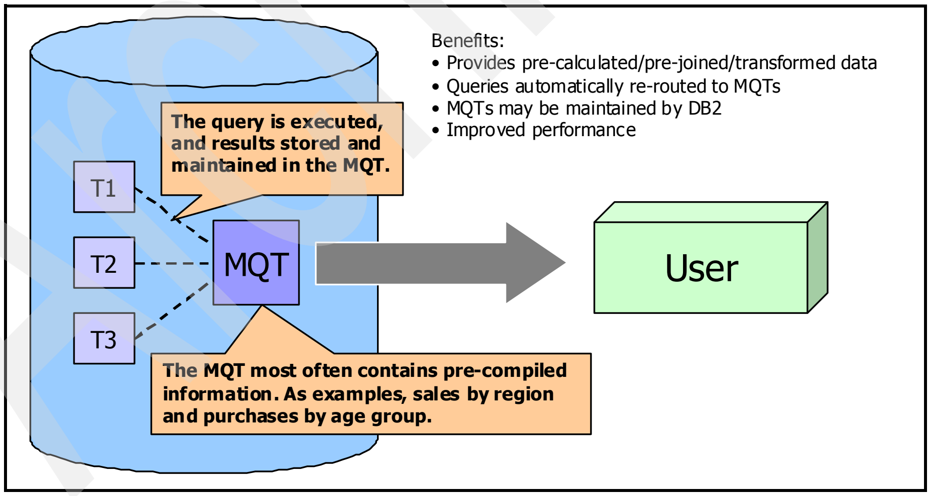

Benefits:
• Provides pre-calculated/pre-joined/transformed data
• Queries automatically re-routed to MQTs
• MQTs may be maintained by DB2
• Improved performance
The query is executed, and results stored and maintained in the MQT.
T1
T2
T3
MQT
User
The MQT most often contains pre-compiled information. As examples, sales by region and purchases by age group.

Figure A-11 MQT

MQTs can significantly improve the performance of queries, especially complex queries. If the optimizer determines that a query or part of a query could be resolved using an MQT, the query might be rewritten to take advantage of the MQT.

Consider a database where queries are run that aggregate data regularly, such as total sales over a given period for various regions. When reports are generated or queries are executed to retrieve this aggregated data, processing can take from minutes to hours each time these queries are executed. You can create an MQT to support these reports and queries. The processing is performed one time to aggregate the data in the MQT. All sales that were made per region per time period can be aggregated for all regions and time periods, concurrently. Each time a query is executed to request sales data for a specific region or a report is generated for all regions, the data can be read from an MQT. When DB2 provides the data by reading from the MQT instead of recalculating aggregated data repeatedly as queries execute, response time is reduced from minutes, hours, or longer, to seconds, in many cases. The cost of (re)processing the requests is reduced to the cost of reading the aggregated data from the MQT, and overall response time is reduced.

MQTs are a powerful way to improve response time for complex queries, especially queries that might require the following operations:

- Aggregating data over one or more dimensions
- Joining and aggregating data over a group of tables
- Combining a commonly accessed subset of data, such as a frequently accessed horizontal or vertical partition
- Repartitioning data from a table, or part of a table, in a partitioned database environment

The advantage of using DB2 MQTs is that a user does not need to know that they exist to use them. Knowledge of MQTs is integrated into the SQL compiler. When a dynamic SQL query is submitted, the DB2 Optimizer examines the query, the related tables, and any MQTs that are based on those tables. If the optimizer determines that an MQT can answer the query at a lower processing cost, it automatically rewrites the query to use the MQT instead of all or of the base tables. The DB2 Optimizer changes it, transparently, without requiring special commands or changes to the application code. If an MQT is used, the EXPLAIN facility provides information about which MQT was selected.

Because the optimizer is aware of all MQTs, no changes to user queries are required after you build an MQT. After the DBA creates the MQT, many queries might share in the benefit of the MQT and users and applications might not even know that they are exploiting this powerful feature.

An MQT is just a table and can be directly accessed by name. In this case, the DB2 Optimizer always accesses the MQT because it is explicitly requested. However, this is discouraged because, when direct dependencies are created on any MQTs by query tools, the ability of developers and DBAs to drop, alter, or otherwise refine these MQT tables is restricted.

Version 9.7 includes enhancements that improve MQT matching capabilities, which improves query performance. MQTs are matched in the following new scenarios:

- An MQT that specifies a view, possibly containing a complex query, can be matched to a query that uses that view. In previous releases, queries that specified a view with a construct such as OUTER JOIN or complex UNION ALL query could not be matched to an MQT. You can now create views for the portion of queries that is not matchable, and create MQTs that do a simple SELECT operation from these views. If the queries also reference these views, the optimizer considers replacing the view with the corresponding MQT.
- Queries containing a SELECT DISTINCT or GROUP BY clause can be matched to MQTs whose definitions start with SELECT DISTINCT.
- Queries containing an EXISTS predicate can also be matched to MQTs with an appropriate EXISTS predicate.
- Additional scenarios involving datetime functions are better matched. For example, queries containing a QUARTER function can be matched to an MQT returning MONTH. Another example is when a query contains DAYOFWEEK function and the MQT contains DAYOFWEEK_ISO (or the reverse scenario).
- Other previously unmatched scenarios are now matched when referential integrity constraints (or informational referential integrity) are defined.

Creating and maintaining MQTs

MQTs can be created by using the **`CREATE TABLE`** command used for regular tables, but specific clauses for MQTs must be included, as in Example A-7. To create an MQT, a user must be granted permission to create tables.

Example A-7 MQT sample

```
Create table sales_summary as (select sales_person, region, sum(sales)
 as total_sales
 from sales group by sales_person, region)
  data initially deferred refresh deferred
```

When discussing the data contained in an MQT, you must also consider data currency. That is, how current is the data? Is it synchronized with the data in the underlying base tables, or have the underlying data been updated since the MQT was last populated? Does currency matter for the application? Sometimes the data needs to be kept current in as close to real-time as possible (for example, to make sure that bank balances are sufficient when a withdrawal is requested).

Alternatively, data might need to be captured in a snapshot and held (for example, when generating month-end or year-end reports). While the production system might need to keep updating, the MQTs can be refreshed at the end of the time period, giving a fixed snapshot for consistent and extensive analysis.

To support these scenarios, MQTs can be maintained automatically by DB2 or maintained manually and updated by a user or an application. Each method has advantages and disadvantages. For example, system-maintained MQTs are always current and available for use, but the data in them cannot be altered. User-maintained MQTs can be altered for what-if types of analysis or can capture a snapshot in time and be held. However, the data in the MQT might not be sufficiently current at any given time unless the user updates the data again. In the latter case, MQTs can also have staging tables where delta data can be queued until processed into the MQT by a table refresh. This can substantially accelerate the update/refresh process.

After the MQT has been created, it has to be populated and synchronized with the data in the base tables. DB2 provides two approaches to MQT maintenance:

- Refresh immediate

  DB2 watches for changes in any of the tables that affect values maintained in an MQT. If an insert, update, or delete occurs in a source table for the MQT, DB2 includes the appropriate changes to the MQT as part of the originating transaction. In this mode, DB2 is responsible for keeping the MQT consistent.

- Refresh deferred

  In this mode, changes to source tables do not trigger DB2 automatic MQT maintenance. This gives the database administrator (DBA) full control of when the MQT maintenance should be performed, and makes the DBA responsible for determining MQT currency. In such a case, there is latency between the contents of the MQT and the contents of the base tables. Refresh-deferred MQTs can be refreshed incrementally with the use of staging tables.

A system-maintained MQT could be populated by DB2 automatically through the use of the **`REFRESH`** or **`SET INTEGRITY`** commands. A user-maintained MQT could be loaded manually by the user through the use of the **`LOAD`** command.

It is likely that many MQTs is a large data warehouse will be maintained as part of the ETL process by using techniques such as utility loads and mass inserts or updates. Where batch processes frequently maintain the base tables upon which the MQTs are defined, the same techniques are likely to be used for the MQTs.

In addition to the features that are specific to MQTs, these tables look and function much like standard DB2 tables. Also, MQTs can be tuned. For example, indexes can be created on MQTs and the RUNSTATS utility can be issued on MQTs. MQTs can also be built on multidimensional clustered tables (MDCs), and they can be MDCs as well.

Replicated tables

The key performance goal for database design with DPF (hash partitioning) is collocation. This is where the joining of tables is done in one or more individual partitions, in contrast with directed or broadcast joins, which involve rows being sent to specific or all partitions.

The partitioning key (used in the CREATE statement to define a partitioned table) is used by the system to distribute as evenly as possible the data for that table across as many database partitions as desired, so as to maximize parallelism and therefore performance. The term collocation is used to refer to tables that have the same partitioning key. Then, when joining these tables, the join can be done with good performance. Therefore, when designing the tables, it is advantageous to find common keys that are good partitioning keys.

Columns are collocatable if the following statements are all true:

- Columns are in the same partition group.
- Columns have the same number of partition key columns.
- Data types of partition key columns are pair-wise compatible.

If tables cannot be partitioned on the same keys as other tables and are modest in size and are typically read only, we recommend that replicated tables be used to assist in the collocation of joins. For example, if a star schema contains a large fact table spread across twenty nodes, the joins between the fact table and the dimension tables are most efficient if these tables are collocated. If all of the tables are in the same database partition group, at most one dimension table is partitioned correctly for a collocated join. The other dimension tables cannot be used in a collocated join because the join columns on the fact table do not correspond to the partitioning key of the fact table. To achieve collocation for a particular small table, create replicated tables. This means that a full copy of the table is maintained in each partition and its rows never have to be sent between partitions to do a join.

When you create a replicated materialized query table, the source table can be a single-node table or a multi-node table in a database partition group. In most cases, the replicated table is small and can be placed in a single-node database partition group, as shown in Example A-8, where TS_SDPG is the tablespace in the single data partition group. You can limit the data to be replicated by specifying only a subset of the columns from the table or by specifying the number of rows through the predicates used, or by using both methods.

Example A-8 Single partition source table

```
CREATE TABLE TIME_DIM (
   TIME_ID SMALLINT NOT NULL,
   QUARTER CHAR(4) NOT NULL,
   MONTH_ID SMALLINT NOT NULL,
   MONTH_NAME CHAR(10) NOT NULL,
   MONTH_ABREV CHAR(3) NOT NULL,
   SEASON  CHAR(6) NOT NULL)
      IN TS_SDPG
```

The replicated table is created using the MQT syntax with the key word REPLICATED, as in Example A-9. This specifies that data stored in the table is physically replicated on each database partition of the database partition group of the tablespace in which the table is defined. This means that a copy of all the data in the table exists on each of these database partitions.

Example A-9 Multi-partitioned replicated table

```
CREATE TABLE TIME_DIM_R AS ( SELECT * FROM TIME_DIM )
   DATA INITIALLY DEFERRED
   REFRESH DEFERRED
      IN TS_PDPG REPLICATED
```

A replicated MQT can also be created in a multi-node database partition group so that copies of the source table are created on all of the partitions. Joins between a large fact table and the dimension tables are more likely to occur locally in this environment than if you broadcast the source table to all partitions.

Indexes on replicated tables are not created automatically. You can create indexes that are different from those on the source table. However, to prevent constraint violations that are not present on the source tables, you cannot create unique indexes or put constraints on the replicated tables. Constraints are disallowed even if the same constraint occurs on the source table.

Replicated tables can be referenced directly in a query, but you cannot use the NODENUMBER() predicate with a replicated table to see the table data on a particular partition.

Use the EXPLAIN facility to see if a replicated MQT was used by the access plan for a query. Whether the access plan chosen by the optimizer uses the replicated MQT depends on the information that needs to be joined. The optimizer might not use the replicated MQT if the optimizer determines that it would be cheaper to broadcast the original source table to the other partitions in the database partition group.

Replicated MQTs improve performance of frequently executed joins in a partitioned database environment by allowing the database to manage precomputed values of the table data.

You are not restricted to having all tables divided across all database partitions in the database. DB2 supports partial declustering, which means that you can divide tables and their tablespaces across a subset of database partitions in the system.

An alternative to consider when you want tables to be positioned on each database partition is to use MQTs and replicate those tables. You can create a materialized query table containing the information that you need, and replicate it to each node.

Compression

Disk storage systems can often be the most expensive components of a data warehouse solution. For large warehouses or databases with huge volumes of data, the cost of the storage subsystem can easily exceed the combined cost of the hardware server and the data server software. Therefore, even a small reduction in the storage subsystem can result in substantial cost savings for the entire database solution.

Row compression

DB2 V9 introduced technology that compresses row data to reduce storage requirements, improve I/O efficiency, and provide quicker data access from the disk. DB2 V9 uses a dictionary-based algorithm for compressing data records. That is, DB2 V9 can compress rows in database tables by scanning tables for repetitive, duplicate data and building dictionaries that assign short, numeric keys to those repetitive entries. Text data tends to compress well because of recurring strings as well as data with lots of repeating characters, or leading or trailing blanks.

DB2 examines entire rows for repeating entries or patterns, and not just particular fields or parts of rows. Take, for example, the two rows in Table A-5. In this example, not only the repeating values (of 500) in the Dept column are compressed, but the repeating pattern (of Plano, TX, and 24355) that spans the City, State, and ZipCode columns is also compressed as a single value.

Table A-5 Repeating patterns

Name	Dept	Salary	City	State	Zip
Fred	500	1000	Plano	TX	24355
John	500	2000	Plano	TX	24355

Figure A-12 compares how DB2 would store the row normally and in compressed formats.

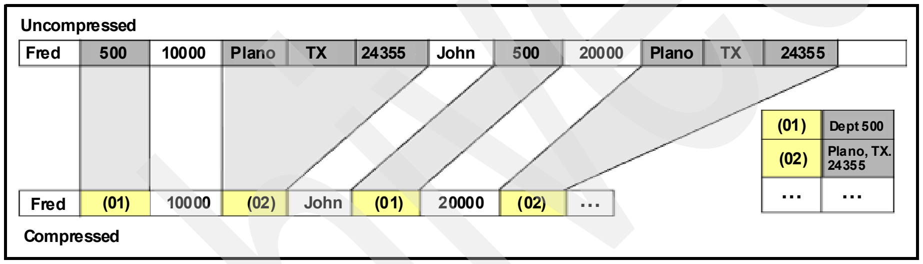


Figure A-12 Comparison of uncompressed and compressed data storage

The dictionary for compression and decompression lookup is stored in hidden objects in the database, occupies little space, and is cached in memory for quick access. Even for large tables, the compression dictionary is typically only on the order of 100 KB. However, there can be instances when certain data sets do not compress well or there are data size conditions that result in little compression. DB2 has intelligent algorithms to determine such scenarios and does not perform compression when it does not yield any disk-space-saving benefits.

The data row compression feature in DB2 for LUW is similar to the compression capabilities available on DB2 for z/OS. However, it differs from the page-level compression techniques offered by other database vendors, where a compression dictionary is built for each page in the database. By building a compression dictionary at the table rather than page level, patterns across the entire table are analyzed, generally resulting in improved disk savings with DB2.

Enabling compression

Data row compression in DB2 can be turned on when tables are created using the COMPRESS YES option. It can also be enabled for an existing table using the ALTER TABLE command, as shown in Example A-10.

Example A-10 Enabling compression

```
CREATE TABLE <table name> --->
         |---COMPRESS NO---|
     ----+-----------------+---->
         |---COMPRESS YES--|

ALTER TABLE <table name> --->
       --+---------------------+---->
         |--COMPRESS--+-YES--+--|
                      |--NO--|
```

The compression takes effect only after the table dictionary is built, which is usually during the table REORG phase, as seen in Example A-11.

Example A-11 Building the table dictionary during REORG

```
>--REORG--<table name>--+-----------------------+---->
                        '--INDEX--<index name>--'
     .-ALLOW READ ACCESS-.
>--+-+-------------------+--+---------------+--+-----------+-->
     '-ALLOW NO ACCESS---'  '-USE--<tbspace>-'  '-INDEXSCAN-'
                   .-KEEPDICTIONARY---.
>--+------------+-+------------------+-+-->
   '-LONGLOBDATA-' '-RESETDICTIONARY---'
```

When compressing a large table, it might be useful to populate the table with a small set of representative or sample data first. The process of building the compression dictionary can be fairly quick using a small data set. And if the set is a good representative sample, the compression works well even on new data that is added to the table, without DB2 having to analyze the new data. If, however, the type of data stored in the table evolves over time, the dictionary can be kept up to date using REORG.

Estimated savings

DB2 provides the INSPECT tool to help you determine the compression ratio estimate for a particular table or data set, as depicted in Example A-12 on page 553. This tool collects a sample of the table data and builds a compression dictionary from it. This dictionary is used to test compression against the records contained in the sample. From this test, compression savings are estimated.

Example A-12 Inspect command

```
INSPECT ROWCOMPESTIMATE TABLE NAME Sales …
      RESULTS KEEP <filename>

db2inspf <filename> <outfile>
```

The output of the INSPECT tool needs to be formatted using a DB2 utility to see the results. The files are found in the db2dump directory. Sample output is illustrated in Example A-13.

Example A-13 Sample INSPECT output

```
DATABASE: SAMPLE
VERSION : SQL09053
2009-09-28-23.07.02.389882

Action: ROWCOMPESTIMATE TABLE
Schema name: SSCH
Table name: SSTRANS
Tablespace ID: 2  Object ID: 23
Result file name: compress.txt

    Table phase start (ID Signed: 23, Unsigned: 23; Tablespace ID: 2) :
SSCH.SSTRANS

      Data phase start. Object: 23  Tablespace: 2
      Row compression estimate results:
      Percentage of pages saved from compression: 54
      Percentage of bytes saved from compression: 54
      Compression dictionary size: 22016 bytes.
      Expansion dictionary size: 32768 bytes.
      Data phase end.
    Table phase end.
Processing has completed. 2009-09-28-23.07.03.734395
```

Benefits of data row compression

DB2 V9 row compression technology is capable of storage cost savings by up to 50% or more by reducing the amount of disk space (and disk sub-system peripherals) required for storing data. The size of database logs can also be reduced because DB2 compresses user data in log records.

This technology can also improve performance in certain scenarios, despite compression/decompression entailing CPU overhead. Accessing data from the disk is the slowest operation in a query or transaction. By storing compressed data on disk, fewer I/O operations need to be performed on the disk to retrieve or store the same amount of data. Therefore, for disk I/O-bound workloads (for instance, when the system is waiting/idling for data to be accessed from the disk), the query processing time can be noticeably improved.

DB2 keeps the data compressed on both disk and memory (DB2 buffer pools), reducing the memory consumed, freeing it for database or system operations. This improves database performance for queries and other operations.

Sample results

The amount of space savings using the data row compression feature in DB2 can vary depending on the data. Clients using beta versions of DB2 V9 have reported savings in excess of 50% and up to 80% for certain large database installations. For one client data set, a 179.9 GB table using 32 KB pages was reduced to only 42.5 GB, a savings of 76.4%.

Figure A-13 illustrates a few other examples of space-saving comparisons using the DB2 data row compression feature on different tables.

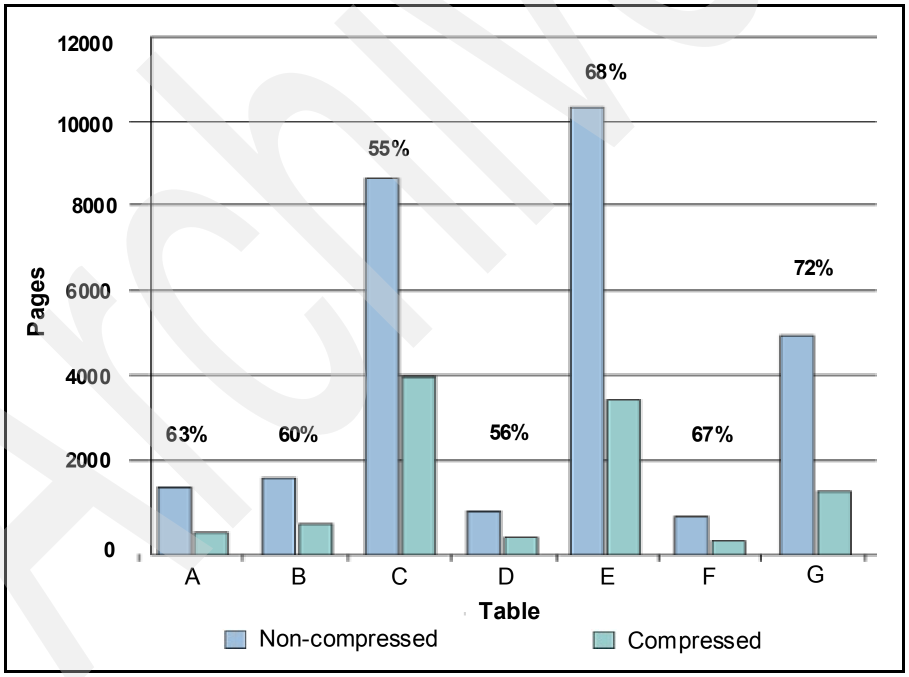


Figure A-13 Examples of space savings with DB2 data row compression

The following list details compression enhancements introduced in DB2 v9.7:

- XML data stored in tables.
- Temporary tables are automatically compressed if there is sufficient memory to build the compression dictionary and the DB2 optimizer determines that temporary table compression is worthwhile. This is based on estimated storage savings and the impact to query performance.
- Indexes can now be compressed in addition to the data. If data row compression is enabled on a table, indexes on the compressed tables are compressed by default (this feature can be explicitly enabled or disabled on an index by index basis).
- Data replication source tables can be compressed.

Other forms of compression in DB2

The following list details additional mechanisms for reducing storage requirements:

- NULL and default value compression

  This type of compression consumes no-disk storage for NULL values, zero length data in variable length columns and system default values.
- Database backup compression

  This compression feature results in smaller backup images that reduce backup storage requirements and make it easier to move backups between systems.
- Multidimensional clustering, which provides a form of index compression

  Significant index space savings can be achieved through block indexes, where one key (or index entry) per thousands of records is used (rather than one key per record with traditional indexes).

Self-tuning memory

Tuning database memory and buffers for optimum performance is effortless with the new self-tuning memory management feature in DB2 V9. It automatically configures database memory settings and adjusts them dynamically during runtime to optimize performance and improve administrator productivity.

Database workloads seldom remain static. Workloads and the environments in which they are run can change over time due to a number of factors, including more users, change in the pattern of queries, running of maintenance tasks, changes in resources consumed by other applications, and so on. Therefore, a system tuned by even the most skilled administrator at one point in time might not be optimal at another time. And the changes might occur in seconds (rather than days or weeks), giving the administrator little time to respond. Database memory settings are especially vulnerable to such changes and can severely impact response times.

The self-tuning memory feature in DB2 V9 simplifies the task of memory configuration by automatically setting values for several memory configuration parameters at startup. The self-tuning memory manager uses intelligent control and feedback mechanisms to keep track of changes in workload characteristics, memory consumption, and demand for the various shared resources in the database and dynamically adapts their memory usage as needed. For example, if more memory is needed for sort operations and buffer pools have excess memory, the memory tuner frees up the excess buffer pool memory and allocates it to the sort heaps. This is depicted in Figure A-14.

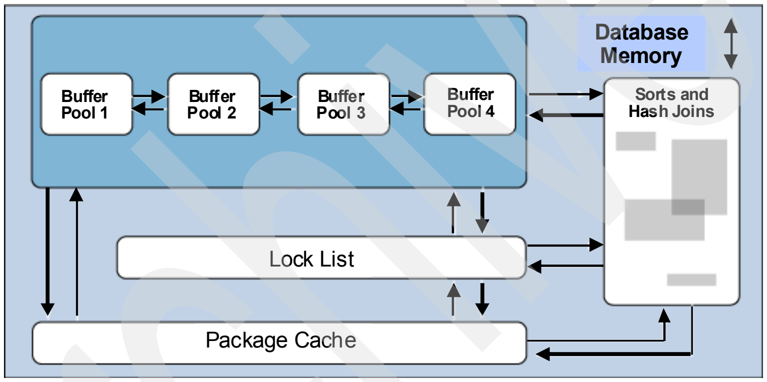


Figure A-14 Database memory

On Windows and AIX platforms, the self-tuning memory feature can also determine the overall database memory requirements and dynamically tune the total database memory usage. That is, on these platforms, in addition to dynamically adjusting the memory utilization between the database resources, DB2 can also consume more physical memory as the workload demands it and free up that memory to the operating system for other tasks and applications when database memory requirements are low.

Enabling self-tuning memory

Self-tuning memory works on the database shared-memory resources. These resources include:

- Buffer pools (controlled by the ALTER BUFFERPOOL and CREATE BUFFERPOOL statements)
- Package cache (controlled by the pckcachesz configuration parameter)
- Locking memory (controlled by the locklist and maxlocks configuration parameters)
- Sort memory (controlled by the sheapthres_shr and the sortheap configuration parameters)
- Total database shared memory (controlled by the database_memory configuration parameter)

The self-tuning memory is enabled using the self_tuning_mem database configuration parameter. In DB2 V9, self-tuning memory is automatically enabled (for non-partitioned databases) when a new database is created. That is, the self_tuning_mem is set to ON, and the database parameters for the resources listed are set to AUTOMATIC. For existing databases migrated from earlier versions of DB2, self-tuning memory needs to be enabled manually. Rather than have all of the database memory resources managed automatically, you can set the desired memory resources (parameters) to AUTOMATIC.

Traditionally, configuring database memory parameters for optimal operation can be a complex and time-consuming task. By setting the memory parameters for the database automatically, DB2 V9 simplifies the task of configuring the data server. This improves DBA productivity, freeing up the administrator to focus on other tasks.

In addition to simplifying memory configuration, this new, adaptive self-tuning memory feature improves performance by providing a superior configuration that is dynamic and responsive to significant changes in workload.

This feature can also be beneficial when there are several databases running on the same system, automatically allowing certain databases to consume more memory during peak times, and freeing it up for other databases and workloads when they need it more.

Self-tuning memory in action

The graphs in Figure A-15 show DB2 self-tuning memory in action for a demanding benchmark configuration. DB2 automatically increases the database shared memory (graph on the left) to satisfy the demands of the workload, resulting in a corresponding increase in query throughput (graph on the right).

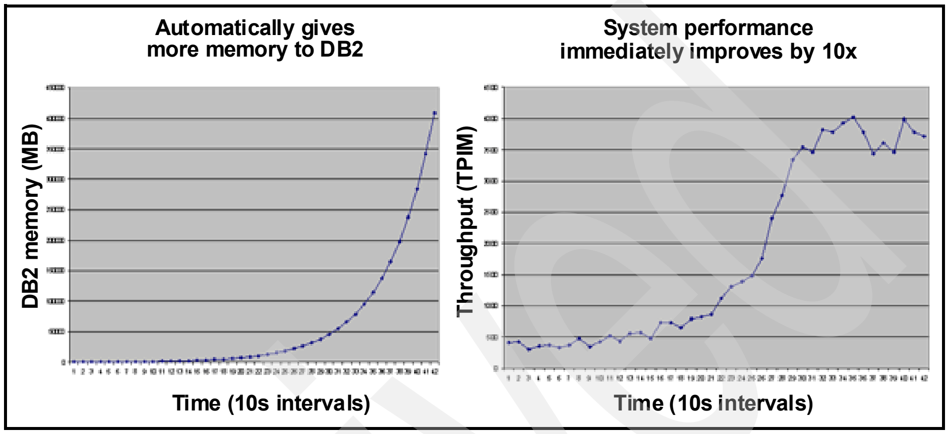


Figure A-15 Effects of automatic tuning on system performance

Self-tuning memory in DB2 9 is a revolutionary feature that can result in significant time savings, simplify memory tuning, and automatically and dynamically optimize performance.

DB2 Design Advisor

Many DBAs would agree that the decisions made with respect to the design of a database are the most challenging, time consuming, and critical to make. The DB2 Design Advisor assists DBAs in making optimal and comprehensive database design decisions. This self-configuring tool simplifies the design process by using workload, database, and hardware information to recommend specific performance acceleration options for routine design tasks.

Specifically, the Design Advisor assists with the following design tasks:

- Index selection
- MQT selection
- Multidimensional clustering selection
- The redistribution of tables

Design Advisor is a tool that can help you improve your workload performance. The task of selecting which indexes, clustering dimensions, or partitions to create for a complex workload can be daunting. Design Advisor identifies virtually all of the objects needed to improve the workload performance. Given a set of SQL statements in a workload, Design Advisor generates recommendations for the following areas:

- New indexes
- New MQTs
- Conversion to multidimensional clustering tables
- The redistribution of tables
- Deletion of objects unused by the specified workload

You can choose to have the Design Advisor GUI tool implement recommendations immediately or schedule them for a later time. Whether using the Design Advisor GUI or the command-line tool, Design Advisor can help simplify tasks. For example, if you are planning for or setting up a new database or partitioning structure.

Use Design Advisor to perform the following tasks:

- Generate design alternatives in a test environment for partitioning, indexes, MQTs, and MDC tables.
- Determine initial database partitioning before loading data into a database.
- Assist in migrating from a non-partitioned DB2 database to a partitioned DB2 database. Assist in migrating to DB2 in a partitioned environment from another database product.
- Evaluate indexes, MQTs, or partitions that have been generated manually.

After your database is set up, use Design Advisor to meet the following tuning goals:

- Improve performance of a particular statement or workload.
- Improve general database performance, using the performance of a sample workload as a gauge.
- Improve performance of the most frequently executed queries, for example, as identified by the Activity Monitor.
- Determine how to optimize the performance of a new key query.
- Respond to Health Center recommendations regarding shared memory utility or sort heap problems in a sort-intensive workload.
- Find objects that are not used in a workload.

Note: Both Cubing Services Optimization Advisor and DB2 Design Advisor make MQT recommendations. There is more information about these topics, in the remaining sections of this chapter, and in Chapter 6, “InfoSphere Warehouse Cubing Services” on page 169.

Collecting workload

Before using Design Advisor, determine how to get a representative sample of the workload to provide as input. This can be challenging due to the ad hoc nature of data warehouse workloads.

DB2 Workload Manager (WLM) can be used to capture activity details that can be used as a source for Design Advisor.

The command line version of Design Advisor provides a direct interface to the WLM activity data (see Example A-19 on page 568). For the GUI version the SQL statements must be exported to a file and imported into the GUI tool.

The WLM provides a number of different ways to select activities:

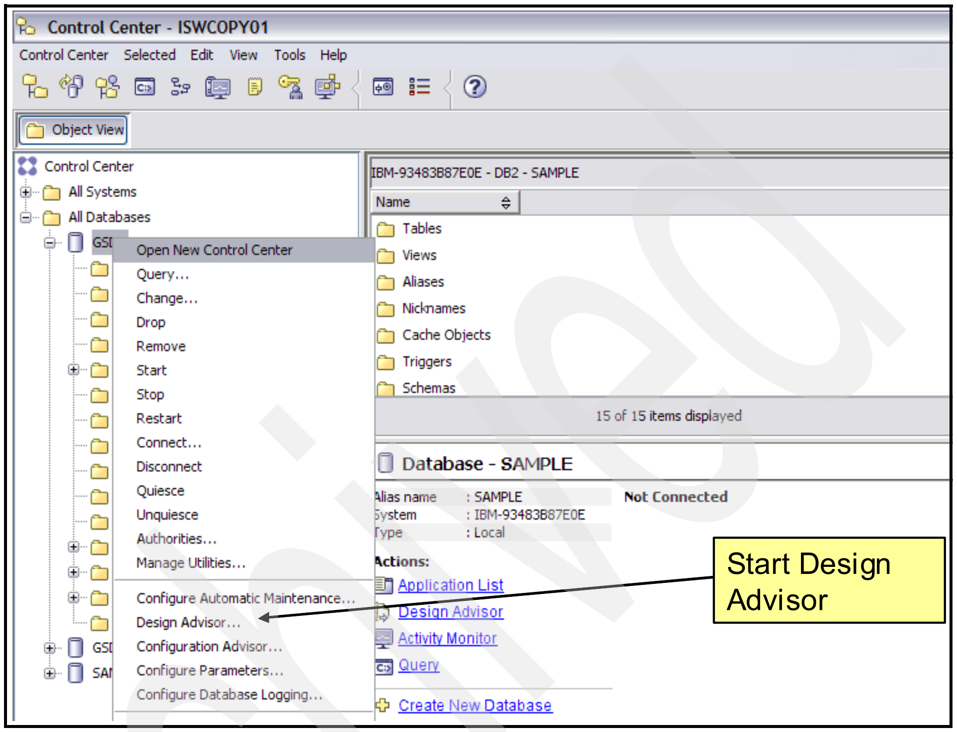
- Queries that exceed a resource use threshold
- Queries from a one or more specific applications
- Queries from one or more specific user groups or connection types

The selection of the workload should match the goals of the tuning exercise. The recommendations should be interpreted in the context of the workload provided to Design Advisor. You can include the entire set of queries, which could be in the 10,000 to 1 million range, but a set of queries ranging from 10s to 100s strikes the best balance between getting good results and consuming DB2 resources.

In this case, use a sample of queries that run on one Monday morning when there is a lot of business objects activity, and during one night when the bigger SAS jobs are running.

Describe the workload to Design Advisor

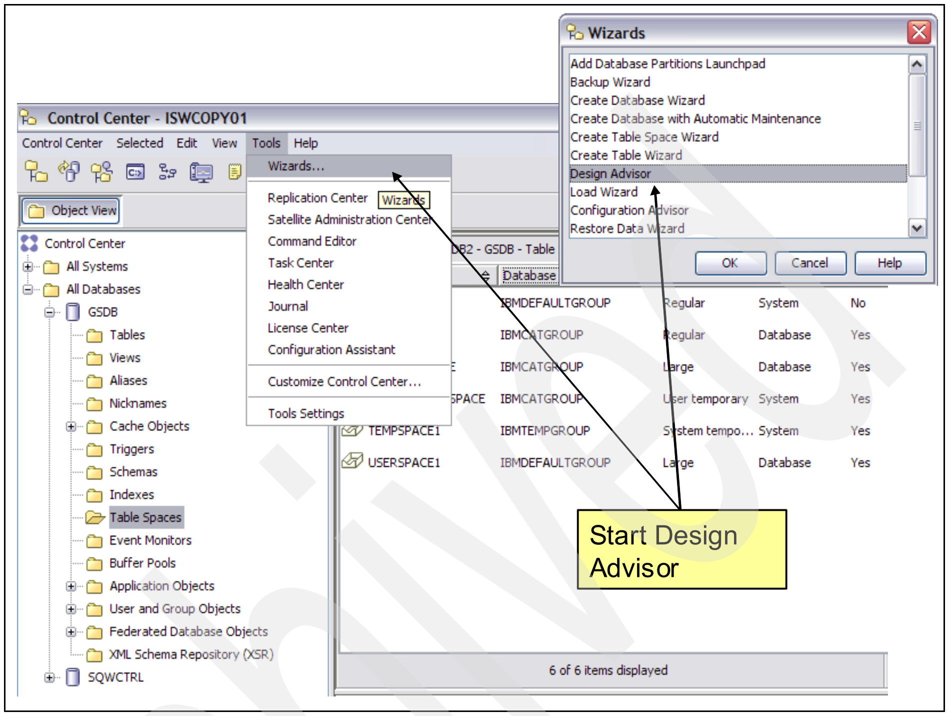
With the workload file in hand, launch the Design Advisor by right-clicking the database from the DB2 Control Center as shown in Figure A-16.



Alias name : SAMPLE
System : IBM-93483B87E0E
Type : Local
Not Connected

Figure A-16 Starting Design Advisor through database

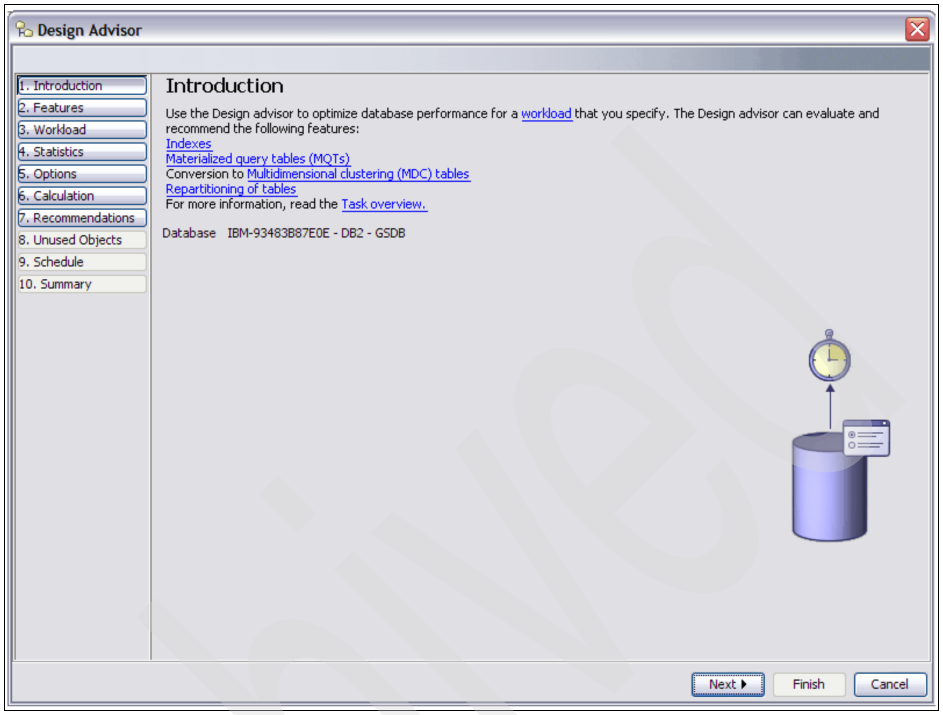
You can also launch it through a DB2 Control Center wizard, as shown in Figure A-17.



	IBMDEFAULTGROUP	Regular	System	No
	IBMCATGROUP	Regular	Database	Yes
E	IBMCATGROUP	Large	Database	Yes
SPACE	IBMCATGROUP	User temporary	System	Yes
TEMPSPACE1	IBMTEMPGROUP	System tempo...	System	Yes
USERSPACE1	IBMDEFAULTGROUP	Large	Database	Yes

Figure A-17 Starting Design Advisor through Control Center wizard

After being launched, an introduction window displays, as in Figure A-18.



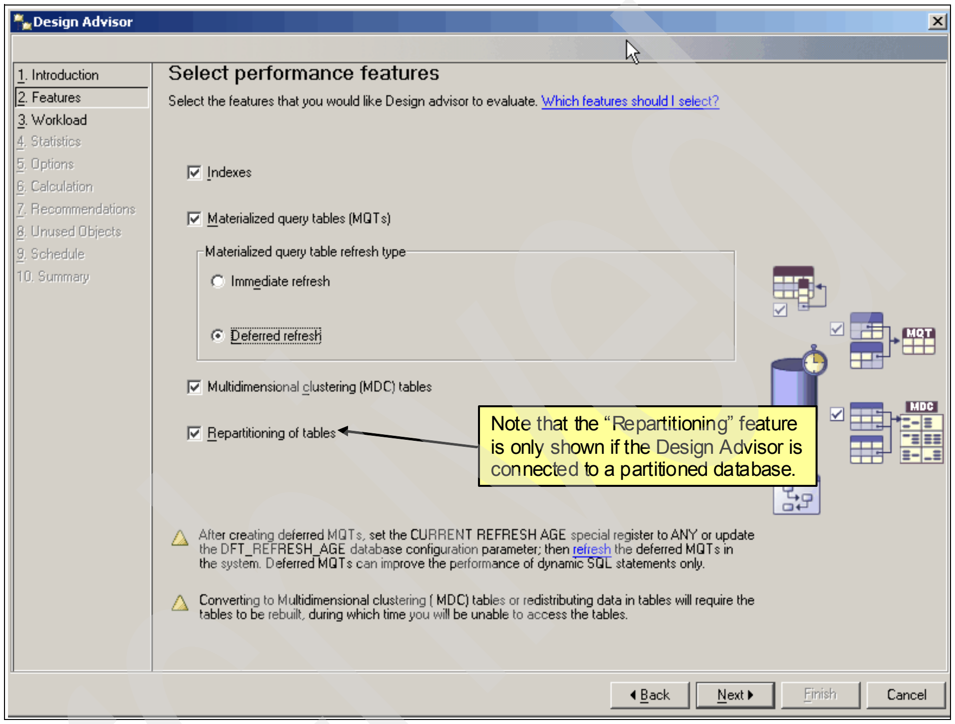
Introduction

Use the Design advisor to optimize database performance for a workload that you specify. The Design advisor can evaluate and recommend the following features:
Indexes
Materialized query tables (MQTs)
Conversion to Multidimensional clustering (MDC) tables
Repartitioning of tables
For more information, read the Task overview.

Database IBM-93483B87E0E - DB2 - GSDB

Figure A-18 Design Advisor Introduction window

Click **Next** and the Design Advisor allows you to select which features should be considered, such as indexes, MQT, MDC, and partitioning. This is depicted in Figure A-19. This example includes all of the options, which ensures that the Design Advisor finds a globally optimal solution by exploiting synergy among the features.



Select performance features

Select the features that you would like Design advisor to evaluate. Which features should I select?

- [✓] Indexes
- [✓] Materialized query tables (MQTs)

Materialized query table refresh type
- ( ) Immediate refresh
- (●) Deferred refresh

- [✓] Multidimensional clustering (MDC) tables
- [✓] Repartitioning of tables

Note that the “Repartitioning” feature is only shown if the Design Advisor is connected to a partitioned database.

After creating deferred MQTs, set the CURRENT REFRESH AGE special register to ANY or update the DFT_REFRESH_AGE database configuration parameter; then refresh the deferred MQTs in the system. Deferred MQTs can improve the performance of dynamic SQL statements only.

Converting to Multidimensional clustering (MDC) tables or redistributing data in tables will require the tables to be rebuilt, during which time you will be unable to access the tables.

Figure A-19 Design Advisor feature selection

Selecting all of the options can make it hard to understand why Design Advisor recommended what it did. Users might prefer to proceed step-by-step at the possible expense of an optimal solution. The following are examples of step-by-step approaches:

- Focus on a single application at a time — its workload and the parts of the schema that it accesses.
- Focus on optimizing access to one table at a time. That is, first identify queries that hit table X and implement only recommendations that pertain to that table. Repeat the process for subsequent tables.
- Focus on a particular feature. For example, focus only on redesigning BigFactTable using MDC and repartitioning. Leave MQTs for a future iteration, perhaps after validating the initial changes in production.

- Consider MQTs with deferred refresh. While this example uses immediate refresh, deferred refresh MQTs are generally more compatible with ETL activity.

Command line Design Advisor (db2advis)

Design Advisor can also be invoked through the command line instead of the GUI, as in Example A-14.

Example A-14   db2advis using the command line

```
db2advis –d TPCD –i tpch_queries.in –m IMCP –k LOW  -l 700 –c
DB2ADVIS_TBSP -f
```

The command line version of Design Advisor provide the following advantages:

- -m IMCP

  This specifies that Design Advisor should consider new indexes (I), new MQTs (M), converting standard tables to MDC tables (C), and repartitioning existing tables (P). The default is indexes only.

- -k LOW

  This specifies to compress the workload to a low degree. As a result, the Design Advisor analyzes a larger set of the workload provided. The default is medium.

- -l 700

  This specifies that any new indexes, MQTs, and so on consume no more than 700 MB. The default is 20% of the total database size.

- -c DB2ADVIS_TBSP

  This specifies a tablespace called DB2ADVIS_TBSP to use as a temporary workspace for generating MQT recommendations. This option is required if you want MQT recommendations and you are running on a DPF (multiple partition) instance. Otherwise, this parameter is optional.

Another useful option is -o output_file. This saves the script to create the recommended objects in a file.

The recommendations appear in the following order:

1. Base tables that include MDC or partitioning recommendations
2. MQT recommendations (first new ones, then existing ones to keep, and finally unused ones)
3. New clustering indexes (if any)
4. Index recommendations (new, keep, unused)

Recommendations for changing one table are shown in Example A-15.

Example A-15 db2advis recommendations

```
-- CREATE TABLE "TPCD"."LINEITEM" ("L_ORDERKEY BIGINT NOT NULL,
-- "L_PART" INTEGER NOT NULL,
-- "L_SUPPKEY" INTEGER NOT NULL,
-- "L_LINENUMBER" INTEGER NOT NULL,
-- "L_SHIPINSTRUCT" CHAR(25)  NOT NULL,
      (11 other columns omitted from this example)
-- MDC409022109290000 GENERATED ALWAYS AS ( ((INT(L_SHIPDATE))/7) )
-- ----PARTITIONING KEY ("L_PARTKEY") USING HASHING
-- ----IN "TPCDLADT"
-- ORGANIZE BY (
-- MDC409022109290000,
-- L_SHIPINSTRUCT )
-- PARTITIONING KEY (L_ORDERKEY) USING HASHING
-- IN TPCDLDAT
--;
-- COMMIT WORK ;
```

Anew partitioning key is recommended (L_ORDERKEY) to replace the current one (L_PARTKEY), which is commented out. The MDC recommendation for this table (ORGANIZE BY clause) includes two dimensions, a generated column (INT(L_SHIPDATE/7) and an existing column (L_SHIPINSTRUCT). Next in the output are recommendations related to MQTs, as shown in Example A-16.

Example A-16 Recommended MQTs

```
-- LIST OF RECOMMENDED MQTs
-- ========================
-- MQT MQT40902204140000 can be created as a refresh immediate MQT
-- mqt[1],   0.009MB
CREATE SUMMARY TABLE "ADVDEMO2"." MQT40902204140000"
    AS (SELECT Q6.C0 AS "C0", Q6.C1 AS "C1", …additional details
omitted here…)
    DATA INITIALLY DEFERRED REFRESH IMMEDIATE PARTITIONING KEY (C8)
    USING HASHING IN TPCDLDAT ;
COMMIT WORK;
REFRESH TABLE "ADVDEMO2"." MQT40902204140000";
COMMIT WORK;
RUNSTATS ON TABLE "ADVDEMO2"." MQT40902204140000";
COMMIT WORK;
-- MQT MQT409022041530000 can be created as a refresh immediate MQT
(… DDL to create this table follows…)
```

The MQT recommendations include estimated size, tablespace to use, partitioning key (if applicable), type of refresh (immediate or deferred), and whether the table is a replica of a base table (indicated by the REPLICATE keyword), which it is not in this case.

Using db2advis has the advantage that it can select its workload directly from the monitor output of the DB2 WLM.

To achieve this, the following prerequisites must be satisfied:

- An activity event monitor table must exist. You cannot import information from activity event monitor files.
- Activities must have been collected using the COLLECT ACTIVITY DATA WITH DETAILS or COLLECT ACTIVITY DATA WITH DETAILS AND VALUES options.
- Explain tables must exist. You can use the EXPLAIN.DDL script in the sqllib/misc directory to create the explain tables.

The activity event monitor can be created with the command shown in Example A-17.

Example A-17 Creating the Activity Event Monitor

```
CREATE EVENT MONITOR DB2ACTIVITIES
    FOR ACTIVITIES
    WRITE TO TABLE
    ACTIVITY (TABLE ACTIVITY_DB2ACTIVITIES
              IN USERSPACE1
              PCTDEACTIVATE 100),
    ACTIVITYSTMT (TABLE ACTIVITYSTMT_DB2ACTIVITIES
                  IN USERSPACE1
                  PCTDEACTIVATE 100),
    ACTIVITYVALS (TABLE ACTIVITYVALS_DB2ACTIVITIES
                  IN USERSPACE1
                  PCTDEACTIVATE 100),
    CONTROL (TABLE CONTROL_DB2ACTIVITIES
             IN USERSPACE1
             PCTDEACTIVATE 100)
    AUTOSTART;
```

The DB2 WLM is configured to capture activities. The code snippet in Example A-18 captures activities at the Workload level.

Example A-18 Configuring WLM to capture activity details.

```
create workload QUERIES applname('db2bp.exe') service class
DS_HIGH_PRI_SUBCLASS under DS_AUTO_MGMT_SUPER collect activity data
with details
```

You can import activity information into Design Advisor by using the db2advis command as shown in Example A-19

Example A-19 db2advis interface with DB2 WLM

```
db2advis -d gsdb -wlm db2activities wl queries
```

Example A-19 shows how to import activities collected by the DB2ACTIVITIES activity event monitor for the QUERIES workload in the GSDB database into design advisor. You can specify the workload or service class name, and the start time and end time.

High availability

The DB2 UDB architecture offers high availability of the stored data through the support of clustered takeovers. In the event of a machine failure, the activities of the failed machine can be taken over by other components of the cluster. With DB2 UDB with DPF, mutual takeover of machines in a cluster is supported. To the user and application developer, the database still appears as a single database on a single computer.

DB2 UDB high-availability cluster support is provided for the Sun, AIX, HPUX, Linux, and Windows platforms. High availability with clustering is provided as indicated in Table A-6.

Table A-6 High availability with clustering support

Platform	**Server**
Windows	Microsoft Cluster Server
Sun Solaris	► Veritas Cluster Server ► Sun Cluster
IBM AIX	HACMP™
HP-UX	HP Service Guard
Linux	► Tivoli System Automation ► Veritas Cluster Server

Standard high-performance mainstream products (such as IBM pSeries® or Sun Enterprise Servers) provide the infrastructure to tie together redundant hardware components with features such as data service monitoring and automatic failover. The cluster software provides transparent takeover of the disk and communications resources of the failed node. It invokes the DB2 UDB failover support to handle the restart of the database (or the partition) from the failed node.

In a DB2 UDB with DPF environment where two system nodes are paired together, and each pair has access to shared disks (disks connected to two nodes), if one of the nodes in a pair fails, the other nodes can take over and the system continues to operate (an active-active failover scenario) and guarantees that the failover workload can be accommodated. While this method provides quick takeover of a failed node, there might be an impact on performance due to an increased load on the takeover node. An alternative to DB2 UDB with DPF mutual takeover is to have the paired node remain idle until the takeover is required. This preserves the overall system performance (an active-passive failover scenario).

An additional feature of failover support is that only queries or connections that require access to the nodes taking part in the failover realize that a failover is in progress. Thus, other workloads are unaffected when resources are moved and recovery procedures take place.

DB2 can also exploit Windows, Linux, SUN, or AIX clustering for scalability and improved performance. To the user and application developer, the database still appears as a single database on a single computer. This provides several benefits:

- DB2 UDB with DPF clustering enables an application to use a database that is too large for a single computer to handle efficiently.
- DB2 UDB with DPF partitioning avoids system limitations to exploit large SMP machines.
- DB2 UDB with DPF takes advantage of cluster, SMP, and MPP systems to provide added performance for complex queries. This is accomplished by spreading the database across multiple servers in a cluster, multiple nodes on an MPP, or multiple partitions in an SMP (or combinations of these scenarios). Individual tables can be spread across differing numbers of nodes, providing many tools for optimizing performance. Indexes for tables and associated logs are located with the tables so that cross-node access is minimal. The distribution of data across these nodes is transparent to the user, who sees complete tables and databases.

- DB2 automatically runs multiple transactions (SQL statements) in parallel by dispatching them to the multiple partitions. No keywords are needed. DB2 UDB can also automatically execute a single query (SQL statement) in parallel by breaking the query into sub-tasks and dispatching each sub-task to a different node. SELECT, INSERT, UPDATE, and DELETE statements are all executed in parallel. The key to efficient parallel processing across nodes is intelligent partitioning and parallel optimization. DB2 automatically partitions the data across the nodes such that the optimizer knows where each row is located and is able to dispatch the processing to the node where the data is located. This minimizes the movement of data between nodes.

  Parallelism is used during the sort phase of the SELECT statement, because sorts are local to the node and also use parallel I/O when appropriate. Parallelism is used during the sort-merge phase, where logically possible, because the optimizer attempts to push processing down to the individual partitions and nodes. The shared-nothing architecture is the best approach for ensuring scalability across multiple nodes, particularly when large databases of multiple terabytes are involved.
- DB2 UDB applications can be ported to systems ranging from XP workstations, UNIX/Linux Servers, and UNIX/Linux/Windows clusters, providing for a wide range of configuration flexibility and processor scalability options. As processing requirements and databases increase, the database manager and server can grow to meet these new requirements.

Workload management

DB2 WLM was introduced with DB2 for LUW version 9.5 and enhanced with version 9.7. It is also available with InfoSphere Warehouse versions 9.5 and later.

A web-based tool was introduced with InfoSphere Warehouse version 9.7 to simplify the development and deployment of a workload management environment for data warehousing.

See Chapter 10, “Workload management” on page 427 for more details about the web-based tool and the underlying DB2 workload management concepts.

B

DB2 for z/OS: Support for data warehousing

In this appendix we provide an overview of the functions and capabilities in DB2 for z/OS that support data warehousing and make it a powerful platform for InfoSphere Data Warehouse and infrastructure for business intelligence (BI).

Functions and capabilities of DB2 for z/OS

In the following sections we discuss and describe the key functions and capabilities of z/OS, System z, and DB2 for z/OS that enable a powerful solution environment for InfoSphere Data Warehouse.

z/Architecture and z/OS

z/OS and the IBM System z10®, System z9®, and zSeries® 890, and 990 systems offer an architecture that provides qualities of service that are critical for for the data warehousing environment.

z/OS, which is highly secure, scalable, and open, offers high-performance that supports a diverse application execution environment. The tight integration that DB2 has with the System z architecture and the z/OS environment creates a synergy that allows DB2 to exploit advanced z/OS functions.

The z/OS operating system is based on 64-bit z/Architecture®. The robustness of z/OS powers the most advanced features of the IBM System z10 and IBM System z9 technology and the IBM eServer™ zSeries 990 (z990), 890 (z890), and servers, enabling you to manage unpredictable business workloads.

DB2 gains a tremendous benefit from z/Architecture. The architecture of DB2 for z/OS takes advantage of the key z/Architecture benefit of 64-bit virtual addressing support which provides it with an immediate scalability benefit.

z/Architecture features that benefit DB2

The following z/Architecture features provide significant value for DB2.

- 64-bit storage

  Increased capacity of central memory from 2 GB to 16 exabytes eliminates most storage constraints. The 64-bit storage also allows for 16 exabytes of virtual address space, a huge step in the continuing evolution of increased virtual storage. In addition to improving DB2 performance, 64-bit storage improves availability and scalability, and it simplifies storage management.

- High-speed communication

  HiperSockets™ enable high-speed TCP/IP communication across partitions of the same System z server. For example, between Linux on System z and DB2 for z/OS.

- Dynamic workload management

  One of the strengths of the System z platform and the z/OS operating system is the ability to run multiple workloads at the same time in one z/OS image or across multiple images. The function that makes this possible is called dynamic workload management, which is implemented in the Workload Manager (WLM) component of the z/OS operating system. The idea of z/OS WLM is to make a contract between the applications and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes, and defines goals for them that express the expectation of how the work should perform. WLM uses these goal definitions to manage the work across all systems of a Sysplex environment.

  The evolution of data warehousing has spawned a diverse set of workloads, each having unique service level requirements. Examples include Operational BI (or tactical queries), analytics, scheduled reporting, refresh processing, and data mining. Additionally, the user base has grown beyond senior executives, mid-level management and business analysts to include customer-facing personnel, such as service representatives. The coexistence of this mixed workload and how to distribute resources has been identified as one of the most important concerns in data warehouse design.

  WLM makes a contract between the installation (performance administrator) and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes. The installation defines business importance and goals for the service classes. WLM uses these definitions to manage the work across all systems of a Sysplex environment. WLM adjusts dispatch priorities and resource allocations to meet the goals of the service class definitions. It does this in order of the importance specified, highest first. Resources include processors, memory, and I/O processing.

  The Intelligent Resource Director (IRD) allows you to group logical partitions that are resident on the same physical server, and in the same Sysplex, into an LPAR cluster. This gives WLM the ability to manage resources across the entire cluster of logical partitions.

- Faster processors

  With more powerful, faster processors, along with the System z Integrated Information Processor (zIIP) specialty engine, DB2 achieves higher degrees of query parallelism and higher levels of transaction throughput. The zIIP is designed to improve resource optimization and lower the cost of eligible workloads, enhancing the role of the mainframe as the data hub of the enterprise.

The database

At the heart of any data warehouse, before you can even consider running a BI application, you need a database management system (DBMS) that uses a relational database management system (RDBMS). Fortunately, System z has DB2 for z/OS. In the beginning of data warehousing, back when it was still referred to a decision support, DB2 for z/OS was at its center. In fact, throughout DB2's long history, it has always managed to deliver product enhancements that championed decision support. In an answer to the changing database landscape of today's challenging warehousing world, IBM is delivering even more significant DB2 for z/OS enhancements than ever before in direct support for data warehousing and BI. The last few DB2 releases have been rich with capabilities to improve your data warehouse and BI experience.

DB2 has a renewed presence in the data warehouse world, because data warehousing is changing. Rather than determining what happened in the past, clients want to use all their information to make immediate decisions. And, instead of allowing only a few to access the valuable data being kept in the data warehouse, today it is being used by an ever growing number of users. The focus is on getting the right data to the right person, at the right time. The data must be available rapidly and be accurate, so it can easily be used to provide valuable information. DB2 for z/OS is uniquely positioned to satisfy these modern data warehouse challenges.

DB2 impact on your data warehouse

DB2 for z/OS has been supporting data warehousing for more than 25 years. It has continually delivered features and functions in direct or indirect support of data warehousing and the associated BI applications. The following list details the more significant DB2 features that can enhance your data warehousing experience:

- Resource Limit Facility (RLF)

  Introduced DB2 V2.1, this allows for the control of the amount of CPU resource that a task, in this case a query, can actually use. RLF affects dynamic SQL, which can comprise a significant portion of the data warehouse SQL workload. As examples, it can be critical in controlling system resources, and can help you control the degree of parallelism obtained by a query.

- Hardware assisted data compression

  Delivered with DB2 V3, this still has a major and immediate effect on data warehousing. Enabling compression for table spaces can yield significant disk savings. In testing, numbers as high as 80% have been observed.

- I/O parallelism, CP parallelism, and Sysplex query parallelism

  These features became available in DB2 Version 3, Version 4, and Version 5 respectively. With the first iteration multiple I/O could be started in parallel to satisfy a read request. Next, a query could run across two or more CPs. For example, a query could be segmented into multiple parts, and each part could run against its own Service Request Block (SRB), performing its own I/O. With the delivery of data sharing, a query can run across multiple CPs on multiple Central Electronic Complexes (CECs) in the parallel Sysplex.

  There is additional CPU used for setup when DB2 first decides to run a query in parallel. There is a correlation between the degree of parallelism achieved and the elapsed time reduction. There are also DSNZPARMs and bind parameters that need to be set before parallelism can be used.

- Data sharing

  This was delivered along with CP parallelism in DB2 Version 4. High availability for data warehousing has now become the norm rather than the exception, and data sharing is capable of giving data warehousing that kind of high availability. DB2 data sharing allows access to the operational data by the data warehouse and analytics, yet still lets you separate those applications into their own DB2, reducing the chances of the data warehouse activity impacting operational transactions.

- Partitioning

  The large volume of data stored in data warehouse environments can introduce challenges to database management and query performance. The table space partitioning feature of DB2 for z/OS currently has the following characteristics to aid in addressing those challenges:

  – Maximize availability or minimize run time for specific queries by allowing queries and utilities to work at the partition level.

  – Grow to 4096 partitions, with a partition being a separate physical data set.

  – Allow loading and refreshing activities, including the extraction, cleansing, and transformation of data, in a fixed operational window.

  – Increase parallelism for queries and utilities. Parallelism can be maximized by running parallel work across multiple partitions.

  – Accommodate data growth. A universal table space can grow automatically up to 128 TB and has the functionality of segmented table spaces while retaining the size and partition independence allowed by a partitioned table space.

  – Perform data recovery or restoration at the partition level if data should become damaged or otherwise unavailable, improving availability and reducing elapsed time.

- Compression

  DB2 compression is specified at the tablespace level. It is based on the Lempel-Ziv lossless compression algorithm, uses a dictionary, and is assisted by the System z hardware. Compressed data is also carried through into the buffer pools. This means compression could have a positive effect on reducing the amount of logging you do because the compressed information is carried into the logs. This reduces your active log size and the amount of archive log space needed.

  Compression also can improve your buffer pool hit ratios. With more rows in a single page after compression, fewer pages need to be brought into the buffer pool to satisfy a query get page request. An additional advantage of DB2 hardware compression is the hard speed. As hardware processor speeds increase, so does the speed of the compression built into the hardware chipset.

  When implementing a data warehouse, the growth in size can become problematic, regardless of the platform. DB2 hardware compression can help address that issue by reducing the amount of disk needed to fulfill your data warehouse storage requirements.

- Parallelism

  One method of reducing the elapsed time of a long-running query is to segment that query across multiple processors. This is exactly what DB2 parallelism does. Parallelism allows a query to run across multiple CPs. A query is segmented into multiple parts, with each part running under its own SRB, and performing its own I/O. Although there is additional CPU used when DB2 decides to take advantage of query parallelism for its setup, there is a close correlation between the degree of parallelism achieved and the time reduction for the query. There also are DSNZPARMs and bind parameters that need to be set before parallelism can be used.

- Star Schema

  There is a specialized case of the use of parallelism, called a star schema. That is the way a relational database represents multi-dimensional data, which is often a requirement for data warehousing applications. A star schema is usually a large fact table with a number of smaller dimension tables.

  For example, you can have a fact table for sales data. The dimension tables could represent products that were sold, the stores where those products were sold, the date the sale occurred, any promotional data associated with the sale, and the employee responsible for the sale. Using star joins in DB2 requires enabling the feature through a DSNZPARM keyword.

How does DB2 for z/OS Version 8 help?

Many features and enhancements in DB2 V8 could directly impact a data warehouse implementation. The following list detail the features and enhancements that can have a positive impact on your data warehouse application:

- Clustering decoupled from partitioning
- Indexes created as deferred are ignored by DB2 optimizer
- Fast cached SQL invalidation
- Automatic space management
- Statements IDs of cached statements as input to EXPLAIN
- Long-running, non-committing reader alerts
- Check In (CI) size larger than 4KB
- Multi-row INSERT/FETCH
- REOPT(ONCE) to reduce host variables impact on access paths
- Index-only access for VARCHAR columns
- Backward index scan
- Distributed Data Facility (DDF) performance enhancements
- Up to 4,096 partitions
- Longer table and column names
- SQL statements up to 2 MB
- Sparse index for star join
- More tables in join
- Common table expressions
- Recursive SQL
- Indexable unlike types
- Materialized Query Table (MQT).

Now we can look those features in more detail:

- Backward index scan

  Indexes are a huge performance asset to a data warehouse. Having the ability to read an index backwards lets you avoid building both an ascending index and a descending index structure. Reducing the number of indexes improves the cost of performing inserts and deletes. Every insert or delete must update every index on the table being changed. Backward index scanning also can reduce the cost of performing updates if the update occurs against a column participating in the index. In addition, backward index scans reduce the volume of disk storage required to build all those indexes. With a backward index scan, you need to use only one index, where you previously needed two.

- Multi-row FETCH and INSERT

  This enhancement lets you read or insert multiple rows with a single SQL statement through an array. It reduces the CPU cost of running a FETCH or INSERT. This feature is completely usable in distributed applications processing using Open Database Connectivity (ODBC) with arrays and dynamic SQL. This also offers significant performance advantages. As an example, it could increase the performance of FETCH processing by 50% and INSERT processing by 20%. In fact, in client testing, this feature averaged 76% improvement for FETCH and 20% improvement for INSERT. Improving INSERT performance and reducing INSERT CPU consumption could be a significant help to your Extract, Transform, Load (ETL) processing.

- Indexable unlike types

  Mismatched data types can now be stage 1 and indexable. This is a huge advantage for those applications that do not support all data types available in DB2 for z/OS.

- Sparse index in memory work files

  For star join processing, sparse indexes can use many work files. DB2 V8 attempts to put these work files in memory, which can result in a significant performance improvement for data warehouse queries that use star joins.

- More partitions (4,096) and automatic space management

  Growth is one of those foregone conclusions of a data warehouse. That is, it is something you just know is going to happen. With 4096 partitions, a data warehouse could grow to 16 TB for a 4 KB page size or 128 TB for 32 KB page. And this is for just one table. DB2 V8 also gives automatic space management, the ability to let DB2 manage primary and secondary space allocations. With DSNZPARM MGEXTSZ activated, DB2 manages the allocation of tablespace extents, ensuring that the tablespace can grow to its maximum size without running out of extents.

Benefit of MQTs

An MQT is a DB2 table that contains the results of a query, along with the definition of the query. It can be thought of as a materialized view or automatic summary table that is based on an underlying table or set of tables. These underlying tables are referred to as the base tables. MQTs are a powerful way to improve response time for complex SQL queries, especially for queries that involve the following situations:

- A commonly accessed subset of rows
- Joined and aggregated data over a set of base tables
- Aggregated or summarized data that covers one or more subject areas

MQTs can effectively eliminate overlapping work among queries by performing the computation once when the MQTs are built and refreshed, and reusing their content for many queries. In many workloads, users frequently issue queries over similar sets of large volume data. Moreover, this data is often aggregated along similar dimensions (for example, time, region). Though MQTs can be directly specified in a user query, their real power comes from the query optimizer's ability to recognize the existence of an appropriate MQT implicitly, and to rewrite the user query to use that MQT. The query accesses the MQT (instead of accessing one or more of the specified base tables), and that shortcut can drastically minimize the amount of data read and processed.

For example, suppose that you have a large table named SALES that contains one row for each transaction that gets processed. You want to compute the total transaction revenue along the time dimension. Although the table contains many columns, you are most interested in these columns:

- YEAR, MONTH, and DAY, which represents the date of a transaction
- REVENUE, which represents the revenue gained from the transaction

To total the amount of all transactions between 2001 and 2008 by year, you would use the query in Example B-1.

Example B-1 Query example

```
   SELECT
   YEAR, SUM (AMOUNT)
   FROM TRANS
   WHERE
   YEAR >= '2001' AND YEAR <= '2008'
   GROUP BY YEAR
ORDER BY YEAR;
```

This query might be expensive to run, particularly if the TRANS table is a large table with millions of rows and many columns. Suppose that you define a system-maintained MQT that contains one row for each day of each month and year in the TRANS table. Using the automatic query rewrite process, DB2 could rewrite the original query into a new query that uses the MQT instead of the original base table TRANS. The performance benefits of the MQT increase as the number of queries that can consume the MQT increase. However, users must understand the associated maintenance cost of ensuring the proper data currency in these MQTs. Therefore, the creation of effective and efficient MQTs is both a science and an art.

How does DB2 9 for z/OS help?

DB2 9 for z/OS delivers more changes that will directly impact your data warehouse and application analytics, including the following:

- New row internal structure for faster VARCHAR processing
- Fast delete of all the rows in a partition (TRUNCATE)
- Deleting first n rows
- Skipping uncommitted inserted/updated qualifying rows
- Index on expression
- Dynamic index ANDing
- Reduce temporary tables materialization
- Generalizing sparse index/in-memory data caching
- Clustering decoupled from partitioning
- Indexes created as deferred are ignored by DB2 optimizer
- Fast cached SQL invalidation
- Statements IDs of cached statements as input to EXPLAIN
- Universal tablespaces
- Partition-by-growth to remove non-partitioned tablespace size limit
- Implicit objects creation
- Clone tables
- MERGE statement
- Identifying unused indexes
- Simulating indexes in EXPLAIN (Optimization Service Center)
- More autonomic buffer pools tuning for WLM synergy
- Resource Limit Facility (RLF) support for end-user correlation
- RANK, DENSE_RANK, and ROW_NUMBER
- EXCEPT, and INTERSECT
- pureXML®

Now we can look at those features in more detail:

- Universal tablespace

  This is a key DB2 enhancement in support of data warehousing. Consider the sometimes unpredictable but expected growth of a data warehouse and the high possibility that many tables could be frequently refreshed. A universal tablespace is a cross between a partitioned tablespace and a segmented tablespace, giving you many of the best features of both. When using a universal tablespace, you get the size and growth of partitioning while retaining the space management, mass delete performance, and insert performance of a segmented tablespace. It is similar to having a segmented tablespace that can grow to a 128 TB of data, assuming the right DSSIZE and right number of partitions are specified, and that also gives you partition independence.

- Index compression

  The first line of defense against data warehouse performance problems, after a well-written query, is creating indexes. With data warehousing, and even for certain types of OLTP, it is possible to use as much, if not more, disk space for indexes than for the data. DB2 9 for z/OS index compression can make a huge difference when it comes to saving disk space. The implementation of index compression is different from data compression. As examples, it does not use a dictionary and there is no hardware assist. Actually, the lack of a dictionary could be a plus. With no dictionary, there is no need to run the REORG or LOAD utilities before compressing index data. When compression is turned on for an index, key and Record Identifier (RID) compression immediately begins.

- Randomized index key

  You want your keys to be spread throughout the index with no hotspots, a task sometimes easier to explain than accomplish. Sometimes things just end up in an incorrect place, no matter how hard you try to pick a key that spreads the data around. Enter DB2 9 for z/OS with a new option for the CREATE and ALTER INDEX SQL statements. Previously, you could specify ascending for forward index scans or descending for backward index scans on each column in an index key. In DB2 9, you have the additional choice of specifying RANDOM for an index column. This causes the index entries to be put in random order by that column. Randomly inserting keys could reduce the chances of contention caused by ascending index inserts or index hot spots. However, indexes created with the RANDOM option do not support range scans.

  You cannot use RANDOM in the following are instances:

  – The key column is VARCHAR and the index uses the NOT PADDED option
  – The index was created with the GENERATE KEY USING clause
  – The index is part of the partitioning key

  The RANDOM clause on an index is available only after you have upgraded your DB2 9 subsystem to New Function Mode (NFM).

- Index usage tracking

  This enables an easy way to determine if an index is no longer being used. If you know there has been no access against a particular index, it would make the decision to delete the index much easier. Therefore, by cleaning up the DB2 catalog (deleting indexes that are no longer used) you can avoid performing unnecessary updates and deletes to that index. This can be especially important for a data warehouse that frequently refreshes its tables. By removing unused indexes, DB2 avoids updating those indexes during INSERT or LOAD processing.

DB2 9 for z/OS added the column LASTUSED to the Real-Time Statistics (RTS) table SYSIBM.SYSINDEXSPACESTATS. LASTUSED contains the date when the index, described by this row in the catalog table, was last used for a SELECT, FETCH, searched UPDATE, searched DELETE, or used to enforce a referential integrity constraint. This column is not updated when the index is changed because of an INSERT or UPDATE. LASTUSED defaults to a value of 1/1/0001. So, if you find an index with the default date, that index has never been used since moving to DB2 9.

Wait a while after upgrading to DB2 9 before taking advantage of this feature to make sure the column gets the opportunity to get updated.

Also, there is an APAR, a somewhat older fix (hiper), APAR PK44579 (maintain SYSIBM.SYSINDEXSPACESTATS.LASTUSED for all cases) that you need to apply if you plan to take advantage of this column.

The tables for RTS are now part of the DB2 catalog.

Index usage tracking is available only after you have upgraded your DB2 9 subsystem to NFM.

- SQL

  Two new SQL statements, MERGE and TRUNCATE, can be effective when used with a data warehouse.

  MERGE, sometimes called *upsert*, lets you change data without needing to know if the row already exists. With MERGE, if an existing row is found, it is updated. If it does not find a row, it performs an INSERT. No more performing a SELECT first to see if the row exists or doing an INSERT or UPDATE and checking to see if it failed.

  TRUNCATE is an easy way to remove all the rows from a table with a single SQL statement. It is useful if you are using DELETE triggers. Also, there is now an APPEND option on the CREATE/ALTER table that tells DB2 to ignore clustering during INSERT processing. This should improve INSERT performance by eliminating the need to determine where the row should go and just placing it at the end of the table.

- APPEND clause

  A data warehouse can have a tremendous amount of SQL INSERT and online LOAD utility processing. Normally, DB2 makes every attempt to place a row in the correct place in a table. But to accomplish this, it must perform a search. The more it has to search, the more expensive the INSERT or LOAD process becomes. Data warehouses might refresh one or more tables using INSERT or LOAD on a nightly basis, which is an expensive, time-consuming operation.

DB2 9 provides a solution. DB2 has added an APPEND clause to the CREATE and ALTER TABLE SQL statements. If APPEND is set to YES, DB2 places a new row at the end of the table, ignoring any form of clustering defined to that table. The end of a table can be defined several ways, depending on the type of object being accessed. As examples:

– For a segmented table space, it is placed at the end of table.

– For partitioned table spaces and partition by range universal table spaces, the row is placed at the end of the appropriate partition.

– For partition by range universal table spaces, the row is placed in any partition with space at the end.

The APPEND clause cannot be used with tables created in LOB, XML, and workfile table spaces. It also is ignored by the REORG utility, so the REORG can be used to maintain cluster order if that order is necessary. The APPEND clause is available after you have upgraded the DB2 9 subsystem to NFM.

- Clone tables

  Tables can often be completely replaced on a weekly, even daily, basis in a data warehouse. Replacing a table can cause an outage, even if that outage seems short. Clone table support in DB2 9 gives you an easy way to create a replacement table while still accessing the original table and using a command to switch the table the SQL accesses. The concept is similar to an online LOAD REPLACE, something unavailable in DB2.

- RANK

  DB2 9 gives Online Analytical Processing (OLAP) functionality with RANK, DENSE_RANK, and ROW_NUMBER. When used in an SQL statement, they return the ranking and row number as a scalar value. Rank is the ordinal value of a row in a defined set of rows. You can specify that the result be returned with (RANK) and without (DENSE_RANK) gaps. ROW_NUMBER is a sequential row number assigned a result row.

- Index compression

  One of the more popular solutions to a query performance dilemma is an index. Adding an index can fix many SQL issues. However, adding an index has a cost, and that is the additional disk space consumed. You have to decide between disk space consumption and a poorly running SQL statement. Moreover, DB2 9 now supports powerful index enhancements. So you can find yourself using more disk storage for indexes in DB2 9. DB2 9 does come with a near-perfect solution for this issue: index compression.

Even though query types used in a data warehouse environment can significantly benefit from the addition of an index, it is possible for a data warehouse to reach a point were the indexes have storage requirements that are equal to, if not sometimes greater than, the table data storage.

Index compression in DB2 9 is a possible solution to an index disk storage issue. Early testing and information obtained from first implementers indicates that significant disk savings can be achieved by using index compression. With index compression being measured as high as 75 percent, you can expect to achieve, on average, about a 50 percent index compression rate. Of course, compression also carries cost. In certain test cases, there was a slight decrease in class 1 CPU time when accessing a compressed index. But you can expect your total CPU time, class 1 and class 2 SRB CPU time combined, to increase. However, CPU cost is only realized during an index page I/O. After the index has been decompressed into a buffer pool or compressed during the write to disk, compression adds zero cost.

Unlike data compression, there is no performance advantage directly from the use of compression. Index compression is strictly for reducing your index disk storage. If any performance gain occurs, it appears when the optimizer can take advantage of one of the additional indexes that now exists, an index that might never have been created because of disk space constraints prior to the introduction of index compression. Index compression is available only after you have upgraded your DB2 9 subsystem to NFM.

- Data-partitioned secondary index

  A data warehouse usually means large volumes of data. That translates to partitioned table spaces, and partitioned table spaces often call for Data-Partitioned Secondary Indexes (DPSIs). DB2 9 has made improvements to DPSIs that should make them more usable and popular. For example, a DPSI can now get index-only access and a DPSI index can be defined as unique when columns in the DPSI key are a superset of partitioning columns. A few more subtle DPSI improvements include enhanced page range screening to help avoid hitting every partition, more parallelism, and more index look-aside.

  Remember the lessons learned in DB2 Version 8 if you use DPSIs. Always include a local predicate on the leading columns of the DPSI index to avoid scanning the entire index structure. Also, remember that when using a DPSI, you will see an increase in the number of VSAM data sets that DB2 will need to use. Make sure you adjust your DSMAX value appropriately.

  All these DPSI index improvements are available in conversion mode (CM).

- Prefetch improvements

  In a data warehousing environment, queries use prefetch. There are situations where a stage 2 predicate is your only option, and stage 2 means prefetch. Prefetch has improved in DB2 9. Prefetch quantity has increased. When a pool size becomes larger than 40,000 pages, a larger prefetch quantity is used. If the buffer pool is greater than 40,000 and less than or equal to 80,000 pages, the prefetch quantity is 64 pages for SQL processing and 64 pages for utilities. The utility page quantity jumps to 128 pages when the VPSIZE * VPSEQT is greater than 80,000 pages. Of course, this description of prefetch quantities is based on 4 K page size. These numbers must be adjusted for other page sizes.

  When DB2 uses dynamic prefetch is also a significant change. Dynamic prefetch is used for index scans and table access through table scans. When dynamic prefetch is picked at bind time, it can switch between sequential prefetch and index access based on the data pages accessed using *sequential detection*. In addition, dynamic prefetch does not use a triggering page and can scan forward or backward, something that is handy for a backward index scan. Dynamic prefetch engines can also run in parallel. A few utilities available through APAR PK44026 are recent converts to dynamic prefetch. For example, dynamic prefetch is enabled for the following phases:

  – UNLOAD phase of REORG INDEX
  – The UNLOAD and BUILD phases of REORG TABLESPACE PART
  – The BUILD phase of LOAD REPLACE PART
  – The RUNSTATS phase of RUNSTATS INDEX

  The changes to dynamic prefetch in DB2 9 apply to single table access, multi-table join, outer join, subquery, and union. Sequential prefetch, as of DB2 9, is used only for table space scans.

  Dynamic prefetch could have performance advantages over sequential prefetch because it is not dependent on the optimizer making the correct decision the first time. It allows the access path to be changed dynamically as data patterns change. The prefetch changes are available in DB2 9 CM.

- Reordered Row Format (RRF)

  This is a significant change of which few are aware. This change affects how the variable length columns of a table are ordered when written to disk. A data warehouse, by its nature, can have a number of variable length columns.

  A challenge for DB2 professionals is placement of the fixed length (CHAR) columns versus the placement of variable length (VARCHAR) columns in a newly created table definition. There have been conflicting views on what is most efficient, and the best coding practice. Most have agreed that the VARCHAR columns should go at the end of the row. Or, at least, infrequently updated variable length columns should go last. However, no matter what order you use to define table columns, after DB2 finds a VARCHAR column when retrieving a row, DB2 must calculate the starting position of subsequent columns in that row. Until DB2 reads the length of the VARCHAR column (the two-byte prefix on every VARCAHR column), it cannot determine where to find the beginning of the next column that needs to be retrieved.

DB2 9 provides a potential solution. After you get to DB2 9 for z/OS NFM, all VARCHAR columns on newly created table spaces are placed at the end of the row. DB2 9 can make this type of decision because the row format in DB2 9 has changed. DB2 9 introduces RRF, a straightforward concept in which a DB2 9 NFM row has all the fixed length columns first, followed by pointers to the beginning of each VARCHAR row. The pointers are followed by the actual variable length data. Rather than scan what could be lengthy columns to find the beginning of a column, DB2 only needs to scan the list of pointers to find the location for the beginning of the column you want.

You get RRF for any table space created in DB2 9 NFM. Additionally, any table spaces, or table space partitions, are converted to RRF when a REORG or LOAD REPLACE is run against that table space or table space partition. Be cautious when working with partitioned table spaces. If you only REORG selected partitions of a table space, you end up with partitions, the ones that have been reorganized, in the new RRF. The remaining partitions that have not yet been reorganized stay in basic row format. Basic row format is the phrase used to describe the row format prior to DB2 9 for z/OS NFM. RRF is available only after you have upgraded your DB2 9 subsystem to NFM.

Conclusion

This has been a brief overview of the z/OS, System z, and DB2 for z/OS features and functions from which a data warehouse or data warehouse application could benefit. There are many more. The attempt here is to demonstrate the mainframes strong support of data warehousing and how closely integrated the hardware, operating systems, and database are on the mainframe.

Large businesses choose DB2 for z/OS because they need a robust database server that ensures superior availability and scalability. Superior availability and scalability in a Parallel Sysplex® environment are the key features that distinguish DB2 for z/OS from other database servers. Because of these qualities, DB2 for z/OS is widely deployed in the following industries:

- Major credit card companies
- Banks
- Insurance companies
- Brokerage companies
- Credit information companies

These are the types of companies that process high volumes of transactions that require millions of concurrent updates every day.

Glossary

Access control list (ACL). The list of principals that have explicit permission (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

Aggregate. Pre-calculated and pre-stored summaries, kept in the data warehouse to improve query performance

Aggregation. An attribute level transformation that reduces the level of detail of available data. For example, having a Total Quantity by Category of Items rather than the individual quantity of each item in the category.

Analytic. An application or capability that performs analysis on a set of data.

Application programming interface. An interface provided by a software product that enables programs to request services.

Asynchronous messaging. A method of communication between programs in which a program places a message on a message queue, then proceeds with its own processing without waiting for a reply to its message.

Attribute. A field in a dimension table.

BLOB. Binary Large Object. A block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted.

Commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins.

Compensation. The ability of DB2 to process SQL that is not supported by a data source on the data from that data source.

Composite key. A key in a fact table that is the concatenation of the foreign keys in the dimension tables.

Computer. A device that accepts information (in the form of digitalized data) and manipulates it for result based on a program or sequence of instructions on how the data is to be processed.

Configuration. The collection of brokers, their execution groups, the message flows and sets that are assigned to them, and the topics and associated access control specifications.

Connector. Used to connect to the input and output ports of operators.

DDL. Data Definition Language. ASQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE DATABASE.

DML. Data Manipulation Language. An INSERT, UPDATE, DELETE, or SELECT SQL statement.

Data append. A data loading technique where new data is added to the database leaving the existing data unaltered.

Data cleansing. A process of data manipulation and transformation to eliminate variations and inconsistencies in data content. This is typically to improve the quality, consistency, and usability of the data.

Data federation. The process of enabling data from multiple heterogeneous data sources to appear as if it is contained in a single relational database. Can also be referred to "distributed access".

Data mart. An implementation of a data warehouse, typically with a smaller and more tightly restricted scope - such as for a department, workgroup, or subject area. It could be independent, or derived from another data warehouse environment (dependent).

Data mart - Dependent. A data mart that is consistent with, and extracts its data from, a data warehouse.

Data mart - Independent. A data mart that is standalone, and does not conform with any other data mart or data warehouse.

Data mining. A mode of data analysis that has a focus on the discovery of new information, such as unknown facts, data relationships, or data patterns.

Data martition. A segment of a database that can be accessed and operated on independently even though it is part of a larger data structure.

Data refresh. A data loading technique where all the data in a database is completely replaced with a new set of data.

Data silo. A standalone set of data in a particular department or organization used for analysis, but typically not shared with other departments or organizations in the enterprise.

Data warehouse. A specialized data environment developed, structured, shared, and used specifically for decision support and informational (analytic) applications. It is subject oriented rather than application oriented, and is integrated, non-volatile, and time variant.

Database instance. A specific independent implementation of a DBMS in a specific environment. For example, there might be an independent DB2 DBMS implementation on a Linux server in Boston supporting the eastern offices, and another separate and independent DB2 DBMS on the same Linux server supporting the western offices. They would represent two instances of DB2.

Database partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

DataBlades. These are program modules that provide extended capabilities for Informix databases, and are tightly integrated with the DBMS.

DB connect. Enables connection to several relational database systems and the transfer of data from these database systems into the SAP Business Information Warehouse.

Debugger. A facility on the Message Flows view in the Control Center that enables message flows to be visually debugged.

Deploy. Make operational the configuration and topology of the broker domain.

Dimension. Data that further qualifies and/or describes a measure, such as amounts or durations.

Distributed application In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

Drill-down. Iterative analysis, exploring facts at more detailed levels of the dimension hierarchies.

Dynamic SQL. SQL that is interpreted during execution of the statement.

Engine. A program that performs a core or essential function for other programs. A database engine performs database functions on behalf of the database user programs.

Enrichment. The creation of derived data. An attribute level transformation performed by an algorithm to create one or more new (derived) attributes.

Extenders. These are program modules that provide extended capabilities for DB2, and are tightly integrated with DB2.

FACTS. A collection of measures, and the information to interpret those measures in a given context.

Federated data. A set of physically separate data structures that are logically linked together by mechanism, for analysis, but which remain physically in place.

Federated server. Any DB2 server where the WebSphere Information Integrator is installed.

Federation. Providing a unified interface to diverse data.

Gateway. A means to access a heterogeneous data source. It can use native access or ODBC technology.

Grain. The fundamental lowest level of data represented in a dimensional fact table.

Instance. A particular realization of a computer process. Relative to database, the realization of a complete database environment.

Java Database Connectivity. An application programming interface that has the same characteristics as ODBC but is specifically designed for use by Java database applications.

Java Development Kit. Software package used to write, compile, debug and run Java applets and applications.

Java Message Service. An application programming interface that provides Java language functions for handling messages.

Java Runtime Environment. A subset of the Java Development Kit that allows you to run Java applets and applications.

Materialized query table. A table where the results of a query are stored, for later reuse.

Measure. A data item that measures the performance or behavior of business processes.

Message domain. The value that determines how the message is interpreted (parsed).

Message flow. A directed graph that represents the set of activities performed on a message or event as it passes through a broker. A message flow consists of a set of message processing nodes and message processing connectors.

Message parser. A program that interprets the bit stream of an incoming message and creates an internal representation of the message in a tree structure. A parser is also responsible to generate a bit stream for an outgoing message from the internal representation.

Meta data. Typically called data (or information) about data. It describes or defines data elements.

MOLAP. Multi-dimensional OLAP. Can be called MD-OLAP. It is OLAP that uses a multi-dimensional database as the underlying data structure.

Multi-dimensional analysis. Analysis of data along several dimensions. For example, analyzing revenue by product, store, and date.

Multi-tasking. Operating system capability which allows multiple tasks to run concurrently, taking turns using the resources of the computer.

Multi-threading. Operating system capability that enables multiple concurrent users to use the same program. This saves the overhead of initiating the program multiple times.

Nickname. An identifier that is used to reference the object located at the data source that you want to access.

Node group. Group of one or more database partitions.

Node. A processing step (in, as examples, data, control or message flows.

ODS. (1) Operational data store. A relational table for holding clean data to load into InfoCubes, and can support query activity.
(2) Online Dynamic Server. An older name for IDS.

OLAP. OnLine Analytical Processing. Multi-dimensional data analysis, performed in real-time. Not dependent on underlying data schema.

Open database connectivity. A standard application programming interface for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group.

Optimization. The capability to enable a process to execute and perform in such a way as to maximize performance, minimize resource utilization, and minimize the process execution response time delivered to the end user.

Partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

Pass-through. The act of passing the SQL for an operation directly to the data source without being changed by the federation server.

Pivoting. Analysis operation where user takes a different viewpoint of the results. For example, by changing the way the dimensions are arranged.

Primary key. Field in a table that is uniquely different for each record in the table.

Process. An instance of a program running in a computer.

Program. A specific set of ordered operations for a computer to perform.

Pushdown. The act of optimizing a data operation by pushing the SQL down to the lowest point in the federated architecture where that operation can be executed. More simply, a pushdown operation is one that is executed at a remote server.

ROLAP. Relational OLAP. Multi-dimensional analysis using a multi-dimensional view of relational data. A relational database is used as the underlying data structure.

Roll-up. Iterative analysis, exploring facts at a higher level of summarization.

Server. A computer program that provides services to other computer programs (and their users) in the same or other computers. However, the computer that a server program runs in is also frequently referred to as a server.

Shared nothing. A data management architecture where nothing is shared between processes. Each process has its own processor, memory, and disk space.

Spreadmart. A standalone, non-conforming, non-integrated set of data, such as a spreadsheet, used for analysis by a particular person, department, or organization.

Static SQL. SQL that has been compiled prior to execution. Typically provides best performance.

Static SQL. SQL that has been compiled prior to execution. Typically provides best performance.

Subject Area. A logical grouping of data by categories, such as customers or items.

Synchronous messaging. A method of communication between programs in which a program places a message on a message queue and waits for a reply before resuming its own processing.

Task. The basic unit of programming that an operating system controls. Also see **Multi-Tasking**.

Thread. The placeholder information associated with a single use of a program that can handle multiple concurrent users. Also see Multi-Threading.

Type mapping. The mapping of a specific data source type to a DB2 UDB data type

Unit of work. A recoverable sequence of operations performed by an application between two points of consistency.

User mapping. An association made between the federated server user ID and password and the data source (to be accessed) used ID and password.

Virtual database. A federation of multiple heterogeneous relational databases.

Warehouse catalog. A subsystem that stores and manages all the system metadata.

Wrapper. The means by which a data federation engine interacts with heterogeneous sources of data. Wrappers take the SQL that the federation engine uses and maps it to the API of the data source to be accessed. For example, they take DB2 SQL and transform it to the language understood by the data source to be accessed.

xtree. A query-tree tool that allows you to monitor the query plan execution of individual queries in a graphical environment.

Abbreviations and acronyms

ACS	Access Control System
ADK	Archive Development Kit
AIX	Advanced Interactive eXecutive from IBM
API	Application Programming Interface
AQR	automatic query re-write
AR	access register
ARM	automatic restart manager
ART	access register translation
ASCII	American Standard Code for Information Interchange
AST	Application Summary Table
BLOB	Binary Large OBject
BW	Business Information Warehouse (SAP)
CCMS	Computing Center Management System
CFG	Configuration
CLI	Call Level Interface
CLOB	Character Large OBject
CLP	Command Line Processor
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CS	Cursor Stability
DAS	DB2 Administration Server
DB	Database
DB2	Database 2
DB2 UDB	DB2 Universal DataBase
DBA	Database Administrator
DBM	DataBase Manager
DBMS	DataBase Management System
DCE	Distributed Computing Environment
DCM	Dynamic Coserver Management
DCOM	Distributed Component Object Model
DDL	Data Definition Language. A SQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE.
DES	Data Encryption Standard
DIMID	Dimension Identifier
DLL	Dynamically Linked Library
DML	Data Manipulation Language. An INSERT, UPDATE, DELETE, or SELECT SQL statement.
DMS	Database Managed Space
DPF	Data Partitioning Facility
DRDA®	Distributed Relational Database Architecture™
DSA	Dynamic Scalable Architecture
DSN	Data Source Name
DSS	Decision Support System
EAI	Enterprise Application Integration
EBCDIC	Extended Binary Coded Decimal Interchange Code
EDA	Enterprise Data Architecture
EDU	Engine Dispatchable Unit
EDW	Enterprise Data Warehouse
EGM	Enterprise Gateway Manager
EJB	Enterprise Java Beans

ER	Enterprise Replication
ERP	Enterprise Resource Planning
ESE	Enterprise Server Edition
ETL	Extract, Transform, and Load
ETTL	Extract, Transform/Transport, and Load
FP	Fix Pack
FTP	File Transfer Protocol
Gb	Gigabits
GB	Gigabytes
GUI	Graphical User Interface
HADR	High Availability Disaster Recovery
HDR	High availability Data Replication
HPL	High Performance Loader
I/O	Input/Output
IBM	International Business Machines Corporation
ID	Identifier
IDE	Integrated Development Environment
IDS	Informix Dynamic Server
II	Information Integrator
IMG	Integrated Implementation Guide (for SAP)
IMS™	Information Management System
ISAM	Indexed Sequential Access Method
ISM	Informix Storage Manager
ISV	Independent Software Vendor
IT	Information Technology
ITR	Internal Throughput Rate
ITSO	International Technical Support Organization
IX	Index
J2EE	Java 2 Platform Enterprise Edition
JAR	Java Archive
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JE	Java Edition
JMS	Java Message Service
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KB	Kilobyte (1024 bytes)
LDAP	Lightweight Directory Access Protocol
LPAR	Logical Partition
LV	Logical Volume
Mb	Megabits
MB	Megabytes
MDC	Multidimensional Clustering
MPP	Massively Parallel Processing
MQI	Message Queuing Interface
MQT	Materialized Query Table
MRM	Message Repository Manager
MTK	DB2 Migration ToolKit for Informix
NPI	Non-Partitioning Index
ODBC	Open DataBase Connectivity
ODS	Operational Data Store
OLAP	OnLine Analytical Processing
OLE	Object Linking and Embedding
OLTP	OnLine Transaction Processing
ORDBMS	Object Relational DataBase Management System
OS	Operating System
O/S	Operating System
PDS	Partitioned Data Set
PIB	Parallel Index Build

PSA	Persistent Staging Area
RBA	Relative Byte Address
RBW	Red Brick® Warehouse
RDBMS	Relational DataBase Management System
RID	Record Identifier
RR	Repeatable Read
RS	Read Stability
SCB	Session Control Block
SDK	Software Developers Kit
SID	Surrogage Identifier
SMIT	Systems Management Interface Tool
SMP	Symmetric MultiProcessing
SMS	System Managed Space
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPL	Stored Procedure Language
SQL	Structured Query
TCB	Thread Control Block
TMU	Table Management Utility
TS	Tablespace
UDB	Universal DataBase
UDF	User Defined Function
UDR	User Defined Routine
URL	Uniform Resource Locator
VG	Volume Group (Raid disk terminology).
VLDB	Very Large DataBase
VP	Virtual Processor
VSAM	Virtual Sequential Access Method
VTI	Virtual Table Interface
WSDL	Web Services Definition Language
WWW	World Wide Web
XBSA	X-Open Backup and Restore APIs
XML	eXtensible Markup Language
XPS	Informix eXtended Parallel Server

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get Redbooks" on page 599. Note that of the documents referenced here might be available in softcopy only.

- *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-7274
- *Data Mart Consolidation: Getting Control of Your Enterprise Information*, SG24-6653
- *Dimensional Modeling: In a Business Intelligence Environment*, SG24-7138
- *Up and Running with DB2 UDB ESE: Partitioning for Performance in an e-Business Intelligence World*, SG24-6917
- *InfoSphere Warehouse: Cubing Services and Client Access Interfaces*, SG24-7582
- *Multidimensional Analytics: Delivered with InfoSphere Warehouse Cubing Services*, SG24-7679
- *Dynamic Warehousing: Data Mining Made Easy*, SG24-7418

Online resources

These web sites are also relevant as further information sources:

- Workload Management (WLM) Tutorial

  http://www.ibm.com/developerworks/data/tutorials/dm-0908db2workload/?S_TACT=105AGY82&S_CMP=MAVE
- Best Practices Workload Management

  http://www.ibm.com/developerworks/data/bestpractices/workloadmanagement/

- Introduction to DB2 9.5 workload management

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.wlm.doc/doc/c0052594.html

- Text analysis in InfoSphere Warehouse, Part 1: Architecture overview and example of information extraction with regular expressions

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0906textanalysis/index.html

- Text analysis in InfoSphere Warehouse, Part 2: Dictionary-based information extraction combined with Cognos reporting

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0907textanalysis2/index.html

- Designing and deploying a security model using IBM InfoSphere Warehouse Cubing Services

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0909securityinfospherecubing/

- Using virtual cubes in IBM InfoSphere Warehouse 9.7 to combine business scenarios and to improve performance

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0909infospherevirtualcubes/

- Q-Replication Roadmap

  http://www.ibm.com/developerworks/data/roadmaps/qrepl-roadmap.html/

- Sample jobs for Q replication and event publishing at the link:

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.swg.im.repl.zoscust.doc/topics/iiyrczossampq.html

- WebSphere MQ documentation can be found at the following link:

  http://www-01.ibm.com/software/integration/wmq/library/

- Detailed instructions on how to setup InfoSphere administration console for System z can be found at the following link:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0905systemzwarehouse/index.html

- A good starting point for all replication product documentation is the following link:

  http://www.ibm.com/developerworks/data/roadmaps/qrepl-roadmap.html

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

InfoSphere Warehouse: A Robust Infrastructure For Business Intelligence

Comprehensive platform for architected solutions

Faster time to value for business businesses

Standards based for flexibility and change

In this IBM Redbooks publication we describe and demonstrate Version 9.7 of IBM InfoSphere Warehouse. InfoSphere Warehouse is a comprehensive platform with all the functionality required for developing robust infrastructure for business intelligence solutions. It enables companies to access and analyze operational and historical information, whether structured or unstructured, to gain business insight for improved decision making. InfoSphere Warehouse solutions simplify the processes of developing and maintaining a data warehousing infrastructure and can significantly enhance the time to value for business analytics.

The InfoSphere Warehouse platform provides a fully integrated environment built around IBM DB2 9.7 server technology on Linux, UNIX and Microsoft Windows platforms, as well as System z. Common user interfaces support application development, data modeling and mapping, SQL transformation, online application processing (OLAP) and data mining functionality from virtually all types of information.

Composed of a component-based architecture, it extends the DB2 data warehouse with design-side tooling and runtime infrastructure for OLAP, data mining, inLine analytics and intra-warehouse data movement and transformation, on a common platform.

SG24-7813-00 0738434329