

Solving Operational Business Intelligence with InfoSphere Warehouse Advanced Edition

Comprehensive solution for operational warehouse challenges

Learning-integrated business intelligence framework

Advanced analytics and integrated tool examples



Whei-Jen Chen
Pat Bates
Tim Donovan
Garrett Fitzsimons
Jon Lind
Rogério Silva



International Technical Support Organization

**Solving Operational Business Intelligence with
InfoSphere Warehouse Advanced Edition**

October 2012

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (October 2012)

This edition applies to IBM InfoSphere Warehouse Advanced Edition Version 10.1.

© Copyright International Business Machines Corporation 2012. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Acknowledgements	xiii
Now you can become a published author, too!	xiv
Comments welcome	xv
Stay connected to IBM Redbooks	xv
Chapter 1. Solving complex operational warehouse challenges	1
1.1 Current business challenges	2
1.1.1 Understand data and the complex warehouse environment	2
1.1.2 Deliver actionable business insights - when and where needed	3
1.1.3 Challenges for complex warehouse environments	4
1.1.4 Gain insight without boundaries with IBM InfoSphere Warehouse	5
1.2 Comprehensive data warehouse solution	5
1.3 InfoSphere Warehouse advanced tooling	10
Chapter 2. Overview of InfoSphere Warehouse Advanced Edition	15
2.1 InfoSphere Advanced Enterprise Edition components	16
2.1.1 DB2 10 Enterprise Server Edition	16
2.1.2 WebSphere Application Server	17
2.1.3 InfoSphere Warehouse Server	17
2.1.4 InfoSphere Federation Server	17
2.1.5 InfoSphere Replication Server	18
2.1.6 Advanced access control	18
2.1.7 IBM Cognos Business Intelligence for Reporting	19
2.1.8 IBM Data Studio	19
2.1.9 Optim Database Administrator	20
2.1.10 Optim Development Studio	20
2.1.11 InfoSphere Warehouse Client - IBM Design Studio and the wh command	20
2.1.12 InfoSphere Data Architect	21
2.1.13 InfoSphere Optim Query Workload Tuner	22
2.1.14 InfoSphere Optim Performance Manager Extended Edition	22
2.1.15 InfoSphere Optim High Performance Unload	23
2.1.16 InfoSphere Optim Configuration Manager	23
2.1.17 DB2 Recovery Expert	24

2.1.18	DB2 Merge Backup	24
2.1.19	InfoSphere Warehouse Packs	24
2.2	InfoSphere Warehouse technical overview	25
2.2.1	InfoSphere Warehouse core architecture	26
2.2.2	Overview of IBM Cognos Business Intelligence	42
2.2.3	Architectural overview of the Optim family of tools	46
2.2.4	InfoSphere Replication Server simplified overview	50
2.2.5	InfoSphere Warehouse Advanced Edition client applications	53
2.2.6	InfoSphere Warehouse Advanced Edition command-line tools and utilities	58
2.3	InfoSphere Warehouse implementation scenarios	59
2.3.1	Scenarios for implementing InfoSphere Warehouse core products	60
2.3.2	Scenarios for implementing the Optim family of tools	64
2.3.3	Alternative IBM solutions for data warehousing	65
Chapter 3. Scenario: Warehouse Live Demo Set		69
3.1	The problem: warehouse growth for “Mom and Pop” retail	70
3.2	Warehouse Live! Demo	71
3.3	Example architecture used in this book	77
Chapter 4. Data modeling: End to end		81
4.1	Start with the business problem	82
4.1.1	Online analytical processing	82
4.2	The logical modeling process	86
4.2.1	Identify the business process	86
4.2.2	Define the granularity	87
4.2.3	Identify the dimension tables	87
4.2.4	Identify the fact tables	92
4.3	InfoSphere Data Architect and logical dimensional modeling	95
4.3.1	A starting point: The physical data model	97
4.3.2	Create a logical data model	102
4.3.3	Add dimensional notation to the logical data model	104
4.3.4	Create a dimensional physical data model to implement OLAP Data Warehouse	106
4.4	OLAP modeling and cubes	107
4.4.1	Cube model dimensional concepts	112
4.4.2	Define a cube model within Design Studio	119
4.4.3	Deploy a cube model to InfoSphere Warehouse as a cube service	134
4.5	Modeling and IBM Cognos BI	138
4.5.1	Cognos Framework Manager	139
4.5.2	Create a Cognos model from cubing services	141
4.5.3	Create a Cognos model from a logical dimensional model within Data Architect	151

4.6 Introduction to InfoSphere Warehouse Packs	156
4.6.1 Installation example of InfoSphere Warehouse Insight Pack	166
Chapter 5. Temporal data management and analytics in an operational warehouse	173
5.1 Temporal data management concepts and use cases	174
5.1.1 Temporal use cases for the operational data warehouse	175
5.1.2 Temporal concepts	179
5.2 Temporal data management in InfoSphere Warehouse Advanced Enterprise Edition 10	181
5.2.1 Temporal data management with system time	182
5.2.2 System-period temporal tables	192
5.2.3 Temporal data management with business time.	202
5.2.4 Application-period temporal tables	210
5.2.5 Bitemporal tables	220
5.2.6 Views and temporal tables	225
5.2.7 Register settings for temporal tables.	226
5.3 Implications of temporal tables on operational warehousing.	228
5.3.1 Temporal tables, complex workloads and concurrency	228
5.3.2 System-period temporal tables and history table storage	231
5.3.3 History table implications for the warehouse archive strategy	233
5.3.4 Backup, recovery, and pruning history with range partitioning	235
Chapter 6. Managing complex query workloads in an operational warehouse environment.	237
6.1 Get started with Optim Performance Manager in your environment	239
6.1.1 Creating a database connection	240
6.1.2 Configuring database monitoring settings.	240
6.2 Optim Performance Manager in a partitioned database environment.	245
6.2.1 Collecting large volumes of performance metrics	246
6.3 Using Optim Performance Manager to implement a Stage1 configuration of DB2 WLM	247
6.3.1 Sample scenario using DB2 WLM and Optim Performance Manager to manage performance	248
6.4 Understanding query workloads in your environment	253
6.4.1 Identify/define query workloads with Optim Performance Manager	254
6.4.2 Create/monitor new workloads with Optim Performance Manager	256
6.4.3 Define service level objectives for all query workloads.	258
6.5 Improving the overall performance of an activity.	259
6.6 Approaches to workload tuning	260
6.6.1 Tuning a workload for optimal performance	260
6.7 Monitoring query workloads: Real-time operational to show the state and health of the system	261

6.7.1 Health monitoring	262
6.7.2 Performance baselines	264
Chapter 7. Understand and address data latency requirements	267
7.1 Understand your data latency requirements	268
7.1.1 Quantify your service level objectives	268
7.1.2 Calculate the data ingest rate	269
7.1.3 Analyze your ETL scenarios	269
7.2 Design and develop your ETL application.	274
7.3 Use SQW and Bulk Load to get batch data into staging tables.	275
7.3.1 SQW data flow	276
7.3.2 SQW control flow	277
7.3.3 SQW and temporal tables	280
7.4 Create and deploy your SQW application	281
7.5 Introduce parallelism to increase ingest volumes	285
7.5.1 How Ingest and MQT tables correlate with data availability and data latency	286
Chapter 8. Building a corporate backup and recovery strategy	289
8.1 Advanced features for backup and recovery.	291
8.1.1 Advanced recovery solutions explained	291
8.2 Plan a backup and recovery strategy	292
8.2.1 Recovery scenarios and objectives.	293
8.2.2 ETL application architecture and process schedule	294
8.2.3 Backup infrastructure	295
8.2.4 Data needs of downstream systems.	296
8.3 Implement a backup and recovery strategy	296
8.4 Implementing Optim High Performance Unload as part of your recovery strategy.	298
8.4.1 Optim High Performance Unload with backup images	298
8.4.2 Optim High Performance Unload with backup images usage	299
8.4.3 Optim High Performance Unload in a production environment.	300
8.4.4 Optim High Performance Unload with named pipes or output files.	301
8.4.5 Optim High Performance Unload control files	302
8.4.6 Create the control file and issue a db2hpu command.	304
8.4.7 Install and configure Optim High Performance Unload.	305
8.5 Example recovery scenarios using Optim High Performance Unload.	308
8.5.1 Recover existing partitioned tables using backup images on disk	308
8.5.2 Recover dropped partitioned table using backup images on TSM and named pipes	309
8.5.3 Recover data using non-production database and backup images on TSM	311
8.6 DB2 Recovery Expert as part of your recovery strategy	313

8.6.1 Schema level repository	314
8.6.2 Log analysis	314
8.6.3 Recovery	318
8.7 DB2 Merge Backup as part of your backup strategy	321
8.7.1 DB2 Merge Backup as part of your infrastructure	323
8.7.2 Install and configure db2mk in partitioned database environment	324
8.7.3 DB2 Merge Backup command line and control files	325
8.7.4 DB2 Merge Backup with local disk backups	327
8.7.5 DB2 Merge Backup with Tivoli Storage Manager	330
Chapter 9. Managing data lifecycle with InfoSphere Warehouse	335
9.1 The value of data and its age	336
9.2 Manage the cost of storage with multi-temperature management	338
9.2.1 Using multi-temperatures features in a sample scenario	339
9.3 Aging data	345
9.3.1 How active is the data	346
9.4 Optim High Performance Unload to archive or migrate cold data	347
9.4.1 Archive cold data	348
9.4.2 Migrate data from production to an alternate database	350
Chapter 10. Techniques for data mining in an operational warehouse	355
10.1 Data mining in an operational warehouse environment	356
10.1.1 Data mining overview	356
10.1.2 Data mining scenarios in an operational data warehousing environment	364
10.2 InfoSphere Warehouse Advanced Enterprise Edition 10.1 tools and features for data mining	368
10.2.1 Source data exploration in InfoSphere Warehouse 10.1	368
10.2.2 Data preparation	376
10.2.3 Data mining modeling in InfoSphere Warehouse Design Studio	380
10.2.4 Performing data mining scoring with other data mining models through PMML	383
10.3 Extended data mining techniques using SAS Enterprise Miner	383
10.3.1 About SAS Enterprise Miner	384
10.3.2 In-database scoring with SAS and InfoSphere Warehouse	386
10.3.3 Access the scoring model	387
10.3.4 Additional scoring examples	388
10.4 Deploy and visualize mining results with IBM Cognos 10 Business Intelligence	389
10.4.1 List of clusters report	392
10.4.2 Breakdown by cluster report	393
Chapter 11. Cognos Business Intelligence for InfoSphere Warehouse	395
11.1 Cognos Business Intelligence 10 and InfoSphere Warehouse	396

11.1.1 IBM Cognos 10 Business Intelligence features	396
11.1.2 IBM Cognos 10 Business Intelligence architecture	399
11.2 Business modeling with IBM Cognos Framework Manager	401
11.3 IBM Cognos Business Insight	402
11.3.1 Interact with IBM Cognos Business Insight Editor	404
11.3.2 Run the IBM Cognos Business Insight dashboard	405
11.4 IBM Cognos disconnected report interaction	406
11.4.1 IBM Cognos Active Report overview	406
11.4.2 IBM Cognos Active Report features	407
11.4.3 Enable an existing report for active reporting	407
11.5 IBM Cognos access from mobile devices	411
11.6 Further information	412
 Chapter 12. InfoSphere Warehouse resilience with Optim Configuration Manager	 413
12.1 Challenges to maintaining an operational warehouse	414
12.2 InfoSphere Optim Configuration Manager	414
12.3 Integration of the Optim family of tools	418
12.4 Monitor a DB2 database configuration	420
12.5 Optim tools runtime client	427
12.6 Optimize database storage with Optim Configuration Manager	435
12.6.1 Example Optim Configuration Manager storage-saving job	436
12.6.2 Automate multi-temperature storage migration with Optim Configuration Manager	440
12.7 InfoSphere Warehouse high availability with Q replication and Optim Configuration Manager	444
 Appendix A. InfoSphere Warehouse Editions comparison	 477
 Related publications	 481
IBM Redbooks	481
Other publications	481
Online resources	482
Help from IBM	483

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

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®	InfoSphere®	Redbooks (logo)  ®
Cognos®	Intelligent Miner®	SPSS®
DataStage®	Lotus®	Tivoli®
DB2®	Optim™	TM1®
developerWorks®	pureQuery™	WebSphere®
IBM®	pureScale®	z/OS®
Information on Demand®	Rational®	
Informix®	Redbooks®	

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Linux is a trademark of Linus Torvalds 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.

NOW, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

IBM® InfoSphere® Warehouse is the IBM flagship data warehouse platform for departmental data marts and enterprise data warehouses. It offers leading architecture, performance, backup, and recovery tools that help improve efficiency and reduce time to market through increased understanding of current data assets, while simplifying the daily operations of managing complex warehouse deployments.

InfoSphere Warehouse Advanced Enterprise Edition delivers an enhanced set of database performance, management, and design tools. These tools assist companies in maintaining and increasing value from their warehouses, while helping to reduce the total cost of maintaining these complex environments.

In this IBM Redbooks® publication we explain how you can build a business intelligence system with InfoSphere Warehouse Advanced Enterprise to manage and support daily business operations for an enterprise, to generate more income with lower cost. We describe the foundation of the business analytics, the Data Warehouse features and functions, and the solutions that can deliver immediate analytics solutions and help you drive better business outcomes.

We show you how to use the advanced analytics of InfoSphere Warehouse Advanced Enterprise Edition and integrated tools for data modeling, mining, text analytics, and identifying and meeting the data latency requirements. We describe how the performance and storage optimization features can make building and managing a large data warehouse more affordable, and how they can help significantly reduce the cost of ownership. We also cover data lifecycle management and the key features of IBM Cognos® Business Intelligence.

This book is intended for data warehouse professionals who are interested in gaining in-depth knowledge about the operational business intelligence solution for a data warehouse that the IBM InfoSphere Warehouse Advanced Enterprise Edition offers.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application

development, database design and modeling, and IBM DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development, and an IBM Certified IT Specialist.



Pat Bates is a data warehouse technical specialist and a member of the Worldwide Information Management Technical Sales Team responsible for InfoSphere Warehouse, IBM Smart Analytics System and Netezza®. He lives and works in southern New Jersey and has been involved with IBM data warehousing and in-database analytics since 2001. With 22 years in the industry, Pat has served IBM Information Management as a product development manager, product manager, and technical sales specialist, and he is a frequent speaker at technical workshops worldwide.



Timothy Donovan is a Senior IT Specialist with the Emerging Technology Services (ETS) organization, part of IBM Software Business. He has recently been working with the IBM Information Management Software Service team, assisting with client data warehouse migrations and new InfoSphere Warehouse solutions. Aligned with Tim's past skill set of high performance computing, Grid, IBM AIX® and Linux, he is currently working to support client projects in new technology areas, with Hadoop and IBM Big Data solutions related to data warehouse environments. Prior to joining ETS, Tim worked for 10 years in the Developer Relations Division, part of the IBM Business Partner program, supporting Independent Solution Vendors (ISVs) with enabling their applications to use a range of IBM technologies.



Garrett Fitzsimons is a data warehousing specialist based in the IBM Dublin lab. He has worked with traditional database and data warehouse applications since 1990 on a variety of platforms in several countries. Garrett has worked on a number of large scale implementations of ERP and data warehousing systems using IBM DB2, Oracle, and Microsoft technologies. He is currently working on providing best practice guidelines for the IBM Smart Analytics System.



Jon Lind serves as a Product Manager for Information Management and the InfoSphere Warehouse product within the IBM Software Group. He works with the DB2 distributed data management team focused on warehouse technologies, and helps clients from around the world adopt IBM innovative warehouse offerings to deliver solutions for their business challenges. Currently, he is responsible for setting product and technology direction for the InfoSphere Warehouse product line. He also works closely with the System Technology group and Business Partners to leverage the hardware technology of IBM and its partners to deploy big data solutions using IBM warehouse technology. Jon joined IBM and the DB2 Technology Development team in Toronto, Canada in 1999, where he was responsible for significant enhancements to the DB2 distributed platform including the re-architecture of the engine for version 8. In 2003, he transferred to Beaverton, Oregon where he was responsible for delivering the High Availability and Disaster Recovery technology. Since then he has been involved in delivery technology innovations for both IBM pureScale®, Oracle compatibility features, and various warehouse technology developments.



Rogério Silva is a data warehousing Technical Enablement Specialist at the Software Group Information Management Technology Ecosystem (IMTE) in São Paulo, Brazil. He has more than 12 years of experience, holding positions in development and consulting. His areas of expertise include database, data warehousing, and business intelligence. He is an instructor of InfoSphere Warehouse and Netezza technical training.

Acknowledgements

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

Anson Kokkat
Paul McInerney
Richard Swagerman
KitMan Cheung
Irina Delidjakova

Oleg Sotnikov
Malcolm Singh
IBM Software Group, Canada

Somil Kulkarni
Matthias Nicola
David Zimmermann
John Rollins
Deirdre Clyne
IBM Software Group, USA

Thomas Schwarz
IBM Software Group, Germany

Michael Wurst
Rich Lubell
Konrad Emanowicz
Gary Murtagh
Pat G O'Sullivan
IBM Research, Ireland

Fernando Henrique Telles de Souza
Eluizio Henrique Saraiva Barretto
IBM Software Group, Brazil

Linda Robinson
International Technical Support Organization, San Jose Center

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>



Solving complex operational warehouse challenges

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. These improvements 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. We can call that “being successful.”

In this chapter we describe the capabilities required for building an operational 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 for solving complex operational warehouse challenges is provided by the IBM InfoSphere Warehouse offering and the Advanced Edition. We describe this offering in detail in the remaining chapters of this book.

1.1 Current business challenges

Access to timely, accurate information is critical for enterprises that are striving to better serve their clients, compete successfully, and foster innovation. Limited by inflexible traditional data warehouse and incomplete business intelligence (BI) solutions, IT teams are often challenged to meet the information needs of business users. The result is a proliferation of isolated data marts and data warehouses that often contain duplicate, incomplete, or conflicting data.

Worse yet, most BI solutions built on traditional data warehouses are a patchwork of components from multiple vendors, creating integration and enterprise standardization hurdles. The downsides to this type of BI environment can include frustrated users, significant delays in getting new reports, high development and maintenance costs, and poorly performing solutions that do not scale. Information technology teams must be able to support business requirements for various users while containing costs, accommodating ever-increasing data volumes, and dealing with the constantly changing needs of different business constituents.

InfoSphere Warehouse Advanced Edition provides a powerful range of capabilities that go beyond those of traditional warehouses. This comprehensive platform integrates the strength of the IBM DB2 database with a dynamic data warehousing infrastructure that can handle the traditional BI workloads and more operational business requirements. In addition, the Advanced Editions provide tools to simplify the administration and maintenance of a complex warehouse environment.

1.1.1 Understand data and the complex warehouse environment

Organizations today need new ways to differentiate their products and services from those of their competitors. As globalization gains momentum, they can no longer rely on geographic or regulatory advantages to give them a competitive edge. At the same time, product differentiation is increasingly difficult, because technology today enables new ideas to be copied quickly.

While organizations search for new ways to stand out from the competition, they must also enhance operational efficiencies. Along with spiraling infrastructure and personnel costs, many face rising expenses associated with ensuring compliance with strict regulations.

To achieve competitive differentiation and enhance operational efficiencies, organizations must use the information collected within and beyond the enterprise. This information holds the key to improving customer service and

generating more highly targeted product offerings, while also improving fraud detection and increasing the accuracy of financial risk analysis.

Organizations must collect information in various forms and from various sources. Information that is structured and unstructured, operational and transactional, as well as real time and historical might be scattered throughout the enterprise and delivered by various outside sources that are beyond the direct influence of the enterprise. Information can be in databases, data marts, and data warehouses; emails and transaction logs; customer call logs; shopper behavior data; and repair orders. It can even live in XML data locked inside transactional systems that cannot be used or analyzed in the enterprise database.

After information is discovered, collected, and verified for quality, it must be analyzed. Organizations must find the hidden relationships, patterns, and trends that will help them generate insights for improving efficiencies and achieving a competitive benefit.

Businesses are also looking at integrating data warehousing into their business processes to optimize revenue potential and guard against losses such as using fraud or client churn. Data warehousing clients are looking for more cost-effective ways of answering the problems of availability, security, and performance.

1.1.2 Deliver actionable business insights - when and where needed

Companies are drowning in data that is captured in various kinds of data repositories. The goal was to capture data in a safe, reliable, and secure fashion. However, the struggle is always to find a simple but powerful way to ingest the data into an environment and then 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 sheer 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. However, there are solutions, and successful companies must employ them.

InfoSphere Warehouse provides a complete, comprehensive, and multipurpose environment that allows organizations to access, analyze, and act on operational and historical information, whether structured or unstructured. With InfoSphere Warehouse, organizations can gain the insight and agility they need to generate new opportunities, contain costs, and satisfy client needs in their increasingly dynamic business environments.

1.1.3 Challenges for complex warehouse environments

Many existing data warehousing solutions have limitations that make it difficult for organizations to fully use the power of information. Technological barriers often restrict the types of information that businesses can use, the people who can access the information, and the speed at which it can be accessed.

Limited reach

Business users must access and analyze a broad array of information, from the unstructured information in call center notes, emails, and blogs to the structured information in databases, spreadsheets, and other data sources. Yet, current infrastructures often have limited reach. The proliferation of transaction systems and the emergence of new data sources available outside the business force users to cast a wide net to gather the information they need to support better decision making.

Limited access

Business intelligence and analytics tools can help, but only if they are widely available rather than limited to high-level decision makers and specialized analysts. Enterprises must deliver useful and timely information to more people as part of everyday business processes to drive innovation and reduce costs.

Limited depth

Many existing information infrastructures also have limited depth. Often business users are unable to get answers to complex ad hoc questions. IT teams need flexible tools that can capture and deliver more types of information in the way that users need it, when they need it, where they need it and how they need it.

Poor flexibility and responsiveness

Older BI systems often restrict the type and quantity of data that certain users can access. Organizations need ways to support a large numbers of diverse users and enable those users to quickly and easily customize how they receive information based on their specific needs.

In many cases, existing information systems lack the responsiveness required by today's business landscape. Organizations can gain a significant advantage over competitors if they can quickly analyze a wide range of information and deliver actionable insights to executives and frontline decision makers. They need solutions that can be implemented swiftly and then deliver rapid results.

Costly and complicated

Some BI solutions can be costly to acquire, difficult to deploy, and complex to manage. Organizations need solutions that can be implemented quickly and

include capabilities that streamline administration and lower the overall cost of ownership.

Lack of real-time operational data

Business decision making often gets stalled or delayed due to non-availability of accurate, real-time operational data, which results in missing out on identifying opportunities and important insights. This happens because of many reasons. Delays in capturing operational data into the data warehouse due to archaic ETL processes are where the problem begins. Performance overheads in handling heavy workloads and complex queries from the data warehouse for analytics processing is another major issue. Non-responsive resource allocation within the data warehouse as per priority and criticality of data, ultimately results in hindering faster analytic results.

1.1.4 Gain insight without boundaries with IBM InfoSphere Warehouse

Businesses must address these challenges and work to achieve on-demand access to insight. If they succeed, they can target smaller customer segments and communicate with them about their individual needs and wants. They can identify and capitalize on even the smallest trends, attaining competitive advantages normally only realized by more flexible and dynamic businesses. They can detect small behavior patterns that can have a significant influence and impact on business in terms of revenue, expenses, and growth. Most important, they can build competitive strategies around data-driven insights and generate impressive business results.

1.2 Comprehensive data warehouse solution

Designed to be a comprehensive data warehouse solution, InfoSphere Warehouse enables organizations to centrally, accurately, and securely analyze and deliver information as part of their operational and strategic business applications. Different from the traditional data warehouses and BI solutions, which can be complex, rigid, and inflexible, InfoSphere Warehouse simplifies the processes of selecting, deploying, and maintaining an information management infrastructure, while offering the flexibility for dynamically integrating and transforming data into actionable business insights and empowering organizations to make timely, informed decisions in real time as business events occur.

In addition, InfoSphere Warehouse Advanced Editions deliver advanced design, administration, and tooling capabilities that facilitate simplified management and deployment of warehouses across a dynamic business and IT landscape.

InfoSphere Warehouse Advanced Editions, which are available in Enterprise and Departmental Editions, deliver a comprehensive set of InfoSphere Warehouse Packs for simple deployment of leading prebuilt industry models and IBM Cognos Business Intelligence reports. This functionality enables organizations to easily understand and gain insight into their markets, campaigns, clients, and supply chains. Additional management and architecture tools included in the Advanced Enterprise Edition can help organizations deploy, manage, grow, and architect increasingly business-critical and complex warehouse environments.

To speed and simplify deployment, InfoSphere Warehouse can be deployed on any VMware server platform. IT teams can bypass installation and setup activities, reduce time-to-value, and free database administrators (DBAs) to focus on higher-value projects. The virtualized InfoSphere Warehouse provides the flexibility to manage fluctuation in database activity. IT teams have the freedom to adjust processors, data storage capacity, and hardware quickly and easily to adapt to changing business needs.

InfoSphere Warehouse is powered by the IBM DB2 for Linux, UNIX and Windows data server. With its massively scalable, shared-nothing architecture, IBM DB2 provides high performance for mixed-workload query processing against both relational and native XML data. Advanced features such as data partitioning, compression, multidimensional clustering (MDC), materialized query tables (MQT), and OLAP capabilities make IBM DB2 a powerful engine for operational warehousing.

InfoSphere Warehouse provides advanced capabilities for data partitioning, giving IT users multiple ways to distribute data across servers for large-scale parallelism and linear scalability. The shared-nothing architecture of IBM DB2 helps ensure that performance will not degrade as the warehouse grows. And because InfoSphere Warehouse can physically cluster data on multiple dimensions, order data by value range, and limit I/O to relevant data partitions, it helps reduce the work needed to resolve many queries.

InfoSphere Warehouse transparently splits the database across multiple partitions and uses the horsepower of multiple servers to satisfy requests for large amounts of information. SQL statements are automatically decomposed into subrequests that are run in parallel across the partitions. Results of the subrequests are joined to provide final results.

Table partitioning offers easy roll-in and roll-out of table data, flexible index placement, and efficient query processing. Rolling in partitioned table data allows a new range to be easily incorporated into a partitioned table as an additional

data partition. Rolling out partitioned table data enables data ranges to be efficiently separated from a partitioned table for subsequent purging or archiving.

Table partitioning enhances the flexibility of table-level administration by allowing administrative tasks to be performed on individual data partitions. These tasks include detaching and reattaching a data partition, backing up and restoring individual data partitions, and reorganizing individual indexes. Time-consuming maintenance operations can be streamlined by breaking them down into a series of smaller operations. For example, backup operations can work data partition by data partition when the data partitions are placed in separate table spaces.

Multidimensional clustering provides a flexible method to continuously and automatically cluster table data in multiple dimensions. This clustering reduces the amount of I/O required. In addition, it helps reduce the need for database maintenance activities such as reorganization.

Using InfoSphere Warehouse, organizations can use storage optimization technology in DB2 that is designed to significantly reduce disk space requirements and improve query performance. The features in InfoSphere Warehouse allow organizations to automatically compress additional types of data, further reduce storage requirements, and provide quick access to data from disk. With DB2, indexes and temporary tables can be automatically compressed to reduce storage costs, which is an especially useful feature in large data warehouse environments.

Adaptive compression can also help reduce storage costs and improve performance, especially for large I/O-bound warehouse applications and query workloads. Data row compression contributes to storage space savings and helps reduce I/O overhead. At the same time, the stored pages are compressed, which further enhances the compression on disk. Also, because data is compressed, more rows can be cached in the buffer pool of the database to improve query response time and DBAs no longer need to perform REORG operations as frequently.

InfoSphere Warehouse workload management capabilities are designed to enable real-time delivery of business insights without compromising performance. With traditional servers, the strain of mixed workloads can inhibit the delivery of information to a broad set of users and applications. With the advanced workload management provided by InfoSphere Warehouse, DBAs can establish and enforce service levels for users by prioritizing queries from different users and applications and then controlling the number of underlying resources dedicated to those processes.

DB2 10.1 enables organizations to transparently load data from external sources into InfoSphere Warehouse databases without downtime, even allowing real-time business analysis and decision making during the loading process. The

Continuous Data Ingest feature allows IT departments to load data across multiple agents at the same time, while also dynamically switching between the various external load sources to help maximize resource utilization. Continuous data ingest eliminates the data ingest latency provided by batch loading data on infrequent schedules, and is significant for business users that need current operational data in the warehouse.

The *Time Travel Query feature* in DB2 10.1 supports fast and easy time-based trend analysis and applications. Through the standard query language, users can query temporal tables to determine the changes that occurred over a range of time. As the data ages and becomes less relevant, it can be removed from the database.

The *InfoSphere Warehouse Advanced Security feature* is designed to help critical systems maintain security for their key data. DBAs can use this feature to decide who has write access and who has read access to individual rows and columns in any table based on either users or groups. After the LBAC rules are defined, data access control is managed by the DB2 data server and is not apparent to users. In addition to access controls, the Advanced Security feature enables DBAs to determine how that data will be presented to an application or user. This allows for different views of the same data, but with various output masks to secure sensitive data.

InfoSphere Replication Server technology is included in all editions of InfoSphere Warehouse. Organizations looking to provide active/active availability can use bidirectional Q-replication between a pair of source and target DB2 for Linux, UNIX, and Windows data servers.

Embedded analytics capabilities deliver a set of sophisticated yet easy-to-use tools within the data warehouse. These tools provide valuable business intelligence to a wide pool of users. The Cubing Services for OLAP feature enables multidimensional data analysis without extracting data from the warehouse. InfoSphere Warehouse includes native support for the Microsoft PivotTable Service, enabling ad hoc analyses or delivery of standard spreadsheet reporting, all while working within the Microsoft Excel application.

In addition, Cubing Services cubes are first-class data providers to the IBM Cognos platform. The entire suite of Cognos clients and applications can use these powerful warehouse-based data cubes. Unlike solutions that require users to extract data from the warehouse, independently analyze it, and then send the results back to the warehouse, InfoSphere Warehouse provides embedded data mining, modeling, and scoring capabilities. These capabilities enable business users to work with current data and deliver analytics in real time, helping them quickly discover revenue opportunities. InfoSphere Warehouse supports standard data-mining model algorithms such as clustering, associations,

classification, and prediction; additional algorithms might be imported in industry-standard Predictive Model Markup Language (PMML) format.

Most BI solutions cannot access most information captured across the organization, such as call center notes, client feedback, and free-form text fields, along with documents and web pages. InfoSphere Warehouse supports the analysis of previously untapped unstructured data, helping to provide additional insights into client and product issues.

IBM Cognos Business Intelligence allows business users to evaluate a rich set of BI capabilities without incurring upfront costs. Business users can easily access data from their data warehouse; reporting and analysis features allow them to deliver relevant information how, when and where it is needed. The web-based user interface, enterprise-class service-oriented architecture (SOA) foundation, and the ability to access any data sources enable business users to easily develop and deploy reports on the data assets within the warehouse. Combined with the Warehouse Packs (available in the Advanced Editions), Cognos Business Intelligence provides a quick way to deploy warehouse reporting and gain rapid value and insights from data.

InfoSphere Warehouse provides a set of tools that help simplify data warehouse and analytics development and deployment. These interfaces enable users to design the warehouse and populate data structures, as well as perform analytics and manage data mining and multidimensional cubing through common interfaces. Design Studio provides a graphical user interface that enables architects to design, model, reverse engineer and validate physical database schemas. Design Studio is based on IBM InfoSphere Data Architect software and can import and export models from various sources, including CA ERwin.

The SQL Warehousing tool enables DBAs to prepare and populate the data warehouse structures required for data mining, multidimensional analytics, and embedded analytics. Data flows, control flows, and transformations can be built by using Design Studio and deployed within the warehouse. InfoSphere Optim™ Development Studio software helps increase development efficiency for Java data access and facilitates cross-system development and migration. It supports development for DB2, Oracle database, and IBM Informix® software. Its SQL outline feature facilitates developer and DBA collaboration by quickly isolating all the SQL for review and enables impact analysis by correlating SQL with source code, database objects, and ALTER requests.

IBM InfoSphere Optim Database Administrator helps organizations manage databases and database changes without disruption, streamlining change-in-place and database migration scenarios. Built-in analysis and migration features help prevent application outages by ensuring that all related objects are migrated. They support outstanding performance by ensuring that indexes are updated, and they facilitate availability by ensuring that privileges

are migrated. InfoSphere Warehouse also includes InfoSphere Optim Performance Manager, which provides out-of-the-box performance monitoring and management to help improve quality of service and prevent impacts to business operations. Its intuitive, web-based user interface provides use-anywhere monitoring, alerting, and diagnosis of potential performance bottlenecks.

1.3 InfoSphere Warehouse advanced tooling

InfoSphere Warehouse Packs are recent additions to the IBM Industry Models portfolio, which provides structured and deployable business content for a growing number of industry initiatives. These optional offerings contain physical data models, data mining examples, and sample reports based on industry-specific business issues that help organizations compress project timelines and reduce cost and risk compared with custom, in-house data warehouse projects.

Each Warehouse Pack contains the physical data models, sample reports, and cubing models necessary for business to deploy a warehouse quickly and efficiently to start loading data and delivering real business value as quickly as possible, while allowing the warehouse to expand and grow and evolve to the specifics of the business.

InfoSphere Warehouse Advanced Editions include warehouse packs for client insight, market and campaign insight, and supply chain insight. With all three InfoSphere Warehouse Packs, organizations can use as many packs as they want to help accelerate the deployment of their data warehouses.

InfoSphere Data Architect is a collaborative data design solution that can be used to discover, model, visualize, relate, and standardize diverse and distributed data assets. When used with the InfoSphere Warehouse Packs, it allows designers to enhance those models and develop and deploy custom models tailored to their specific industry or business environment.

InfoSphere Optim Query Workload Tuner helps cut development costs and improve performance by providing expert advice on writing high-quality queries and improving database design. Its easy-to-use advisors help developers write more efficient SQL queries, identify query candidates, and facilitate analysis of those queries. In addition, Query Workload Tuner allows for the analysis of not only a single query, but also can analyze a workload of queries to suggest changes that would help improve performance.

InfoSphere Optim High Performance Unload enables DBAs to work with large quantities of data with minimal effort, and helps them produce fast results. It can

run concurrently with production applications and helps reduce the risk of online production delays by fitting work into tight or shrinking batch windows. InfoSphere Optim High Performance Unload can recover data from single table failures or accidental table drops by extracting tables from a backup image. It also helps speed database migrations, enabling organizations to more quickly realize the value of new hardware and software investments.

DB2 Merge Backup is designed to minimize the impact of backups on production systems and shorten recovery times on production servers. It helps ensure a complete, accurate DB2 backup as data volumes increase, and can merge several incremental backups with the latest full backup to create a new, more current full backup.

DB2 Recovery Expert helps reduce the amount of time it takes to recover production warehouses. It can slim log transactions by creating subsets by table, and assist DBAs and IT support teams in determining the best path of recovery and the necessary resources. The log analysis feature enables DBAs to monitor activity to sensitive DB2 tables. DBAs can view data changes by dates, users, tables, and other criteria. They can also isolate accidental or unwanted changes made to the database tables and quickly undo those changes with minimal disruption to the business.

InfoSphere Optim Configuration Manager employs a central repository designed to help IT departments manage client configurations, track configuration changes, and resolve problems by working with current and historical configuration data. DBAs can modify properties of deployed database clients to improve database transaction performance, enforce security and privacy controls, and allocate data clients among data servers to balance workloads.

Managing the configuration of database clients also helps DBAs track unwarranted changes, find root causes, and ensure that clients are set up to run database transactions correctly. InfoSphere Optim Configuration Manager facilitates discovery of client information for databases such as host name, IP address, user ID, client version information and more to help organizations manage data more effectively and efficiently.

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 tools 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 time

consuming, not only 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 by using the SQL Warehousing tool (SQW).

The SQW tool is part of the Eclipse-based Design Studio. SQW enables DBAs to define data transformation and data movement tasks by 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).

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 IBM 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 design feature decreases the learning curve as users move from tool to tool. The InfoSphere Warehouse Design Studio is based on the 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.

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 that represents the physical characteristics of a database. The data model can be newly developed by using the data model graphical editor, or can be reverse-engineered from an existing database. The physical model is also used to provide metadata information to the other functional components of InfoSphere Warehouse, such as the SQL Warehousing tool, administration, and control.



Overview of InfoSphere Warehouse Advanced Edition

In this chapter we outline the components of IBM InfoSphere Warehouse Advanced Edition 10.1. We detail the main building blocks that make up the core product and provide a brief overview of the tooling components delivered within the licensing structure of InfoSphere Warehouse Advanced Edition.

Next, we discuss these tools in greater detail and give examples of their usage. We also explain the business benefits that can be derived from implementing these tools within your own environment.

Finally, we discuss the architecture of the main components and possible implementation methods.

2.1 InfoSphere Advanced Enterprise Edition components

InfoSphere Warehouse Advanced Edition is the perfect option for medium- to large-size organizations looking to implement a robust scalable enterprise data warehouse. This complete solution is composed of a number of IBM products, all of which can be purchased independently. However, when combined under the Advanced Edition umbrella, the full value proposition cannot be surpassed.

Individually, these IBM products offer a great solution for different sections of a data warehouse implementation. Combined within the Advanced Edition licence, though, they complement each other to provide not just an advanced and superior data warehouse solution, but one that address every aspect of a company's business intelligence needs.

The inclusion of the three InfoSphere warehouse packs that cover “Customer Insight,” “Market and Campaign Insight,” and “Supply Chain Insight” adds to the productivity by allowing rapid development and implementation of both the data warehouse and business intelligence solution.

The following sections provide a brief description of the individual IBM products brought together to create IBM InfoSphere Advanced Enterprise Edition.

Appendix A, “InfoSphere Warehouse Editions comparison” on page 477, provides a comparison of the components and additional packages included within the various versions of InfoSphere Warehouse product sets.

2.1.1 DB2 10 Enterprise Server Edition

At the heart of InfoSphere Warehouse is the IBM award-winning relational database DB2 for Linux, UNIX, and Windows Version 10. It provides a high performance, scalable, and secure relational database engine with many powerful features aimed at reducing development and storage costs, while boosting productivity. Of the many new features that can be found in DB2 10, the following are of great significance to its utilization within a data warehouse environment.

- ▶ Deep compression capabilities - Reduce the upward spiral on your storage costs without sacrificing performance.
- ▶ Multi-temperature storage management - Automatically utilize the best form of storage for your data needs (fast storage for important data, slower storage for historical or archival data) without needing vast amounts of costly manual administration.

- Time travel query: Reduce development time and improve the Extract, Transform, and Load (ETL) capability.

To address simplicity, the IBM DB2 relational database engine can be installed on a single hardware server, or as a partitioned relational database over many hardware servers aimed at providing high performance.

2.1.2 WebSphere Application Server

IBM WebSphere® Application Server is a scalable Java application server that is capable of being implemented in either a single or multiple instance configuration. It is therefore able to support varying workloads, from small departmental web applications to the demands of large scale, dynamic applications within an enterprise environment that require redundancy and failover capabilities.

The included WebSphere Application Server is required to host the InfoSphere Warehouse server application that provides the runtime elements of the Extract, Transform and Load, cubing services, and data mining elements of the data warehouse solution.

2.1.3 InfoSphere Warehouse Server

IBM InfoSphere Warehouse Server is the runtime and administration element of the complete InfoSphere Warehouse solution. The runtime elements are a J2EE application, hosted within the WebSphere Application Server.

One of the functions of the InfoSphere Warehouse runtime server, working in partnership with the DB2 database engine, is to provide the infrastructure to perform data movement and transformation (ETL) operations on the data contained within the database.

The runtime server also provides advanced analytics in the form of OLAP cubing services and data mining for both discovery and predictive analytics, along with text analytics for working with unstructured content.

For more detail about InfoSphere Warehouse Server, see 2.2, “InfoSphere Warehouse technical overview” on page 25.

2.1.4 InfoSphere Federation Server

InfoSphere Federation Server enables a data warehouse to reach beyond the confines of the local instance of the DB2 database. This function provides a consolidated view of the selected data taken from disparate database sources,

which can include other relational databases such as Oracle and Microsoft SQL Server. These remote data sources are made to appear as local tables within the DB2 instance, with InfoSphere Federation Server hiding the true complexity of connecting to these remote sources.

This functionality benefits the business by allowing the data warehouse to extend its profile to remote data sources, either in support of the ETL process or in providing real time views of the data while creating business reports.

2.1.5 InfoSphere Replication Server

InfoSphere Replication Server provides a solution to the need for data distribution, data synchronization, and data consolidation between the warehouse and applications within the enterprise environment. Both SQL-based and queue-based replication models are supported within the replication server, providing high-volume, low latency replication with conflict detection and resolution.

InfoSphere Replication Server includes support for partitioned database environments, automatically merging the changes captured from logs. Although data filtering can be used to ensure that only the data of interest is replicated, conflict detection and resolution features can also allow backup systems to be utilized to even out production workloads.

2.1.6 Advanced access control

Implementing a data warehouse solution demands many requirements, one of which is the security of the data contained within. Included within InfoSphere Warehouse Advanced Edition is advanced access control functionality.

Within an enterprise environment, with many users and application requesting access to the data contained within the warehouse, an enterprise security authentication process of validating these users and applications is required. IBM Advanced Access Control allows these users or applications to authenticate through an enterprise Lightweight Directory Access Protocol (LDAP) server, an advanced security process involving Kerberos, or using custom authentication plug-ins, thereby allowing security to be managed from a central location within the enterprise and driving down administration costs.

Row and Column access control with DB2 10 allows fine-grained control of what data within a table a user can access. The ability to restrict access to particular rows and to mask columns, while the user is unaware of the missing data, reduces both administration and storage, thus driving down costs.

2.1.7 IBM Cognos Business Intelligence for Reporting

Having a great data warehouse with many advanced features is irrelevant if you cannot access and present this information in a concise and business-like fashion. For this reason, InfoSphere Warehouse Advanced Edition includes Cognos 10 Business Intelligence (BI) for reporting.

From professional report writers who need to produce hundreds of reports, to rushed business users requiring an ad hoc report for the next sales meeting, Cognos 10 BI provides the tools needed to produce detailed and accurate reports, with access to near-real time data. The partnership between Cognos 10 BI and InfoSphere Warehouse server is fully demonstrated by the Cognos BI server's utilization of the OLAP cubing services, which provide a vast performance boost to the production of business reports.

Included within the InfoSphere Warehouse Advanced Edition are both the Cognos 10 BI server and Frameworks Client applications, allowing for the full spectrum of an enterprise BI solution.

2.1.8 IBM Data Studio

The latest version of IBM Data Studio has been radically enhanced from its humble beginnings to become an advanced solution for developing, managing, and monitoring heterogeneous database environments. Although it is aimed at increasing the productivity and collaboration of both database developers and administrators, its ease of use and common platform also greatly support the most casual of database users.

By utilizing the Eclipse Workbench Platform, IBM Data Studio becomes both an extendible environment (through the use of plug-ins) and a familiar Integrated Development Environment (IDE), not just for database developers but also for those working with Java, PHP, and Perl.

IBM Data Studio has been extended to include the functionality of both InfoSphere Optim Development Studio and Database Administrator, and Data Studio Health Monitor. This in turn removes the need for multiple tools and reduces complexity and support issues.

Advanced integration points with other IBM tools such as Rational® Application Developer, InfoSphere Data Architect, and InfoSphere Optim pureQuery™ Runtime support a collaborative working environment.

The latest DB2 capabilities including adaptive compression, time travel query, row and column access control, and multi-temperature data management are all supported within IBM Data Studio, as is the functionality to develop and test user

defined functions and stored procedures, tune SQL queries, create and edit physical data models, and provide multisystem deployment features. As of DB2 10.1, IBM Data Studio replaces DB2 Control Center, and is available at no cost as a download from the IBM website.

2.1.9 Optim Database Administrator

Originally provided as a stand-alone tool for the management of databases, and to enable database changes to be applied without disruption, Optim Database Administrator and its functionality have now been incorporated within Data Studio.

2.1.10 Optim Development Studio

Also originally provided as a separate tool for the design, development, and deployment of SQL scripts, stored procedures, and data-related web and Java applications, Optim Development Studio has now been incorporated within Data Studio.

2.1.11 InfoSphere Warehouse Client - IBM Design Studio and the wh command

IBM Design Studio is an integrated development environment, based on the original IBM Rational Data Architect product, that brings to InfoSphere Warehouse a set of rich development tools needed in the creation of BI solutions. Design Studio integrates the tasks required in the development of solutions into a single graphical user interface, based on the popular Eclipse Workbench. The tool provides the following functionality within a common, easy-to-use interface:

- ▶ Physical data modelling
- ▶ ETL flow development and testing
- ▶ OLAP cube modelling and services
- ▶ Data mining modelling
- ▶ Text analytics

By utilizing Design Studio, developers can connect to multiple databases as either source or targets in support of their operations, work with physical data models, created as new, or by reverse-engineering from a DDL file or an actual database. From within the same tool, they can develop and test ETL flows, data mining flows, develop OLAP cubes, and prepare warehouse applications ready for deployment to the runtime elements of the InfoSphere Warehouse.

IBM Design Studio is provided as one component of InfoSphere Warehouse Client. The warehouse command line tool **wh** provides an alternative interface to the runtime elements of InfoSphere Warehouse, to that of a web browser.

This alternative interface provides developers and administrators with a fast and efficient method of configuring, scheduling, and running tasks within the InfoSphere warehouse runtime, from the command line of the native operating system.

2.1.12 InfoSphere Data Architect

InfoSphere Data Architect is a collaborative integrated data design environment that is aimed at the higher role of the solution architect who is required to discover and model the warehouse solution while taking into account its relationship within the wider corporate environment.

This tool provides architects with the functions needed to create and visualize a logical model that addresses the BI requirements. Simultaneously, it provides the facilities needed to transform this model into a physical data model and from there to a DDL script that can be easily and rapidly deployed.

The tool can also be used to leverage standard deployment models, develop collaboration between architects, developers, and administrators, and maintain data lifecycle policies. Data models developed within the tool can be exchanged with other IBM development tools and applications such as Cognos Frameworks. They can also be published into an HTML format for hosting on a website in support of audit and collaboration purposes.

The physical data models developed within InfoSphere Data Architect have a common format that is shared between Data Studio and Design Studio, as are the interfaces within these IDEs. This helps developers by providing a consistent look and feel, and can accelerate the development process.

When used in conjunction with the InfoSphere Warehouse Packs, InfoSphere Data Architect allows designers to enhance the standard included models to develop and deploy custom models tailored to their specific industry or business environment.

For your awareness: InfoSphere Data Architect was formerly known as IBM Rational Data Architect.

2.1.13 InfoSphere Optim Query Workload Tuner

InfoSphere Optim Query Workload Tuner is a utility that provides database administrators and developers with the required tools to capture and analyze either individual or entire workloads of SQL queries from a variety of sources. When working in partnership with Data Studio or Design Studio, these SQL queries can be analyzed, with easy-to-use wizards providing detailed and meaningful statistics, execution visualization, advice about which indexes to apply, and expert tuning recommendations.

This utility does not stop at simply improving the performance of SQL queries, however. It also enhances the efficiency of developers and database administrators by helping to provide expert advice on improving database design.

InfoSphere Optim Query Workload Tuner must be implemented within one of the development client tools such as Data Studio and Design Studio, or with InfoSphere Optim Performance Manager.

2.1.14 InfoSphere Optim Performance Manager Extended Edition

InfoSphere Optim Performance Manager provides a comprehensive and real-time monitoring solution for DB2 and InfoSphere Warehouse database implementations. Its easy-to-use web interface provides database administrators and IT support staff with the health and status of multiple database environments.

InfoSphere Optim Performance Manager allows you to set proactive alerts, perform deep-dive problem determination, view both real time and historical performance data, monitor and report usage trend analysis, and proactively manage your database environment.

Rapid deployment can be achieved because end-to-end transaction monitoring for many business applications has been preconfigured and is ready for immediate use. When combined with Optim Query workload Tuner, SQL queries can be captured and analyzed, with performance recommendations being developed.

InfoSphere Optim Performance Manager Extended Edition extends its functionality beyond performance monitoring to include the ability to set and monitor response time objectives for database workloads. It is implemented as a web-based application, with sophisticated interfaces into the data sources it monitors.

2.1.15 InfoSphere Optim High Performance Unload

InfoSphere Optim High Performance Unload is the ultimate tool for unloading, extracting, and repartitioning data from DB2 databases. Data migrations can now be performed with minimal effort, a high degree of accuracy, and within shorter time frames.

The ability to apply filters to the data source allows precise and targeted unloads, while allowing the database to remain online and available to applications. This utility can also extract tables and data from incremental, delta, and full backup images, allowing rapid data restores in response to accidental table drops and data deletion.

With the ability to bypass the DB2 engine and work directly on the data stored on disk, there is minimal impact on database performance. Repartitioning, an important element of data warehousing, is supported by performing both unloads and splits within a single operation.

Note: Optim High Performance Unload is executed exclusively from the command line.

2.1.16 InfoSphere Optim Configuration Manager

Driven by the need to eliminate costly outages caused by human mistakes, and to bring order to their database environments, many business have developed costly in-house tools for the management of their database environments. InfoSphere Optim Configuration Manager provides an improved solution to this dilemma.

This tool provides a means for gathering information about database and client configurations and centralizing this information, to create a more manageable and effective infrastructure. InfoSphere Optim Configuration Manager also tracks changes to database and client configurations, thereby helping to resolve the causes of performance degradation, or help in delivering a constant database structure over multiple database implementations. It can also be used in the delivery of database migrations to new hardware or from earlier versions of DB2.

Administrators can modify properties of deployed database clients to improve database transaction performance, enforce security and privacy controls, and allocate data clients among data servers to balance workloads.

From InfoSphere Optim Configuration Manager's web interface, database administrators can enforce common client configurations, while also tracking underutilized storage resources. When combined with DB2 compression and

multi-temperature storage functionality, InfoSphere Optim Configuration Manager allows administrators to automate storage procedures, thus reducing storage costs and enhancing savings to the business.

2.1.17 DB2 Recovery Expert

DB2 Recovery Expert is a tool designed to help reduce the time taken by database administrators and support staff in recovering damaged or dropped database objects from within production database, quickly and safely without the need for a full database recovery. By analyzing log transactions, DBAs can isolate accidental or incorrect changes made to tables and objects and undo these changes with minimal impact to the business.

DB2 Recovery Expert can also be applied to reduce log transactions and only keep itemized, by table, those subsets that assist in determining the best possible path and those resources required for a speedy recovery. Administrators can monitor critical tables, and view changes made by date, user, table, and other criteria.

Utilizing a simple web interface, administrators can rapidly implement recovery strategies to best protect business database assets.

2.1.18 DB2 Merge Backup

DB2 is designed to minimize the impact of backups and shorten recovery times on production servers. Due to time constraints, DBAs often can create only incremental backups of their databases. This process has the benefit of only needing short periods of time, allowing for a fast turnaround on production databases. However, restoring a database from such a backup strategy can be time-consuming because each incremental backup has to be restored.

DB2 Merge Backup can merge several incremental backups with the latest full backup to create a new, more current full backup. This backup can then be used to merge with the next day's incremental backup to produce the next full backup.

Note: DB2 Merge Backup is implemented from a command line interface.

2.1.19 InfoSphere Warehouse Packs

InfoSphere Warehouse Packs are specially created “enablers” with the goal of helping customers implement rapid solutions to their data warehouse requirements. They have been developed from years of experience within IBM

Service teams, who have helped implement many successful data warehouse solutions for their customers.

These offerings contain physical data models and sample reports based on specific business issues (Customer Insight, Supply Chain Insight, and Market & Campaign Insight) that help drastically reduce project implementation times, costs, and the risk involved for customers, as compared with normal custom data warehouse projects. These packs cover the following areas:

- ▶ InfoSphere Warehouse Pack for Customer Insight enables organizations to gather, organize, and use relevant information about current customers, thus bringing them closer to their customers.
- ▶ InfoSphere Warehouse Pack for Market and Campaign Insight enables organizations to gather, organize, and use relevant information about the overall market and how to acquire new customers.
- ▶ InfoSphere Warehouse Pack for Supply Chain Insight enables organizations to gather, organize, and use relevant information about their supply chain and associated performance.

2.2 InfoSphere Warehouse technical overview

InfoSphere Warehouse Advanced Edition aims to bring together all the components required for a successful, cost-efficient data warehouse solution. The components range from the development tools needed to create your ETL, OLAP, and data mining, to the BI tools used in understanding your market.

InfoSphere Warehouse Advanced Edition also offers the tools needed to manage your backup strategy, drive consistency across your business, and bring out the best performance of your data warehouse and the applications connected to it.

In this section, we provide insight into the technical architecture of InfoSphere Warehouse, taking into account the development and runtime components of the core product. Additionally, we include the technical architecture of the Cognos 10 reporting server and its relationship within the complete solution.

Because InfoSphere Warehouse Advanced Enterprise Edition also includes a number of Optim products, we include insight into Optim architecture to create a complete overview of an implemented data warehouse solution. For completeness, we briefly discuss the various metadata databases required within an implementation of an InfoSphere Warehouse and the Optim tool set.

2.2.1 InfoSphere Warehouse core architecture

At the center of the InfoSphere Warehouse lies the DB2 relational database engine, which provides not only a repository for user data, but also the infrastructure to support the many functional operations performed on the data. Together with the InfoSphere Warehouse application hosted within a WebSphere Application Server, these elements combine to form the runtime component of an InfoSphere warehouse solution.

A number of client products, Data Studio, Design Studio, and web browser, provide the development tools and administration components required in the support of these runtime elements. Figure 2-1 shows the complete functional component architecture of InfoSphere Warehouse.

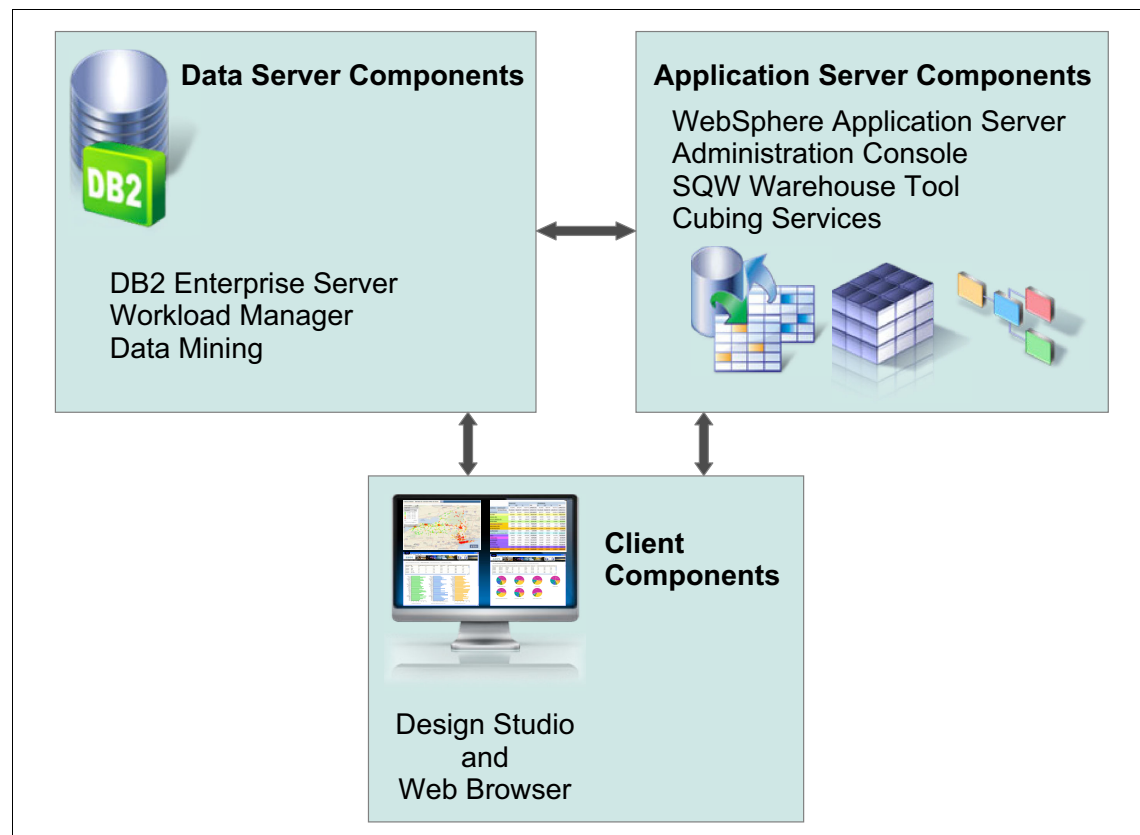


Figure 2-1 Functional component architecture of InfoSphere Warehouse

The instance of the DB2 relational database at the core of the data warehouse can be configured either as a single or multiple partitioned database, and installed on a single hardware server or on numerous hardware servers, as detailed in 2.3.1, “Scenarios for implementing InfoSphere Warehouse core products” on page 60.

This DB2 flexibility results in significant power being available to the main repository of your warehouse data, often called the *execution database*. In addition to the main data repository and execution database, the same DB2 instance hosts two additional databases. These much smaller databases contain the metadata required by both the InfoSphere Warehouse runtime and the Cognos BI server applications. Figure 2-2 details the DB2 instance and the default application databases.

Name change: In InfoSphere Warehouse 10.0, the metadata database is called DDSDB. In earlier versions of InfoSphere Warehouse, the metadata database was known as SQWCTRL.

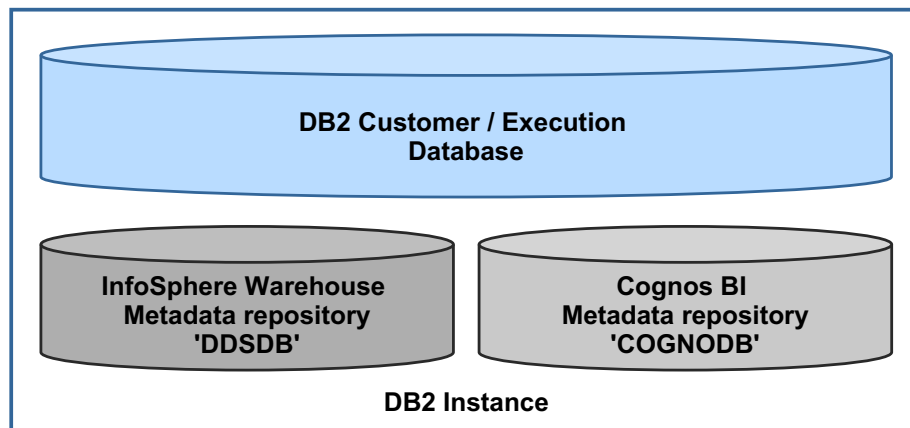


Figure 2-2 Databases contained within the DB2 instance

The DDSDB metadata database holds information relating to the InfoSphere Warehouse runtime component including such details as:

- ▶ Database connection and JDBC drivers
- ▶ SQL Warehousing (SQW) applications, including control flow, job variables, and system resources
- ▶ Scheduling details of the SQW jobs
- ▶ The progress of SQW job instances and their statistics, time taken to execute, completed successfully or not, and logs

The COGNODB metadata database contains the information needed by the Cognos BI server and Cognos Frameworks. This includes such items as the reports created, data source connections, and the modelling information used within Cognos Frameworks.

There should never be a reason for direct connection to either of these metadata databases by users or administrators, with only the respective runtime applications requiring to make connections to access the data within. With this in mind, both products supply tools for the administration of these metadata databases because without them, the runtime element of InfoSphere Warehouse and the Cognos BI server will not function.

Storing important business data is not the only function that DB2 has to perform. A major part of any data warehouse is the requirement to perform ETL processes upon the contained data. It is because of this requirement that the full power, performance, and functionality of DB2 can be applied.

The second major component of InfoSphere Data Warehouse is the runtime application component. This enterprise Java application performs a number of important functions within the complete solution, including:

- ▶ The administration console for the InfoSphere Warehouse solution
- ▶ Storing, executing, and managing the ETL processes
- ▶ Storing and managing the cubing services
- ▶ Storing and managing the data mining services

Figure 2-3 shows the InfoSphere Warehouse Administration Console.

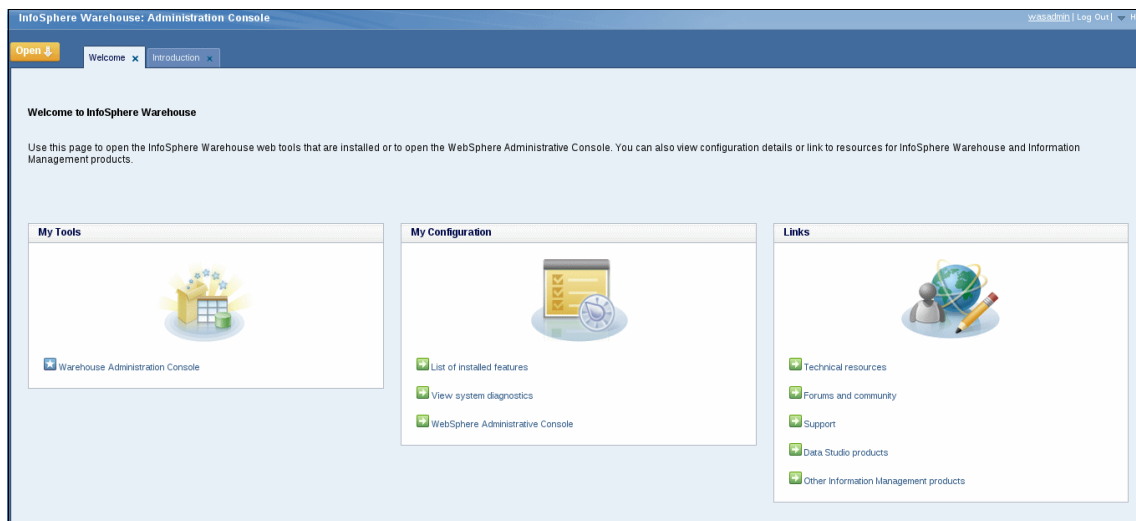


Figure 2-3 InfoSphere Warehouse Administration Console

Because the runtime and administration console component is a J2EE Java application, it requires hosting within an application server. For this reason, InfoSphere Warehouse comes supplied with a licensed copy of IBM WebSphere Application Server.

During the installation of the InfoSphere Warehouse product, its installation tool configures a single instance of a WebSphere Application server that can be implemented in a number of configurations.

This installation tool then installs the InfoSphere Warehouse runtime component (which is a Java application labeled “IBMDDataToolsWeb”) into this instance, as shown in Figure 2-4.

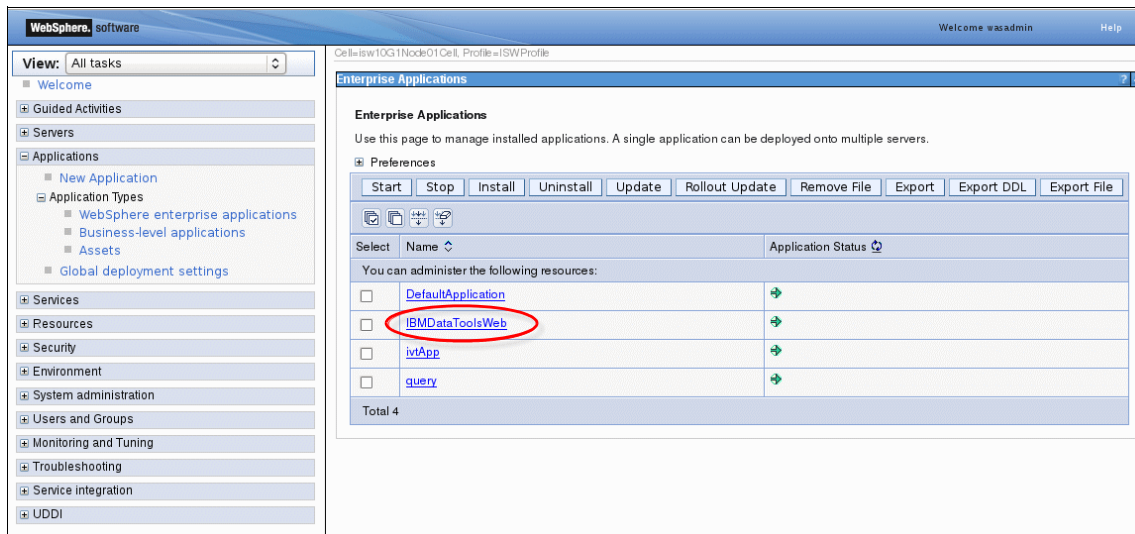


Figure 2-4 InfoSphere warehouse runtime J2EE application IBMDDataToolsWeb

The InfoSphere Warehouse administration console has a web-based interface that allows any browser to be used in the configuration and management of the functional elements of the runtime environment. Thus, a single browser can be utilized to handle all production, test, and development environments.

To connect to the InfoSphere administration console, point a browser at this address, where Hostname is the IP address of where the InfoSphere Warehouse application is hosted, 9080 is the default port used by the application, and /ibm/console is the location of the initial login window:

`http://Hostname:9080/ibm/console`

Application users can be defined during the initial product install by using the WebSphere Administrative Console, or as the defined users of the native operating system.

After logging into the console with a valid user, the initial Welcome window is presented as shown in Figure 2-3 on page 28. A user can enter the Warehouse Management Console through the My Tools section, or launch the WebSphere administrative console through the My Configuration section.

The Links section of the Welcome panel contains web links to help and to IBM educational website resources.

The “Open” button is located on the top left corner of the Welcome window. When Open is selected, it displays the main menu to various sections of the console. Figure 2-5 on page 31 shows these sections:

- ▶ InfoSphere Warehouse
 - Introduction: The initial help information
 - SQL Warehousing: For ETL application installation, management, and execution
 - Cubing Services: For the management of cubing services
 - Mining: For the management of mining flows
- ▶ Common Services: shared facilities among the various elements of SQL, cubing, and mining
 - Manage Connections: For defining connections to different data sources
 - Manage Data Server Drivers: For defining new JDBC drivers to different types of data sources
 - Manage System Resources: For defining system resources such as variables used within flows, servers, and their host names
 - Configuration: For configuring elements of the system, that is, logging locations and email services
 - Manage Logs: For viewing logs from SQL execution instances and cubing services

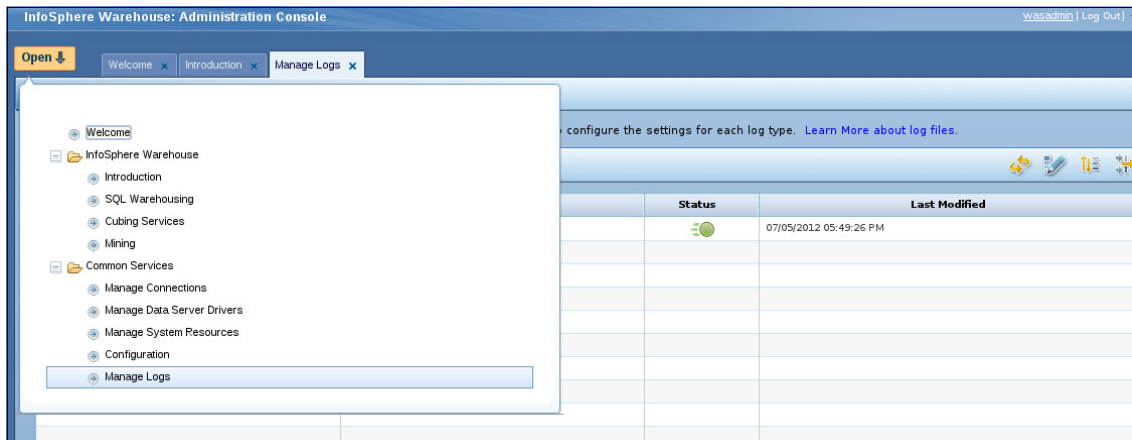


Figure 2-5 Menu options within the InfoSphere Warehouse Administration Console

One of the functions of the runtime element of InfoSphere Warehouse is to perform the Extract, Transform, Load (ETL) operations on the data contained within the execution database. These operations are performed by SQW warehousing applications, which consist of control flows and data flows that have been created using the SQW development tool Design Studio.

From Design Studio, data flows can be created using building blocks called “operators.” Each operator performs a specific SQL operation or DB2 command. At a higher level, these data flows are then encompassed into what are called control flows. Control flows are then used to sequence one or multiple data flows, along with additional control flow operators.

The sequence of these data flows can be configured to execute either in parallel for greater performance, or in serial when one data flow is dependent on the operations of another. Along with required start and stop operators, control flows have additional operators for performing common DB2 table functions, such as runstats, reorgs, row compression, and roll-in and roll-out of table partitions. There are also auxiliary operators that can perform file write, read, and file wait operations, along with emailing, DB2 stored procedure calls, and custom SQL commands.

This visual building process allows complex ETL operations to be developed. Figure 2-6 shows a common control flow being developed within Design Studio.

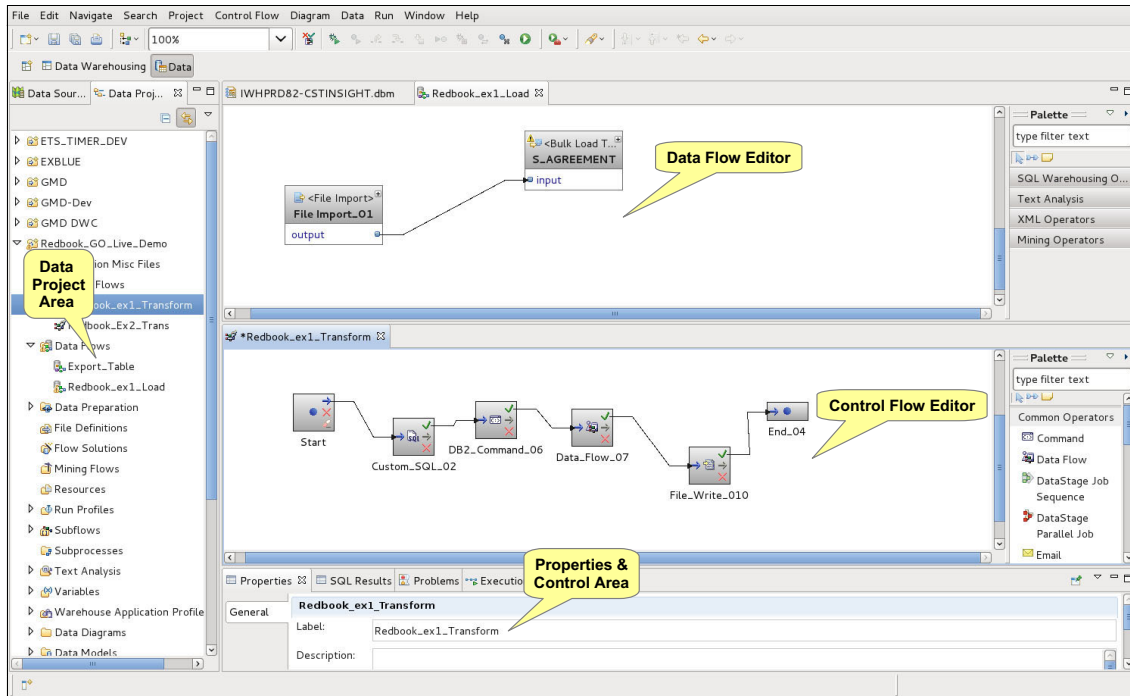


Figure 2-6 Example control flow being developed and tested within Design Studio

From within Design Studio, SQW data and control flows can also be tested and debugged against real databases, then grouped into a compressed file, in what is termed a warehouse SQW application. These SQW applications are then deployed through the administration console into the SQW runtime element called the Data Integration Service (DIS). Information about the flow, such as its name, comments, the resource it uses, data connections, and variables, are also stored at the same time into the metadata repository.

An SQW data and control flow are in fact Java code files that include embedded SQL calls. These Java files are then bundled into a compressed file and loaded into the DIS unit. The DIS unit then manages the execution of these individual Java files (SQW control and data flows) and the embedded SQL code.

The execution of one of these Java files, or in InfoSphere Warehouse terms, instance execution of a control flow, can be instigated using the administration console, either as a one-time manual process or automatically at predefined time intervals. These scheduled times can be defined as regular time intervals or specific times of the minute, hour, day, week, or month.

On execution of a control flow, the DIS retrieves all the required information from the metadata repository and then performs the necessary SQL commands and functions that represent the original operators used within Design Studio to build the data and control flows. In the DIS, these SQL commands and functions, just as the operators do in Design Studio, can perform work on both local and remote databases along with tasks on the native local operating system.

During execution, the DIS monitors the SQL commands and functions that make up the data and control flows, along with any messages related to their execution, and stores these within log files. The DIS also gathers runtime and job completion statistics, all of which are stored back within the metadata repository. The logs, messages, and statics can then be viewed from within the administration console. Figure 2-7 shows the relationship of the DIS unit within the system.

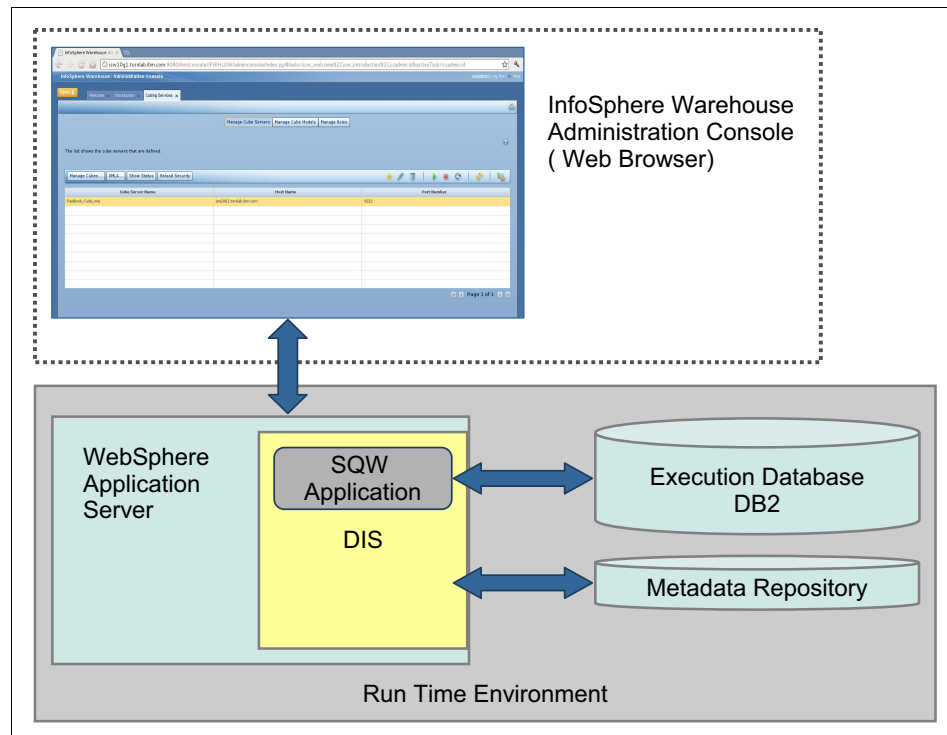


Figure 2-7 DIS architecture

The management and execution of SQW flows to perform ETL tasks is only one of three major functions implemented as part of the InfoSphere Warehouse application hosted within the WebSphere Application Server. The second major function of this application is the management of OLAP Cubing Services servers.

A cubing services server, which for simplicity is called a cube server, is the runtime element of the Cubing process. This cube server is an independent Java process that hosts the various cubes, receives incoming connection and query requests, processes the requests, constructs the result sets, and returns them to the calling application. This Java process executes independently of the installed WebSphere Application Server, but is required to reside on the same physical server, to enable being managed by the administration console application.

Figure 2-8 details the relationship between the InfoSphere Warehouse administration console, the WebSphere Application Server, and a cubing server hosting a number of cubes.

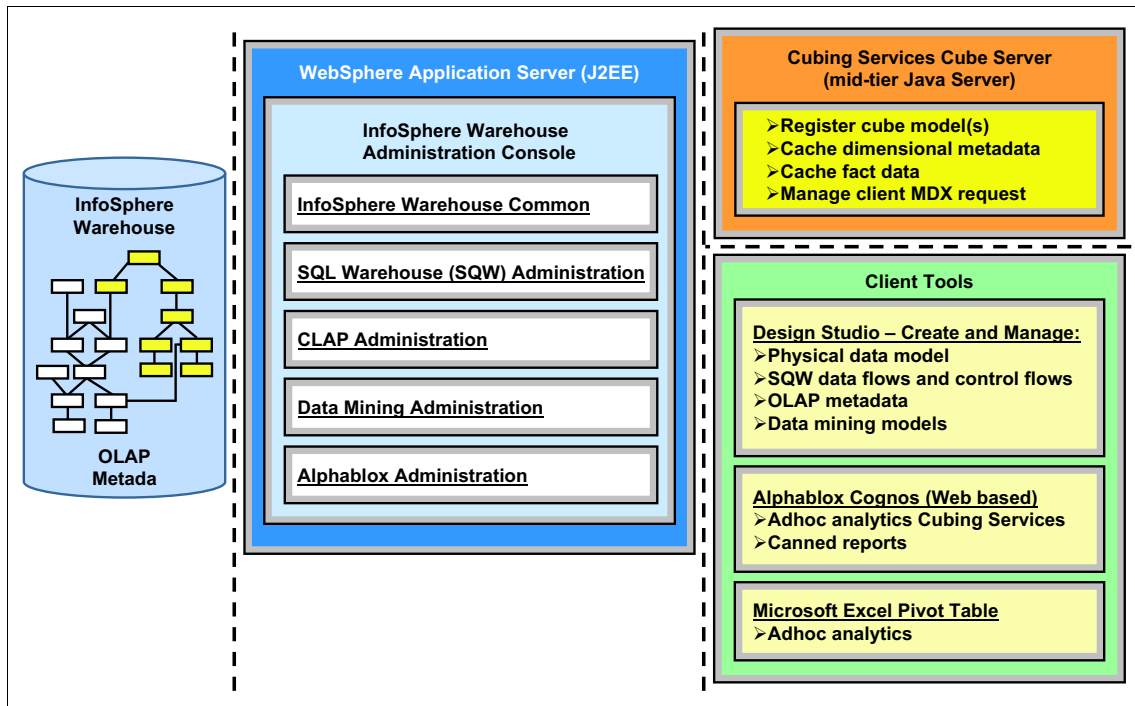


Figure 2-8 Relationship between InfoSphere Warehouse administration console and a cube server

Before a cube can be implemented within a cubing server, it first has to be defined and tested. This job again falls to the development tool Design Studio,

within which a cube can be first modeled, then created, and finally tested. Figure 2-9 shows an example cube model within Design Studio.

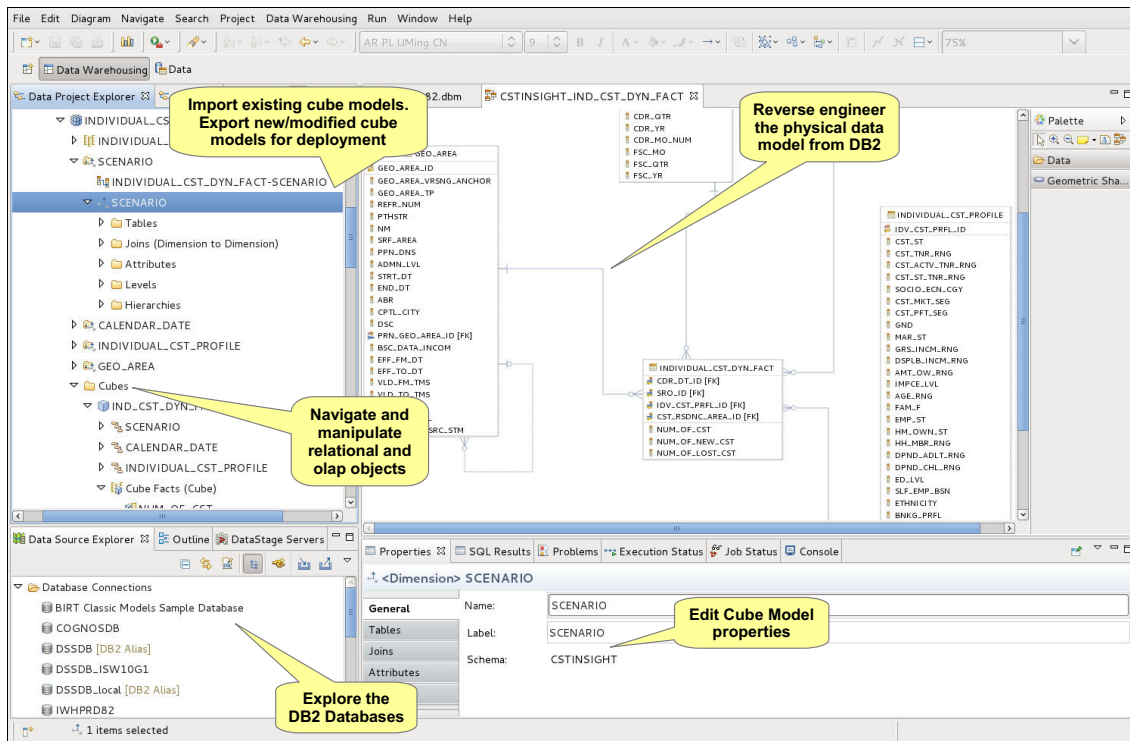


Figure 2-9 Cube model within Design Studio

From within Design Studio, when a cube model has been successfully implemented, it can then be deployed to the InfoSphere Warehouse server. This process is performed by exporting the cube model directly into the OLAP metadata repository, which is a sub-part of the InfoSphere Warehouse metadata database.

To deploy the cube model, follow these steps:

1. From within Design Studio, select and right-click the cube model, then select **Deploy OLAP Objects**. This starts the wizard to export the cube model to the metadata repository and create the required XML file.
2. On the warehouse administration console, under the Cubing Services tab, select **Manage Cube Models** to start the process of importing the cube model.

3. Start the import wizard by selecting **Import Cube Model**.

The wizard, through an XML file created in Design Studio, retrieves the definition of cube model within the repository, and lists within the Cube Models table.

4. Create a cube server from the administration console.

From Manage Cube Servers tab, click **Create** to start the wizard. The wizard asks a number of question such as which network port address has to be assigned to the cube server, in our case we used port 9212. Multiple cube servers maybe created, but each needs their own network port address. At this point, a cube still does not exist, until it has been instantiated in a cube server.

5. From Manage Cube tab, select **Manage Cube Servers**.

The wizard provides a panel (Figure 2-10) that can be used to assign the already imported cube model into the Cube Server.

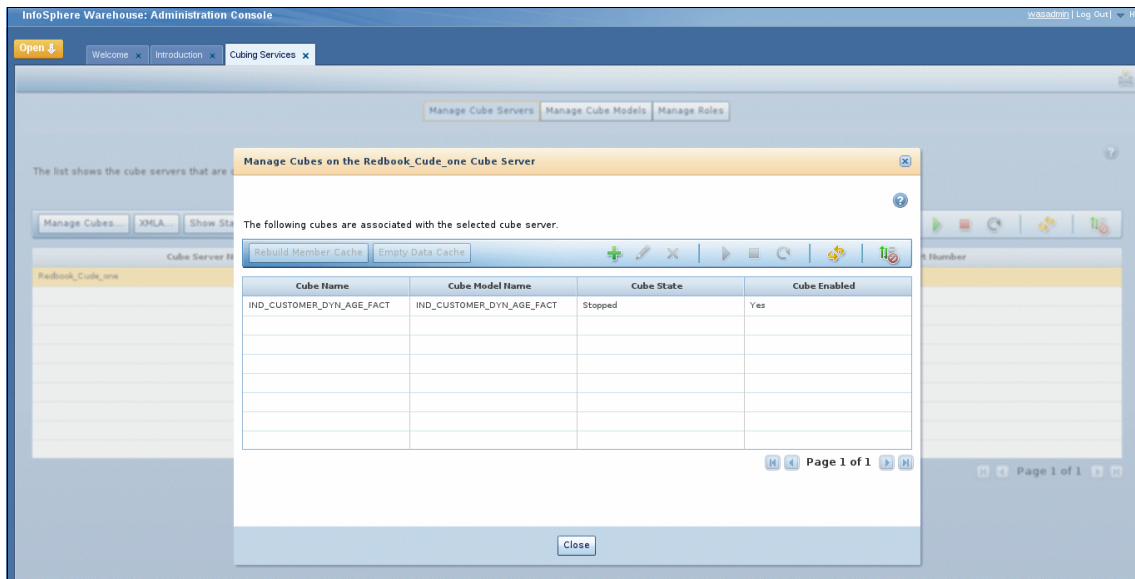


Figure 2-10 Adding a cube to a cube server

The cube server maintains a metadata cache for each running cube that contains member information for each member of every dimension. These member arrays essentially provide a multidimensional address to access each data cell in the cube. The cube server also maintains a dynamic data cache to manage cell data that has been loaded or calculated recently by the cube server. The cube server does not cache all cell data, only a small subset that is currently active.

Other than the data in the dynamic cache, the Cubing server does not persist any of its own cell data. Instead, it accesses the data in the database. This is why it is often referred to in terms of “No Copy” analytics.

To enhance performance of the cube server, the DB2 server might have one or more MQTs implemented. A cube server communicates with the DB2 database through a JDBC connection using standard SQL.

The client tools and applications, such as IBM Cognos BI and Microsoft Excel, talk to a cube server through industry standard APIs for OLAP:

- ▶ XML for Analysis (XMLA) is an XML-based web services API used by tools such as IBM Cognos BI.
- ▶ OLE DB for OLAP (ODBO) is an ActiveX-based Microsoft standard that is based on OLE DB and used by applications such as Microsoft Excel.

These client tools and applications, when connecting to the cube server, use standard Describe functions in these APIs to discover the available cubes and their metadata within the cube server. To query the cube data, both XMLA and ODBO API clients use an industry standard OLAP query language called Multidimensional Expressions (MDX).

When a cube server is started by the cube Service manager within the administration console, the cube server initializes both the member and data cache areas. The cube server sends SQL queries to the Cubing Services metadata repository that is hosted within the DB2 database, to discover which cubes are running on this server. It then loads all of the member metadata into the member cache. At the same time, a small item of information for each and every member of each dimension is loaded into the member cache.

The cube server also initializes the calculation engine work area in memory, where runtime operations are executed, such as sorts and calculations. The data cache is then initialized for the cube. The data cache is addressable through the member arrays in the member cache. The cube server is now ready to start receiving data requests from clients.

When an MDX query request is received from a client application like IBM Cognos, the query is parsed for correctness and analyzed against the metadata information for the cube. The cube engine evaluates the query and determines the size and shape of the result set. It then constructs an empty result set in the work area, though it does not have any data in the cells yet.

The calculation engine next requests data that it needs to populate the result cells from the data cache. Because this is the first such request, the data cache is empty, so no data can be returned. Due to this empty data cache, the cube engine sends SQL queries to the database to obtain the data necessary to

process the query and construct the result set. This is accomplished through one or more queries. The SQL results are then loaded into the data cache in the form of cube subsets or “cubelets”. These cubelets are used to maintain the cache data as efficiently as possible.

The final result sets are then constructed from the data of one or more cubelets. This OLAP result set then is returned to the query client.

When new queries are received and processed, the cube engine continuously seeks to use the data from the cubelets in the cache. This avoids the need for new SQL queries to the database, which are themselves time consuming. The cube engine uses cubelets when available to satisfy all or even just part of the query data requirements.

The cache size limits are configurable. When the thresholds are reached, the least recently used cubelets are flushed from the cache to make room for new cubelets. This data cache can be preloaded or seeded at startup by the specification of a seed query defined in the cube server metadata. Changes to data in the database require flushing of the data cache. Figure 2-11 on page 39

illustrates the relationship between the query clients, the cube server, and the metadata repository.

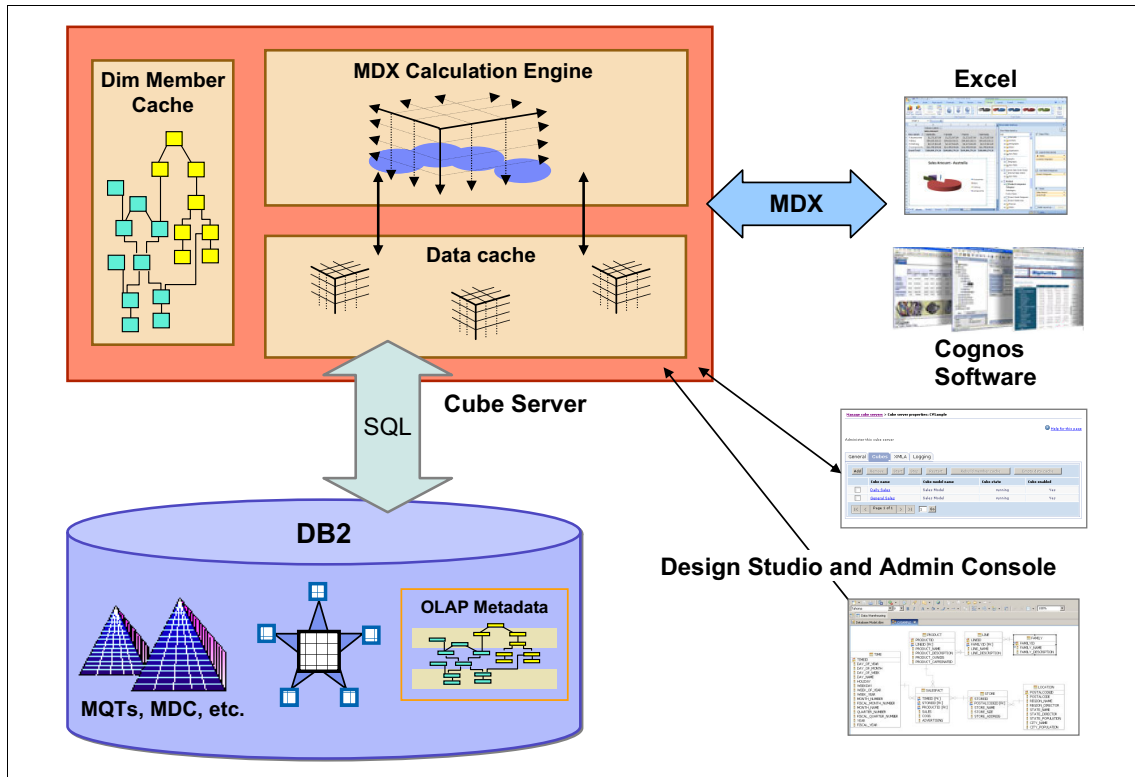


Figure 2-11 Relationship between query agents and cube server

The final function of the InfoSphere Warehouse application hosted within the WebSphere Application Server is performing data mining operations.

Data mining is about discovering patterns and relationships hidden in data. The in-database data mining capabilities are integrated with existing systems to provide scalable, high performing predictive and pattern analysis without moving your data into proprietary data mining platforms. The IBM DB2 relational database contains database extensions that perform data mining functions. Data mining is composed of three steps:

- **Modeling**

Modeling is the process of analyzing existing data outcome. This process starts by applying a series of mathematical functions to the historical data to derive a set of business rules that are collected together to form a data mining model. These models are represented using the Predictive Model Markup

Language (PMML), which can then be exchanged with other IBM modeling tools such as SPSS®. Design Studio provides a graphical method for working with data mining models. The SQL API for modeling provides a number of mining functions by which these PMML models can be stored within DB2 tables.

- Scoring

Scoring is the process of taking new data and applying it to an existing model to predict the outcome. The business rules that make up the data mining model are then applied to new data sets to determine the appropriate predicted outcome. The results of which are then stored back within the DB2 tables. There are two types of scoring, Real-time scoring and Table scoring.

- Visualization

Visualization is the graphical interface used to view and evaluate the mining results. Applications can call these visualizers to present model results, or you can deploy the visualizers as applets in a web browser for ready dissemination.

For a database to be used for data mining, it must be “enabled.” Enablement is the process of adding a series of stored procedures, user-defined functions, and methods to the database that become available for mining activities against the database. A database can be enabled for Data Mining in one of three ways:

- From a DB2 command line
- From the Design Studio, used with development databases
- From the Administration Console, used with production databases

Within InfoSphere Warehouse, the data mining process starts with Design Studio, which provides a set of tools to perform a range of mining tasks. These include methods to sample and explore the data within the database, and tools to obtain an overview of the statistics and distributions of the data within a table. These statistics are performed by the extensions within the database. Therefore, large data sets can be computed because no data transfers have to take place. These statistics also remain within the database, and can be retrieved by any

client. Figure 2-12 illustrates an example of the statistics that can be applied to a table.

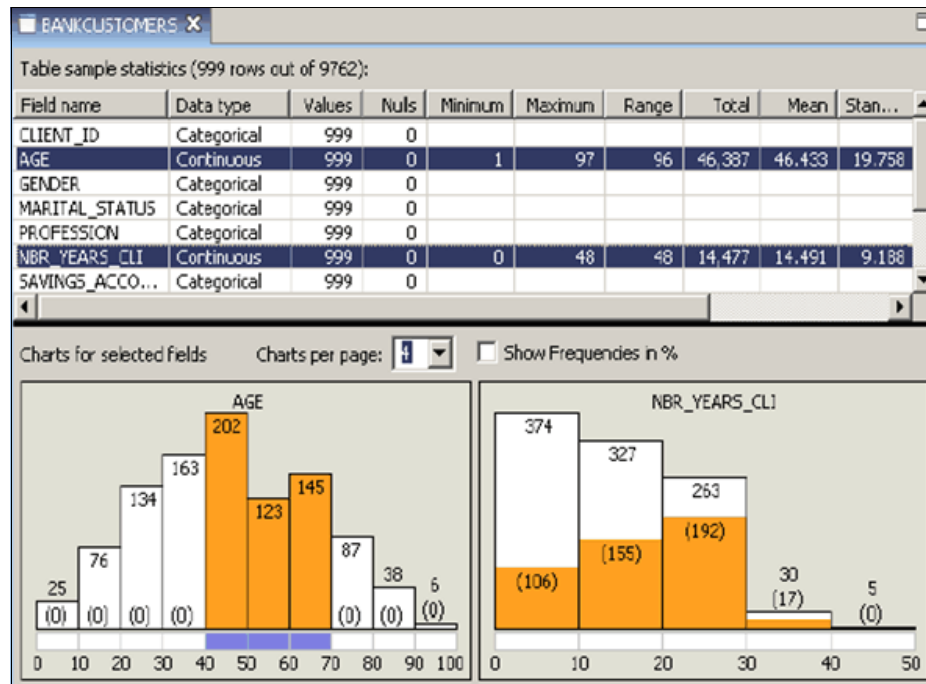


Figure 2-12 Table statistics for Mining Operations

You can develop mining flows using a similar visual building tool that is used in developing SQW data flows. These mining flows can also be executed against the database from within Design Studio.

Figure 2-13 on page 42 shows a data mining flow example within Design Studio. In the same way that data flows are added to the control flows to allow for their

deployment, mining flows can also be added to the control flows. The deployment process with the administration console is also similar.

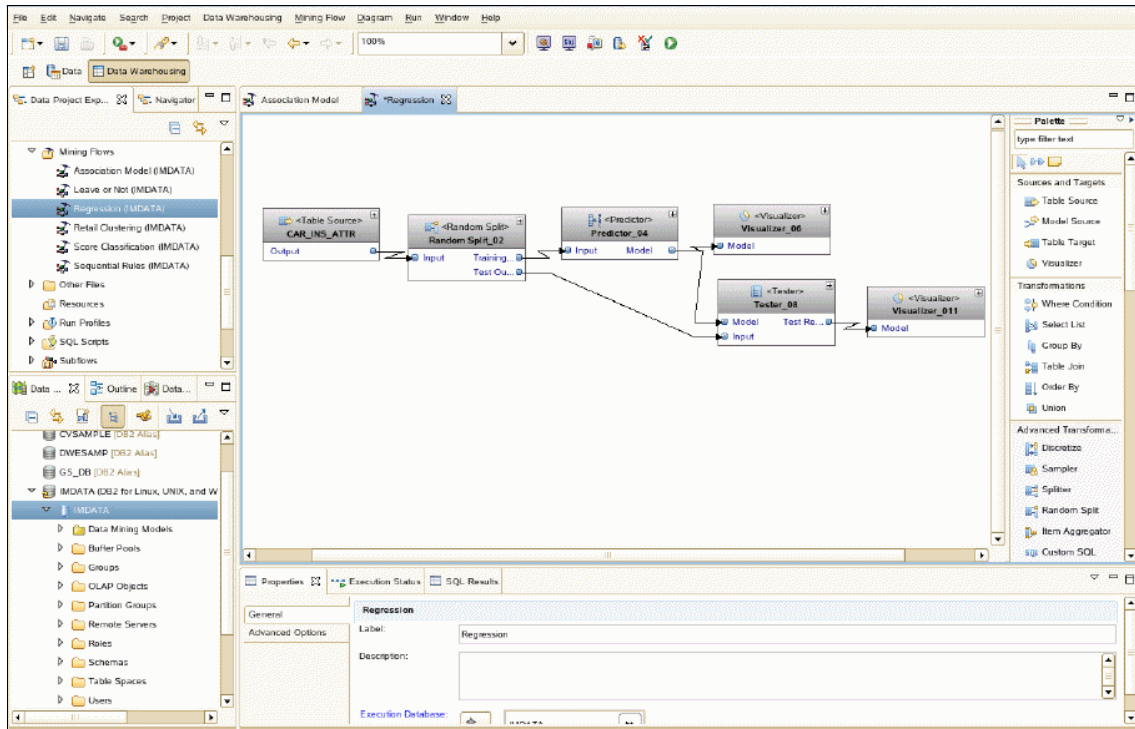


Figure 2-13 Example data mining flow in Design Studio

Finally, you can use the Design Studio to manage and visualize the data mining models residing in the database.

2.2.2 Overview of IBM Cognos Business Intelligence

In this section we provide a brief overview of the technical components of IBM Cognos Business Intelligence. For a more detailed description, see *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

The IBM Cognos Platform is built on a web-based, service-oriented architecture (SOA) that is designed for scalability, availability, and openness. This n-tiered architecture is made up of three server tiers:

- ▶ The web tier
- ▶ The application tier
- ▶ The data tier

The IBM Cognos Platform provides optimized access to all data sources, including relational data sources and online analytical processing (OLAP), with a single query service. In addition, this query service understands and takes advantage of the data source strength by using a combination of open standards such as SQL99, native SQL, and native MDX to optimize data retrieval for all these different data providers. The IBM Cognos BI user interfaces are accessed through the web tier. Figure 2-14 illustrates a typical deployment of the IBM Cognos Platform.

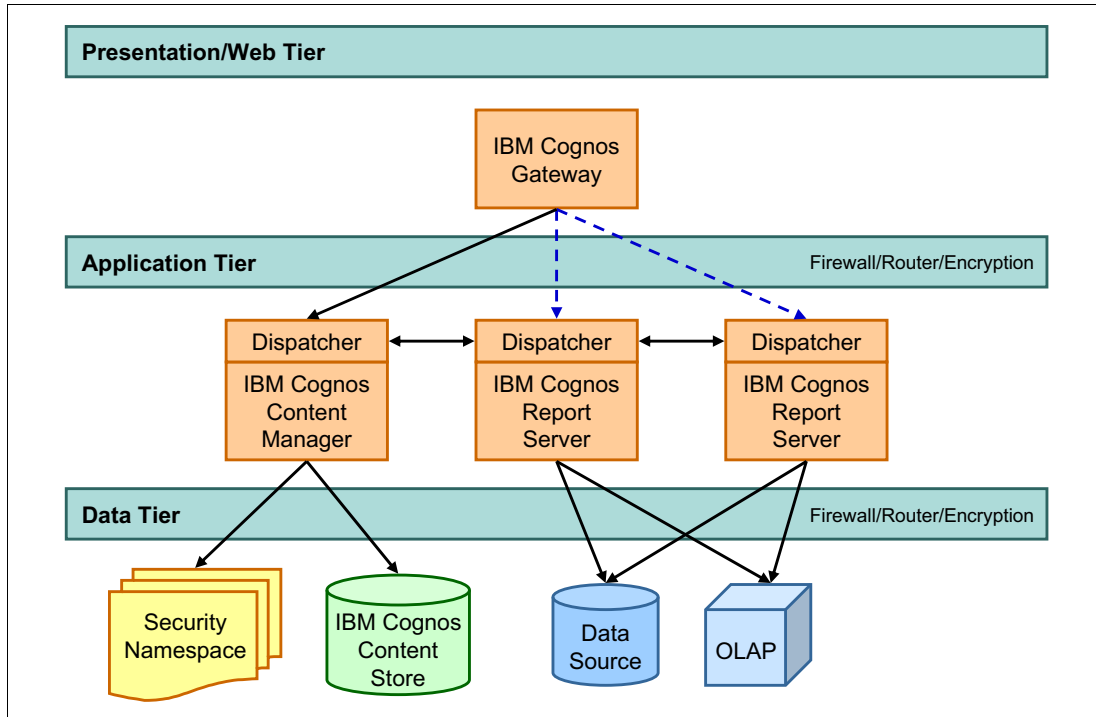


Figure 2-14 Typical Cognos BI deployment

The web tier provides user session connectivity to IBM Cognos BI applications. The IBM Cognos components that fulfill this role are referred to as the IBM Cognos Gateway. This component manages all web communications for the IBM Cognos Platform. The workload on the IBM Cognos Gateway server requires minimal processing resources, and for high availability or scalability requirements, multiple gateways can be implemented.

The application tier for the IBM Cognos Platform is made up of three main server components:

- IBM Cognos Dispatcher

- ▶ IBM Cognos Report Server
- ▶ IBM Cognos Content Manager

IBM Cognos Dispatcher performs the load balancing of requests at the application tier. The IBM Cognos Dispatcher component is a lightweight Java servlet that manages (and provides communication between) application services.

The main service that is responsible for application-tier processing is the report or query service. IBM Cognos Dispatcher starts IBM Cognos Report Server processes dynamically as needed to handle the request load. An administrator can specify the maximum number of processes that these services can start, and the minimum number of processes that are to be running at non-peak times.

IBM Cognos Content Manager is the IBM Cognos Platform service that manages the storage of the following customer application data:

- ▶ Security settings and configurations
- ▶ Server configuration settings
- ▶ Packages
- ▶ Dashboards
- ▶ Metrics
- ▶ Report specifications
- ▶ Report output

The IBM Cognos Content Manager function is to publish models, retrieve or store report specifications, handle scheduling information, and manage the IBM Cognos security name space. Information is maintained in a metadata database that is referred to as the content store database; within an InfoSphere Warehouse environment it is commonly given the database name “COGNODB” within the DB2 instance. A minimum of one IBM Cognos Content Manager service is required for each IBM Cognos Platform implementation.

The IBM Cognos BI data tier contains the following:

- ▶ Content store: Contains data required for Cognos BI to operate, such as reports specifications, published models and the packages that contain them, and connection information to data sources.
- ▶ Data sources: Known as query databases, these are the source of information accessed by the Cognos BI application; they can include relational databases and dimensional or OLAP cubes.
- ▶ Metric store: A relational database that contains the content for metric packages.

The IBM Cognos Platform provides, through the data tier, a single point of access to all data sources.

During the installation of IBM Cognos BI, you must specify where to install the gateways, Application Tier Components, and Content Manager. These IBM Cognos BI components can all be installed on one computer, or distributed across a network on multiple hardware servers. The server components within all three tiers can scale easily, either horizontally or vertically, to meet a variety of business intelligence processing requirements.

Figure 2-15 illustrates a detailed view of IBM Cognos BI installed on a single server.

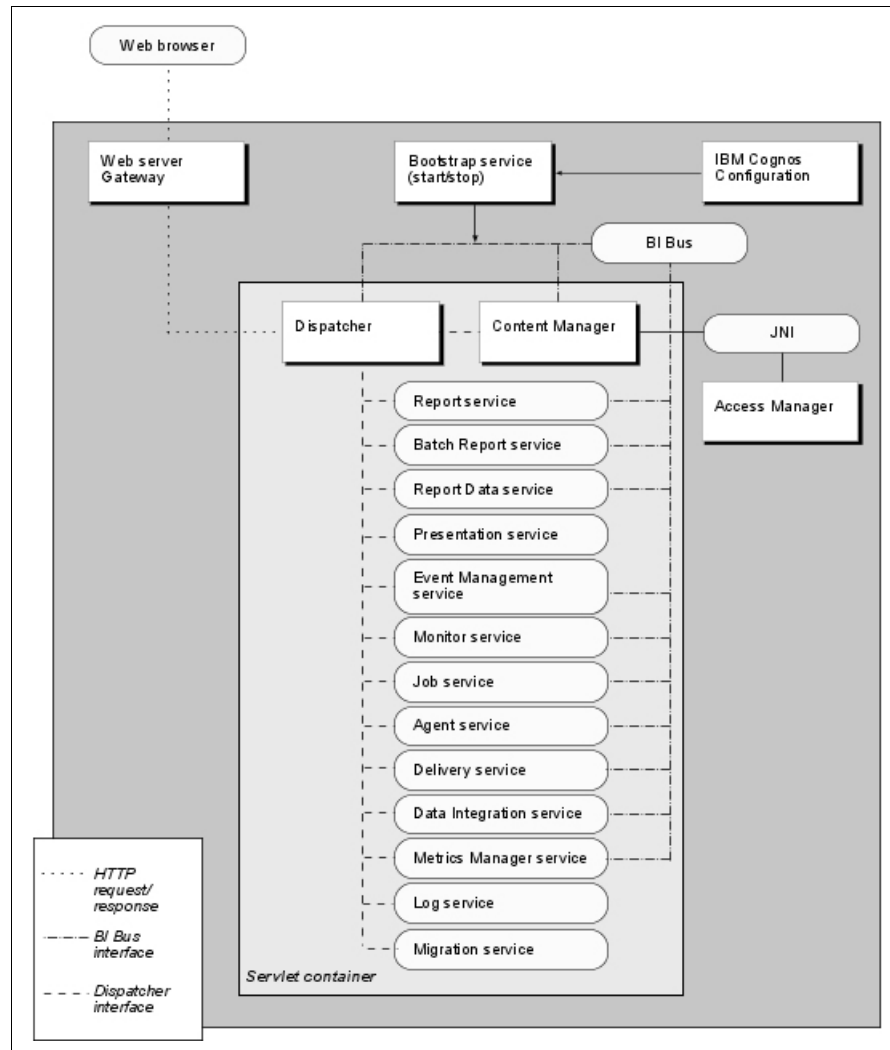


Figure 2-15 Detailed view of IBM Cognos Business Intelligence

IBM Cognos BI component installations use Tomcat as the default application server for running the IBM Cognos Data Manager Network Services SOAP server.

Refer to the *IBM Cognos BI Installation and Configuration Guide* for complete information about the installation and initial configuration process. For detailed information about IBM Cognos Platform architecture and server deployment options, refer to the *IBM Cognos BI Architecture and Deployment Guide*. All IBM Cognos documentation can be found on the IBM website at the following location:

http://www.ibm.com/support/entry/portal/documentation/software/cognos/cognos_business_intelligence_and_financial_performance_management

2.2.3 Architectural overview of the Optim family of tools

In this section, we provide insight into the common general principles that have been applied to this family of products. For more detailed information about the individual products, refer to the architecture and installation guides for the individual tool.

The InfoSphere Optim database family of tools is designed to provide a complete solution to the needs of those dealing with database environments, from data

architects and developers to users and DBAs. Figure 2-16 depicts the complete family of InfoSphere Optim tools.

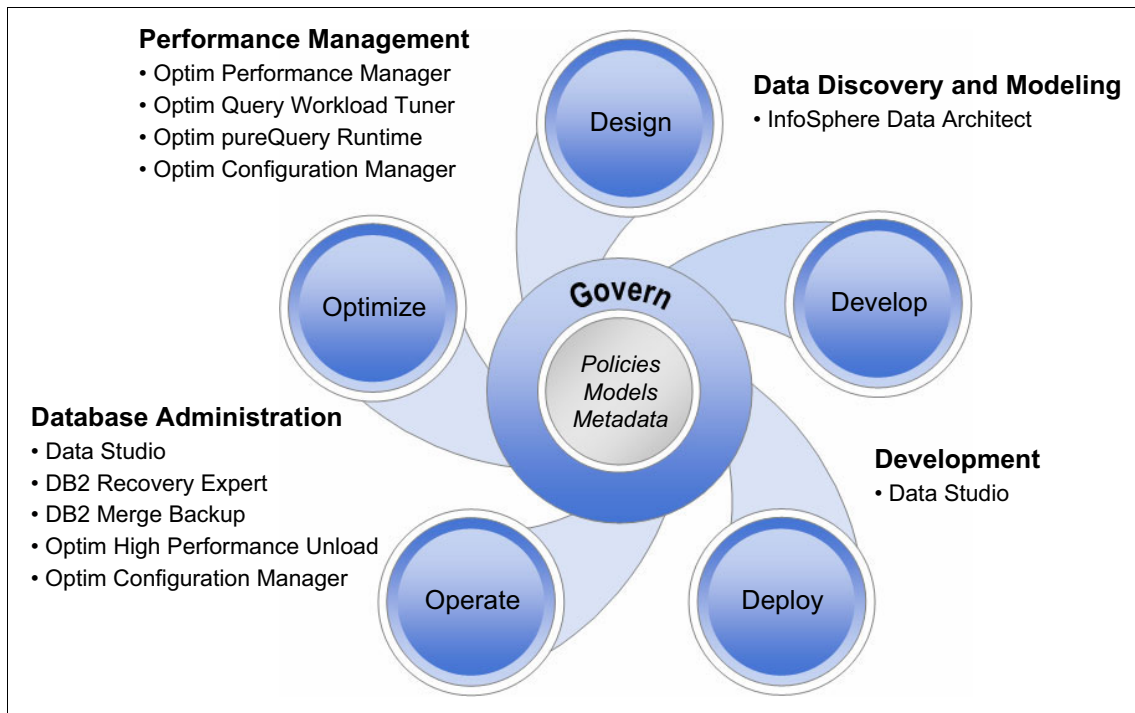


Figure 2-16 InfoSphere Optim database tools

Each InfoSphere Optim product has been designed to address a specific requirement, and as such the functionality of each product needs to be different. Where possible, however, they have been developed to share a number of common principles. An example of this concept is that the majority of the tools use a web interface for their operation, management, and configuration. These web interfaces are individually driven by an SOA service, implemented as a self-contained Web 2.0 application.

These applications have been developed using the Representational State Transfer (REST) architectural style that allows services to be exposed and consumed over the web using only a simple URL. The benefit of this is a small installation footprint, while providing powerful functionality.

An example of this type of Web 2.0 REST implementation can be seen in Figure 2-17 on page 48, which illustrates the simplified architecture of

InfoSphere Optim Configuration Manager. In this view, the Optim Configuration Manager server is the REST service, which drives the web console.

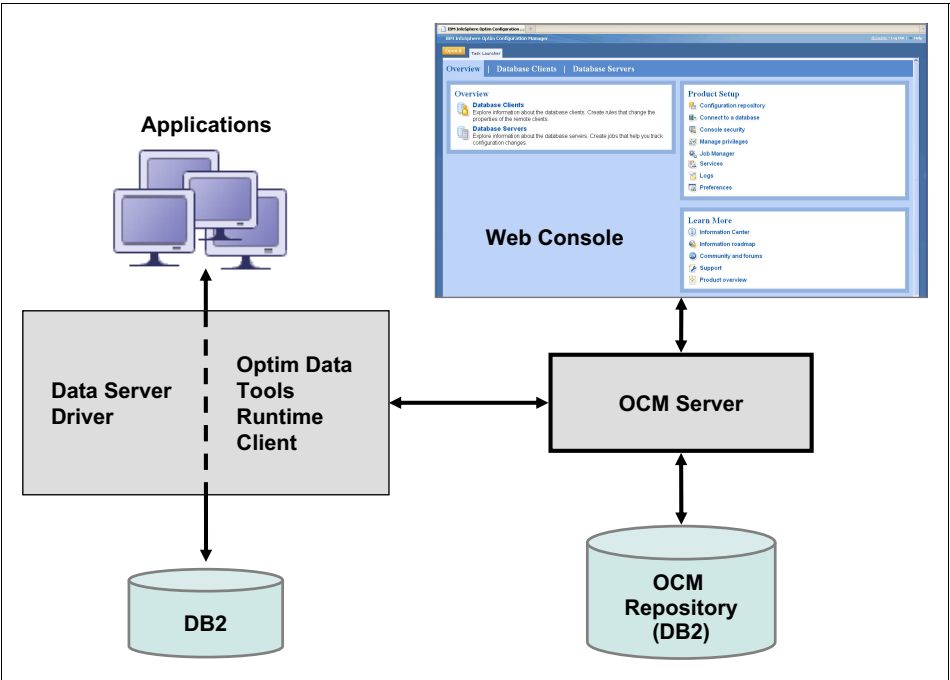


Figure 2-17 Simplified Optim Configuration Manager architecture

Figure 2-18 illustrates a simplified architecture InfoSphere Performance Manager. Again, the Web Console server is a REST Web 2.0 service.

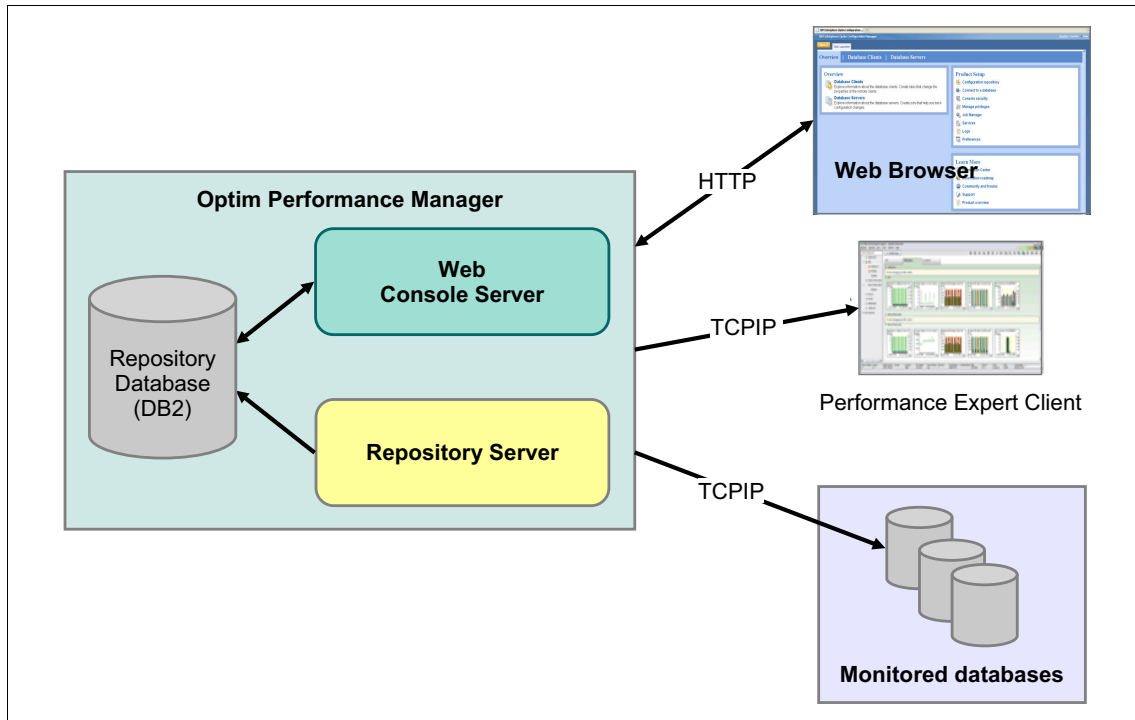


Figure 2-18 Simplified architecture view of InfoSphere Optim Performance Manager

The Optim tools have been designed to utilize the same metadata database for their internal requirements, which again reduces complexity and improves performance.

Another example of where these Optim tools work as a family can be found when working with the Optim client components, that is, elements that are installed on the servers being monitored. This is achieved during the installation process for each Optim product, with the installation program automatically determining compatible Optim components and placing these components into a common shared location. These InfoSphere Optim products use, and can share, the following common client components:

- ▶ IBM InfoSphere Optim Configuration Manager
- ▶ IBM InfoSphere Optim Performance Manager Extended Edition
- ▶ InfoSphere Optim pureQuery Runtime

2.2.4 InfoSphere Replication Server simplified overview

The key to achieving high data extract performance with InfoSphere Replication Server is the fact that all major relational databases use a log to record changes to the database. For example, if a record is inserted into a table, a log record storing the inserted data is written to the database log so it can be used for recovery purposes. Similarly, updates and deletes are also logged with sufficient data needed for recovery purposes.

InfoSphere Replication Server understands how to read the log to pick up the changes in the database. Therefore, instead of scanning an entire database to find the inserted, updated, and deleted records, InfoSphere Replication Server only has to read the log data that was changed.

Main components of InfoSphere Replication Server architecture are listed here:

- ▶ **Capture:** This component is responsible for reading the database log and capturing the changes. The component typically runs on the same machine where the source database resides, although for some databases, it can also run remotely from the source database.
- ▶ **Apply:** This component is responsible for implementing the changes gathered from the source database by the capture component and then writing these changes to the target database. The target tables can be either the same structure as the source table or different. Both data inserts and data replace commands are supported.
- ▶ **Admin:** This component provides an interface to configure and administer InfoSphere Replication Server. Both a user interface and a command-line interface are available.

InfoSphere Replication Server can be configured to implement each of three types of replication methods to move data between the source and the target data sources. These three replication methods make use of two completely different technologies. These three replication methods are:

- **SQL Replication** - This application process uses SQL commands to move the data between a set of staging tables on the source system to tables on one or many target systems.

These staging tables are used to store the captured changes within the logs by using the SQL Capture process, as shown in Figure 2-19.

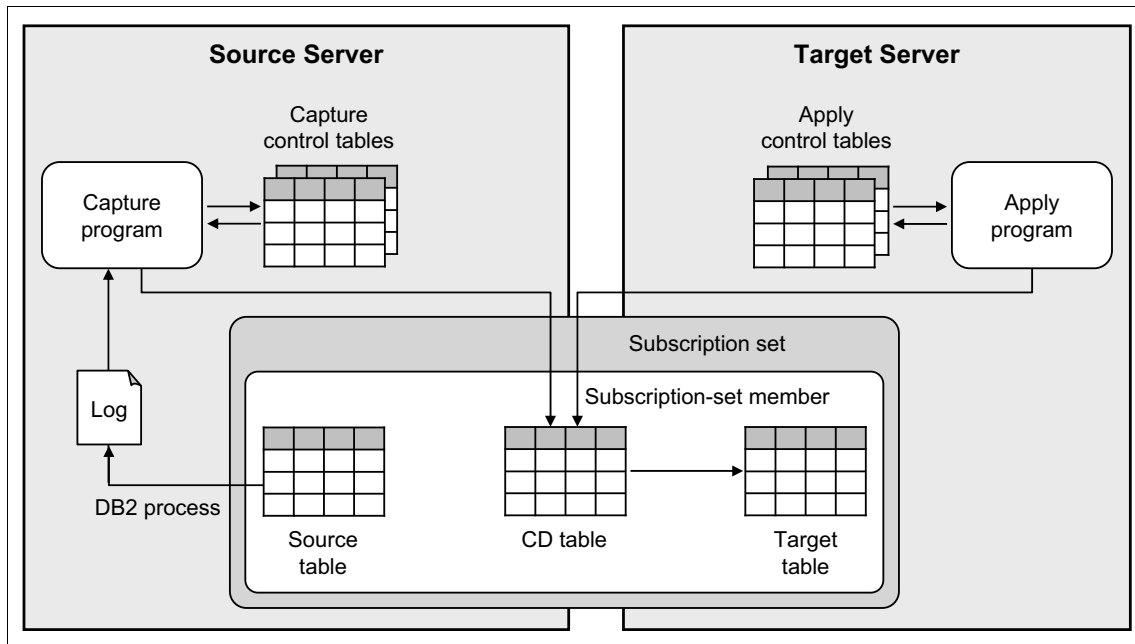


Figure 2-19 SQL replication

- **Q Replication** - This second replication process uses MQ as a transport medium. Within this process, the Q-Capture unit retrieves the changes from the logs. However, instead of storing them within staging tables, the Q Replication places them directly onto an MQ queue.

At the target system, these changes are then retrieved using a second MQ process, which then writes these changes into the target tables. This method

of replication gives near-real time performance. This replication method is shown in Figure 2-20.

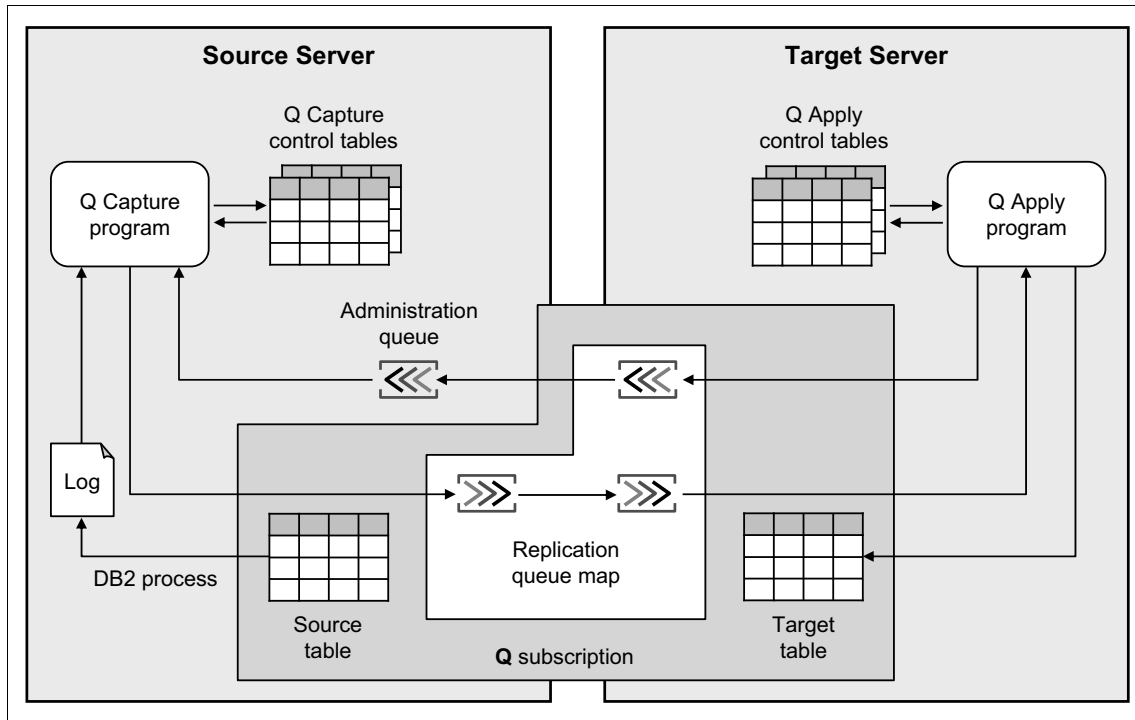


Figure 2-20 Q Replication process

- Event Publishing - This third replication method uses the same capture process as that of Q Replication with the data being published onto the MQ

queue. However, in this method the data is placed on the queue as either XML or comma-separated values (CVS).

At the target end, the MQ reading process is normally a user-developed application that uses this data. Figure 2-21 shows the Event Publishing replication method.

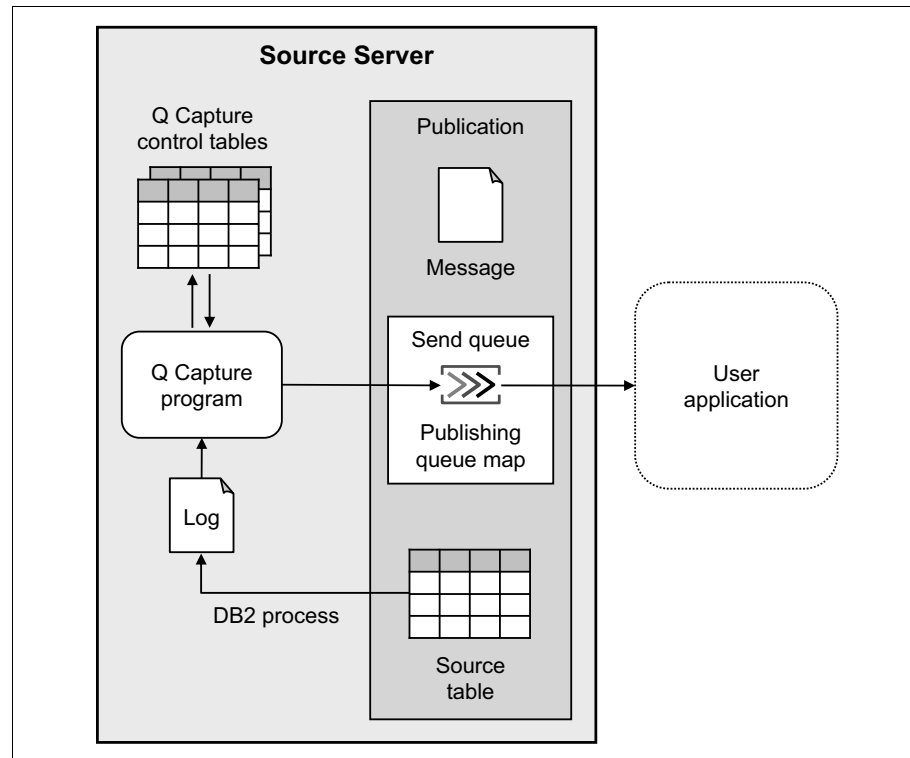


Figure 2-21 Event Publishing replication method

For more detailed information, refer to the *Replication and Event Publishing* section of the Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

2.2.5 InfoSphere Warehouse Advanced Edition client applications

InfoSphere Warehouse includes a rich set of development tools, and with the exception of the Cognos BI tools, they are all based on the universal Eclipse Workbench Platform. Eclipse is an open source development platform comprised of extensible frameworks, tools, and runtimes for building, deploying, and managing software across the software lifecycle. A number of companies,

including IBM, have adopted the Eclipse Platform as the base for a wide range of their development tools. Generally, for ease of installation, IBM ships each of these IDE Eclipse-based tools as a complete self-contained development environment, when in reality, they are extensions to the standard Eclipse Platform that have been developed by IBM.

The Cognos BI client tools, Query Studio, Analysis Studio, Report Studio, and Event Studio, are all web-based. The Cognos Frameworks is a separate Windows application.

Each of these tools, included in InfoSphere Warehouse Advanced Edition, is focused on a different role within the complete lifecycle of building a data warehouse environment. The roles and their respective clients are listed here:

- ▶ Data Architect - Aimed at the role of data architecting and the need to capture and model the business requirements, this tool can be used to develop both logical and dimensional logical models. These models themselves can become the basis for the physical models used in the creation of the warehouse data schema.
- ▶ Design Studio - Aimed at developers who must create the SQW applications that perform the ETL processes and cubing and mining services.
- ▶ Data Studio - Aimed at both end users and DBAs.
- ▶ Cognos BI - This Cognos tool group includes the Cognos Framework Manager and the Cognos Studio web tool set, and is aimed at BI users who must create reports and develop dashboards.

Data Architect

Data Architect is made available under the umbrella of the Advanced Edition licence, although it can be purchased from IBM as an independent product. This product was originally known as Rational Data Architect.

Design Studio

Design Studio is not an individual product but rather the main component of the InfoSphere Warehouse Client product, and it is listed under this name within the IBM software portfolio. It is made available as a standard element with all types of the InfoSphere Warehouse Server solution. It has been developed as an advanced, integrated development tool to handle the majority of the main tasks required to implement a data warehouse.

Following the principles of the Eclipse Workbench, Design Studio encompasses a number of different tools within one development environment, with each tool

aimed at performing an element of the total warehouse solution. Design Studio can thus perform the following tasks:

- Physical data model development

This tool within Design Studio is based on the original Rational Data Architect product, and allows the modeling of DB2 databases. Both the physical and logical database models can be created from scratch or captured from either a DDL or a real database.

Included within this modelling element is a method that enables you to graphically view the tables and their interrelationships. The model can be edited, meaning that new tables can be created and existing tables can be altered. This allows these alterations to be retrospectively applied back to the execution database within the IDE, or saved as a DDL file. The physical data model is also used within the other sections of Design Studio to provide metadata about the execution or remote databases.

- SQL warehousing tool for ETL development, testing and deployment

Commonly known as SQW, this tool allows for the development of data flows and control flows. These flows can be tested and debugged within the IDE, with connections to the execution database and remote data sources.

After development has completed, these flows are packaged into a warehouse application with the required generated Java and SQL code and the required resources for deploying the DIS. The SQW IDE also includes integration points with the IBM Data Stage product.

- OLAP Cubing models development

This component of Design Studio handles the implementation of cubing services. The tool can be used to add OLAP features to a physical model, which in turn then represents an OLAP cube model.

Additionally, this tool can be used to verify the cubing model and implement an execution instance of the cube service within the Design Studio Client. This model can then be transferred to the metadata database, and through an XML file be deployed within the InfoSphere administration console for execution as a cube service on the server.

- Data Mining models and flows development:

This function enables you to develop data mining models and flows without the need for in-depth data mining knowledge. The graphical interface of Design Studio allows for the easy conversion of business rules into mining flows, which can then either be shared with other mining tools such as SPSS and SAS, or deployed to the mining runtime server.

Data Studio

Before the release of DB2 10, both the client and server components of DB2 contained Java GUI applications to assist with both the management and access of DB2 databases and servers. However, these Java tools have now been deprecated and replaced with the more advanced tool, Data Studio.

Available from IBM at no cost as a web download, Data Studio tool has been developed to work with a greater range of relational database, including DB2 for IBM z/OS®, DB2 for i, Informix, and other non-IBM databases. The tool is available in three installation formats:

- ▶ The Data Studio full client provides an integrated development environment for database administration, and routine and Java application development, that can be installed with other IBM software products so that they can share a common environment.
- ▶ The Data Studio administration client is an alternative to the full client that provides a small footprint, stand-alone integrated development environment for database administration and non-Java routine development.
- ▶ The Data Studio web console provides health and availability monitoring features and job creation and management tools for your databases.

Data Studio includes the functionality originally shipped as Optim Development Studio, Optim Database Administrator, and Data Studio Health Monitor within one single offering.

Additional information can be obtained from the eBook: *Getting Started with IBM Data Studio for DB2*, which is available at the following website:

<http://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+IBM+Data+Studio+for+DB2>

Note: Data Studio is available at no cost as a download from the following website:

<http://www.ibm.com/software/data/optim/data-studio/>

IBM Cognos BI: Cognos Framework Manager and Cognos web-based tools

The included Cognos BI server comes with its own complete range of developments tools, all of which have to be accessed on a Microsoft Windows-based client system. All but one of the Cognos tools are web-based and come contained as integral part of the main Cognos BI server. They are accessible from the main Cognos menu as shown in Figure 2-22 on page 58.

These tools only function correctly when accessed from within an Internet Explorer browser, due to the need to use Microsoft Windows technology. The exception to the Cognos web-based tools is Cognos Frameworks, which is a separate Windows-only executable.

A complete list of the tools included as part of Cognos BI are:

- ▶ IBM Cognos Business Insight

This tool can create sophisticated interactive dashboards using IBM Cognos content and external data sources such as Cubing Services. Users can view dashboards and reports, open dashboards to manipulate the content, and email these dashboards.

- ▶ IBM Cognos Query Studio

With this tool, users with little or no training can quickly design, create, and save reports to meet reporting needs that are not covered by the standard, professional reports created in IBM Cognos Report Studio.

- ▶ IBM Cognos Analysis Studio

With this tool, users can explore and analyze data from different dimensions of their business. Users can also compare data to spot trends or anomalies in performance. IBM Cognos Analysis Studio provides access to dimensional, online analytical processing (OLAP), and dimensionally modeled relational data sources. Analyses created in IBM Cognos Analysis Studio can be opened in IBM Cognos Report Studio and used to build professional reports.

- ▶ IBM Cognos Report Studio

This is a robust report design and authoring tool. Using IBM Cognos Report Studio, report authors can create, edit, and distribute a wide range of professional reports. They can also define corporate standard report templates for use in IBM Cognos Query Studio, and edit and modify reports created in IBM Cognos Query Studio or IBM Cognos Analysis Studio.

- ▶ IBM Cognos Event Studio

You can use IBM Cognos Event Studio to set up agents to monitor data and perform tasks when business events or exceptional conditions occur with that data. When an event occurs, people can be alerted to take action, agents can publish details to a portal, deliver alerts by email, run and distribute reports, and continue to monitor the status of events.

- ▶ IBM Cognos Administration

This tool provides a central management interface that contains the administrative tasks for IBM Cognos BI. It provides easy access to the overall management of the IBM Cognos environment and is accessible through IBM Cognos Connection.

► IBM Cognos Framework Manager

This is the IBM Cognos BI modeling tool for creating and managing business-related metadata for use in IBM Cognos BI analysis and reporting. Metadata is published for use by reporting tools as a package, providing a single, integrated business view of any number of heterogeneous data sources. OLAP cubes are designed to contain sufficient metadata for business intelligence reporting and analysis. Because cube metadata can change as a cube is developed, IBM Cognos Framework Manager models the minimum amount of information required to connect to a cube. Cube dimensions, hierarchies, and levels are loaded at run time.

Figure 2-22 shows the browser interface to the Cognos BI tools provided.

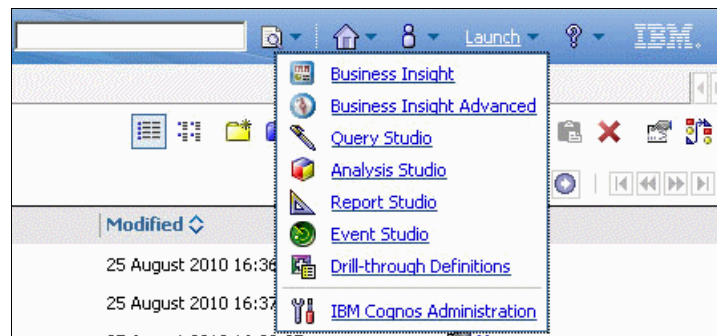


Figure 2-22 IBM Cognos BI tools implemented within a browser interface

2.2.6 InfoSphere Warehouse Advanced Edition command-line tools and utilities

InfoSphere Warehouse Advanced Edition comes supplied with a number of products and utilities, which primarily function from the native operating system command line of the system they are installed. For a Microsoft Windows operating system, this is the DOS command line. For versions of Linux, it is the bash command line. These command-line products and tools are listed here:

► DB2 Client

This is a set of bundled utilities and tools supplied as the client package for IBM DB2 product set. It provides the functions and tools required to both connect and work with DB2 relational databases installed on both local and remote hardware servers. Many DB2 Client utilities, such as LOAD, EXPORT, REGEN, and REORG, are made available to the ETL processes from within both the DIS runtime and Design Studio through the direct calls to the DB2 command line, because these functions cannot be executed using SQL or a JDBC database connection.

In this context, both the Design Studio and the DIS runtime write the required commands into a temporary file and then execute a call to the DB2 Client with the request to read and process the commands found within this file.

- ▶ InfoSphere Warehouse **wh** command

The InfoSphere Warehouse common client product set comes with an additional command line utility called **wh**. This utility has been developed to allow deployment, configuration, and running of SQW flows and resources within the InfoSphere Administration Console from the command line.

- ▶ DB2 Merge Backup

This is a command-line utility that can merge full or incremental DB2 backups to build a new full backup. This utility can be executed remotely from the actual DB2 server machine to free server resources. DB2 Merge Backup uses the DB2 database history to determine which backups to merge. Then DB2 Merge Backup accesses these backups and builds a new full backup that contains relevant information from the source backups.

- ▶ Optim High Performance Unload

This utility performs high-speed, bulk data unloads for DB2 databases. It is a stand-alone utility program that can be executed while the DB2 database manager is running, and can access the same physical files as the DB2 database manager. It can also operate in a stand-alone mode and as a DB2 stored procedure.

The benefits of Optim High Performance is that it will allow extraction even if the DB2 instance is down, and also can perform extractions on a different system.

2.3 InfoSphere Warehouse implementation scenarios

In this section, we discuss a common set of topologies that can be implemented to create a complete InfoSphere Warehouse environment. With the large number of products included within InfoSphere Warehouse Advanced Edition, there can ultimately be a large number of permutations of how all these products can be installed.

Here we present the most common scenarios. To simplify the discussion, we divide the solution into two sections. The first section focuses on components that make up the core of an InfoSphere Warehouse environment, and that are shipped with all versions of the product. The second section focuses on the tools and utilities that only come as part of the Advanced Edition.

In the last section, we touch on the “black box” solutions currently supplied by IBM as a complete system that includes both a hardware component and a software component.

2.3.1 Scenarios for implementing InfoSphere Warehouse core products

The components that make up a core InfoSphere Warehouse implementation can be divided into four Installation categories as follows:

- ▶ **Data Server component**

This category covers the main DB2 platform, which is supported on AIX, HP-UX, Solaris, various Linux implementations, and Windows. For a warehouse implementation, consider only the 64-bit versions of DB2.

- ▶ **Application server component**

This category covers the IBM WebSphere Application server, an integral part of the warehouse product set. This component also includes the InfoSphere Warehouse server application that is installed within an instance of the IBM WebSphere Application Server.

- ▶ **Clients**

This category covers all of the command line and GUI-based platforms that are normally installed on a user’s personal computer or notebook. Because the management, configuration, and use of many of the server applications and tools is through a web browser, count this as a client tool.

However, because all modern operating systems now include a web browser by default, no web browser is shipped with the product. Clients supported can either be 32-bit or 64-bit, and Windows-based or Linux-based.

- ▶ **Documentation**

This category includes both the online and PDF versions of the product set documentation.

Installation considerations: For InfoSphere Warehouse 9.7.3 and below, IBM WebSphere Application Server is installed as part of the installation process if an already installed version of IBM WebSphere Application Server was not selected to be used by the user.

With InfoSphere Warehouse 10, however, the installer requires an already installed copy of IBM WebSphere Application Server. Thus, before running the installer for InfoSphere Warehouse 10, a user must have installed a copy of IBM WebSphere Application Server by using the IBM Installation Manager application.

The InfoSphere Warehouse components covered in the categories listed can be installed onto a hardware platform in a range of topologies. The three most common topologies are listed here:

- ▶ One tier - Often utilized as a development, test, and education environment, all the major components, including the clients, are installed on a single hardware platform. In this case, the platform can only be a server installed with either a Windows or a Linux operating system.

A virtual machine image can be used in this type of environment, meaning a Linux VMware image such as that supplied as part of an IBM 5070 Smart Analytics Solution.

- ▶ Two tier - Although also primarily used in a development and test environment, this topology can be utilized, with the proper server and storage, for a smaller warehouse implementation.

In this topology, the client components are often installed on a single client Intel-based hardware platform running either Windows or Linux. The server components are then all installed on a second platform. In this topology, the clients and servers may be on completely different operating systems, and even on different hardware-based platforms. For example, the clients might be on Linux on an Intel-based platform, while the servers are on AIX on a Power-based platform. In such a case, the server components include the DB2 database engine and a single instance of the WebSphere Application Server, loaded with the InfoSphere Warehouse application.

- ▶ Three tier - In this topology, the client components are installed on a single client hardware platform, the DB2 components are installed on a second hardware platform, and the WebSphere Application Server components are installed on the third hardware platform. For ease of administration, it is advisable to have the operating system and hardware platform of the DB2

and WebSphere servers be identical, although the DB2 server will host a larger number of disks.

It might also be advisable for performance reasons to install a local copy of DB2 on the WebSphere Application Server to host the runtime metadata databases of the InfoSphere Warehouse application, leaving the main DB2 engine for simply the production database.

Figure 2-24 on page 63 illustrates the three most common topologies.

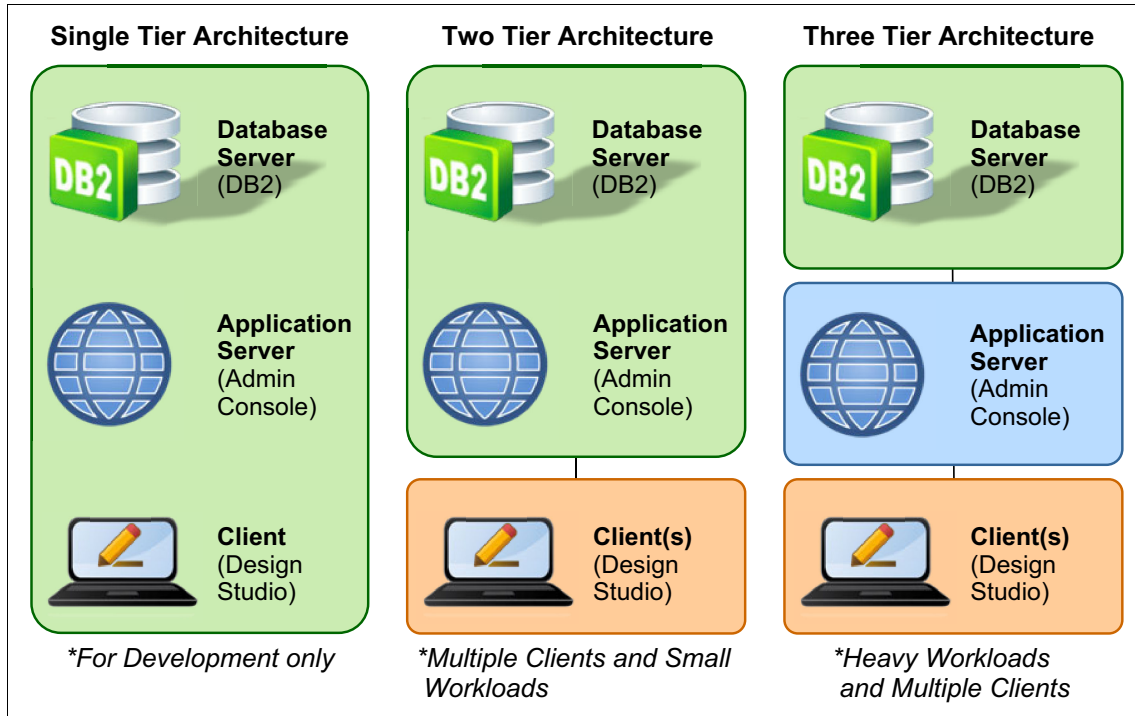


Figure 2-23 The three common topologies of InfoSphere Warehouse architecture

What is not shown in this figure is what can be called a “multi-tier” implementation. This is because, although there would be multiple hardware platforms, in principle it is still only three software elements to the implementation, and thus from a software view point, it would still be a three-tier implantation. The additional hardware platforms introduced in this extended three- or multi-tier implementation will all be installed and running additional DB2 relational database engines to create a partitioned database.

For completeness, a partitioned DB2 database can also be configured to execute on a single hardware platform that contains multiple central processor units. The DB2 database engine hides the complexity of a partitioned database from the

user and presents the user with what looks to be a single database. Because DB2 has no reasonable limit to the number of physical servers that a partitioned database can be installed on, this provides an InfoSphere Warehouse with both performance and scalability growth options.

Figure 2-24 demonstrates the three-tier implementation on multiple physical hardware platforms. Notice in this figure that the database server has an administration database node and multiple data nodes. It is the administration server that a user connects to and sends SQL and DB2 commands.

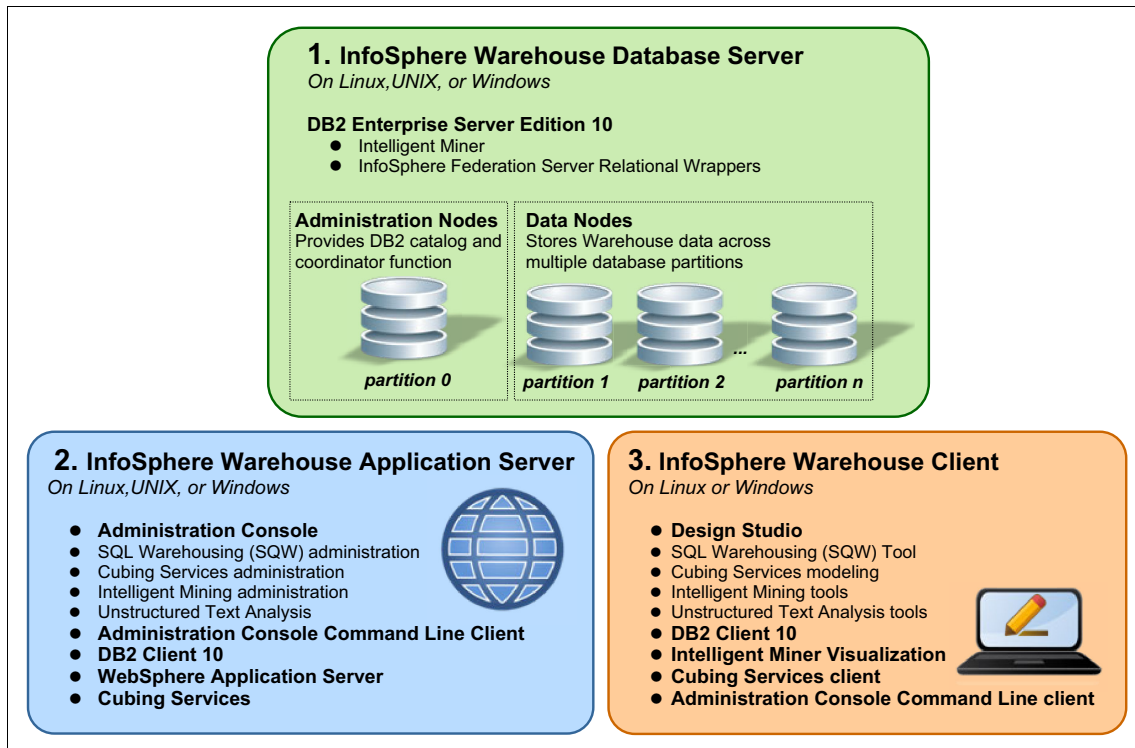


Figure 2-24 A three-tier solution on multiple tier physical servers

IBM Cognos BI is not installed by using the InfoSphere Warehouse install manager, but by using its own install process. Nevertheless, it is an important part of the data warehouse environment and is included with all versions of the IBM InfoSphere Warehouse product range. A standard installation of IBM Cognos BI includes a number of components that implement Cognos services. Included within these components are copies of the open source Apache Tomcat application server. It is possible, through a manual process, to replace the Apache Tomcat application server and host these Cognos elements within a WebSphere Application Server.

Regarding the topologies listed, the Cognos BI element is, by default, installed within the same tier as that of the InfoSphere Warehouse J2EE application and the WebSphere Application Server. If additional performance is required for supporting a larger number of Cognos BI users, the Cognos services, similar to those of DB2, might be distributed over a number of physical servers.

2.3.2 Scenarios for implementing the Optim family of tools

In this section we discuss how we can introduce the family of Optim tools, included as part of the Advanced Edition, into this data warehouse environment.

In a one-tier topology, that is, the type of implementation utilized for development and limited testing, it might be necessary to install one or more of the Optim tools. InfoSphere Optim Query Workload Tuner, InfoSphere Optim High Performance Unload, and InfoSphere Data Architect are command-line utilities that will not have a significant impact on the system.

Other tools, such as InfoSphere Optim Performance Manager and InfoSphere Optim Configuration Manager, require an Optim metadata database that has to reside within a DB2 instance that is different from the default warehouse instance. This is because the Optim instance must be implemented with a different set of DB2 configuration settings.

Although it is feasible to install and execute the InfoSphere Warehouse Server and Client applications along with the Cognos BI server and the InfoSphere Optim family on one platform, in reality it is impractical. If all of these elements need to be hosted on one physical server, a more suitable solution is to create a virtual two-tier solution.

In this scenario, one virtual platform hosts all the components found within a single tier implementation of InfoSphere Warehouse. The second virtual platform, with its own operating system, hosts the required InfoSphere Optim tools, along with a copy of DB2 into which the Optim metadata database is to be loaded. Both of these virtual platforms can then be hosted on the same physical server. This scenario was used in the creation of our test environment as outlined in more detail in Chapter 3, “Scenario: Warehouse Live Demo Set” on page 69.

To implement a two-tier environment, whether using two virtual platforms or with two physical servers, the practical solution is to host the InfoSphere Warehouse components on one platform and the Optim tools on a second platform.

If, because of the size or number of data warehouse systems being monitored, the InfoSphere Optim tools require additional resources, they can be then

sub-divided further into individual platforms. Even in this format they can still share resources, meaning the Optim metadata database.

2.3.3 Alternative IBM solutions for data warehousing

In this section we briefly discuss the alternative product offerings available from IBM Information Management for providing a complete data warehousing solution, to give you an understanding of their relationship to the InfoSphere Warehouse product range. The three main solutions are listed here:

► **IBM Smart Analytic System**

This is an integrated solution that encompasses both hardware and software into an integrated platform. It provides rapid deployment and a faster implementation solution, while also lowering the cost.

The IBM Smart Analytics System is a set of offerings that covers a wide range of base hardware platforms, thereby allowing flexibility of both cost and performance and giving the choice of the best solution to meet a customer's challenges. IBM InfoSphere Warehouse is one of the main software components within the IBM Smart Analytics System. Figure 2-25 illustrates one of the IBM Smart Analytic System offerings.

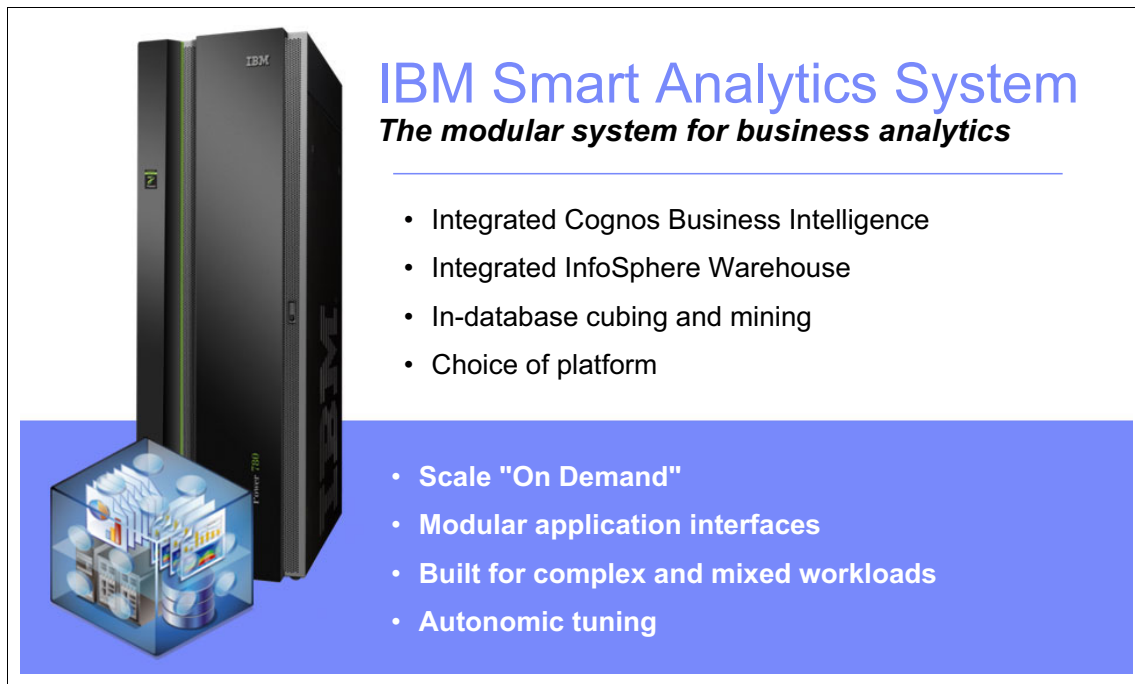


Figure 2-25 IBM Smart Analytics System

► IBM Netezza Analytics

This is also an embedded, purpose-built, advanced analytics platform consisting of special scalable, high-performance hardware combined with advanced built-in software that provides an integrated database and a massively parallel analytic platform designed to work with petascale data volumes.

This technology solution is aimed at enterprise customers who have BI questions, where both performance and the volumes of data result in a normal solution being impracticable. Figure 2-26 illustrates the IBM Netezza Analytics solution.



Figure 2-26 IBM Netezza Analytics solution

► IBM InfoSphere DataStage

This is a software platform that can be installed on a range of high-performance hardware platforms. It provides these unique capabilities:

- A powerful ETL solution that supports the collection, integration and transformation of large volumes of data, with data structures ranging from simple to highly complex. It can manage data arriving in real time and data received on a periodic or scheduled basis.

- A scalable platform that enables companies to solve large-scale business problems through high-performance processing of massive data volumes. By leveraging the parallel processing capabilities of multiprocessor hardware platforms, IBM InfoSphere DataStage Enterprise Edition can scale to satisfy the demands of ever-growing data volumes, stringent real-time requirements, and ever-shrinking batch windows.
- Comprehensive source and target data sources support for a virtually unlimited number of heterogeneous data sources and targets in a single job includes text files; complex data structures in XML; ERP systems such as SAP and PeopleSoft; almost any database (including partitioned databases); web services; and business intelligence tools such as SAS.
- Real-time data integration support operates in real-time. It captures messages from Message Oriented Middleware (MOM) queues using JMS or WebSphere MQ adapters to seamlessly combine data into conforming operational and historical analysis perspectives. IBM InfoSphere Information Services Director provides a service-oriented architecture (SOA) for publishing data integration logic as shared services that can be reused across the enterprise. These services are capable of simultaneously supporting high-speed, high reliability requirements of transactional processing and the high volume bulk data requirements of batch processing.

There are points of integration between InfoSphere Warehouse and InfoSphere DataStage. An example is the ability for Design Studio to both import datastage flows and call for the execution flows on an InfoSphere DataStage system.



Scenario: Warehouse Live Demo Set

In this chapter we introduce the business scenario that creates the need for our business intelligence (BI) solution, and helps to showcase our example environment. Later chapters of the book utilize this scenario and, more importantly, the example environment, as the base for demonstrating the key elements of the relevant chapter.

By utilizing the same environment for all the examples and applications, we further highlight the interlinked relationship of the complete solution found within InfoSphere Warehouse Advanced Enterprise Edition.

3.1 The problem: warehouse growth for “Mom and Pop” retail

InfoSphere Warehouse can be implemented to help deliver the key information required to make informed decisions based on the data available. In this scenario, as with most businesses, we start with our problem. Then, however, in this chapter we move forward almost to the end of our story. We do this by demonstrating how the analytics tools can present our transformed data in a manner that best informs us of the information available, and allows us to interrogate this information, to help drive business solutions.

Our example describes a medium-sized retail chain that has a number of outlets in both the US and Canada. The CIO has been asked by the CEO to provide more accurate information about their products and more importantly, the buying habits of their clients.

Until now this information had been gathered from a number of sources and then correlated and presented to senior management as a monthly set of reports. From these reports the CEO built a picture of the business and planned the actions of the company including which products to stock, when to place these items on sale, and determine which stores were in a profit or loss status.

But it was taking too long to obtain and read these reports, and then make decisions and feed the necessary actions back to the stores. The company's main competitors had been able to respond to the market faster and outsell this retail chain. The CEO concluded that the report process was too slow, cumbersome, and inaccurate to enable the company to be competitive in a 21st century business arena. Seeking a better way of doing business, the CEO challenged the CIO to improve the situation, and quickly, to enable the company to survive.

The CIO consulted with the in-house IT team to determine the best way to resolve this problem. IBM was the retail chain's IT supplier, so the IT team turned to the IBM account team for help in handling this challenge.

The IBM account team met with both the CIO and IT team and explained the benefits of applying an InfoSphere Warehouse solution to address the issue. Although the CIO expressed concern about how long it might take to implement this solution, this concern was soon alleviated when the account team proposed using the solution packs that were developed for data warehousing.

When implementing the IBM InfoSphere Warehouse Pack For Customer Insight, a business receives a set of prebuilt data models, consolidated database schema structures, and report templates that support a consolidated view across

products and clients. That, in turn, provides a true analysis of the business, including items such as the most and least profitable clients. The packs also enable a high degree of reuse, automation, and data governance.

Working in conjunction with IBM Software Services, the in-house IT team successfully implemented their data warehouse solution by using the InfoSphere Warehouse Advanced Enterprise Edition and the packs for Customer Insight, Market and Campaign Insight, and Supply Chain Insight.

At the next senior management meeting the CIO was able to dispense with monthly paper reports by using Cognos Mobile reporting, a component of InfoSphere Warehouse, to demonstrate how the company's three loyalty programs were received by their clients, and to compare membership rates across all three programs. The CIO additionally was able to show the costs associated with running the loyalty programs, and the gross and net margins and profits associated to client transactions within the loyalty programs.

3.2 Warehouse Live! Demo

In this section, using the same data set as the CIO presented to the CEO, we explain how we can view the same reports that were presented through either a web browser or a mobile device running "Cognos Mobile viewer" on iPad or Android.

The Cognos reports presented by the CIO in our scenario have been made available by the IBM Information Management team so that clients can gain insight to the capability, power, and simplicity that the IBM Cognos BI application can bring to the navigation and presentation of large volumes of data. The demo is known as the "IBM Cognos Warehouse Live! Demo" and it is available through a number of options:

- ▶ As a VMware image, available through your IBM representative.
- ▶ By installing the IBM Marketing VMware image that, at the time of writing, can be found on the IBM internal website:

http://passitalong.tap.ibm.com/passitalong/topic/show/familiarize_yourself_with_the_warehouse_demo?from_lp=2765

- ▶ By directly connecting to a hosted copy of the VMware image on the IBM internal website. At the time of writing, the Cognos Warehouse Live! Demo, is available at:

<http://vmpoc.torolab.ibm.com/ibmcognos>

To access the demo with Cognos on Mobile, use this web address:

<http://vmpoc.torolab.ibm.com/ibmcognos/m/index.htm>

- By building your own Cognos BI demo using the IBM demo versions of IBM InfoSphere Warehouse, IBM Cognos BI server, and the elements contained within the Customer Insight Pack.

The demo supports both the HTML using a browser and iPad Cognos mobile application access.

To provide you with a basic understanding of what is possible with both the demo and the power of the Cognos BI application, we demonstrate how to use the Cognos BI demo in this section. This example shows how you can use the advanced features of Cognos BI to easily navigate the data contained within the data warehouse.

1. After you obtain access to the Cognos demo, either through a copy of the VMware image or through the direct internal web link, connect to the Cognos BI server using a suitable web address:

`http://HOSTNAME/ibmcognos`

In this example, HOSTNAME will be replaced with the host address of the Cognos BI server.

2. Select **Public Folders** → **Customer Insight** from the menu option in the center of the window (Figure 3-1).

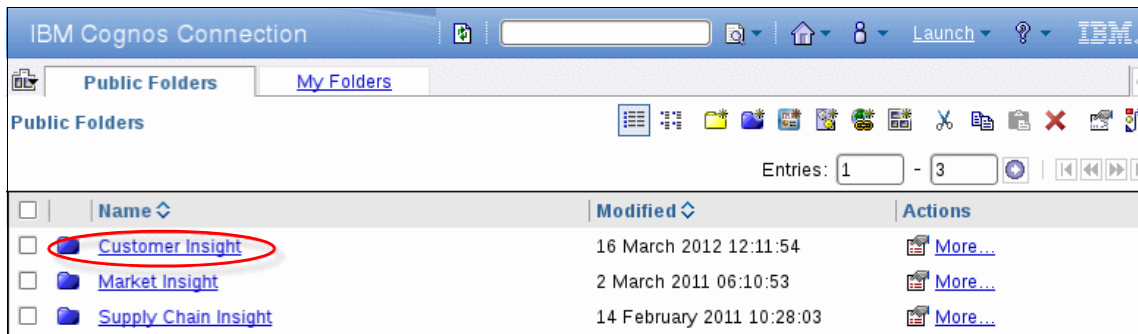


Figure 3-1 Select Customer Insight to start demo

3. Select **Sample Reports** from the menu options presented on the second window. This lists a complete set of possible reporting options (Figure 3-2).

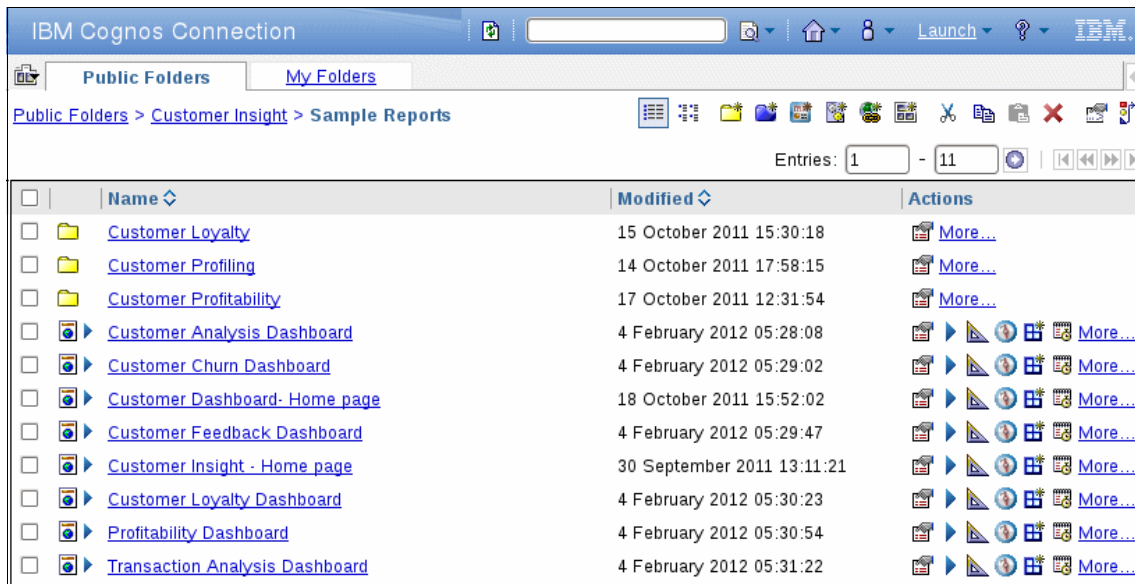


Figure 3-2 Possible reporting options within the Cognos Live Demo

4. From the final menu window, select **Customer Loyalty Dashboard**.
Figure 3-3 shows the Customer Loyalty Dashboard from Cognos 10.

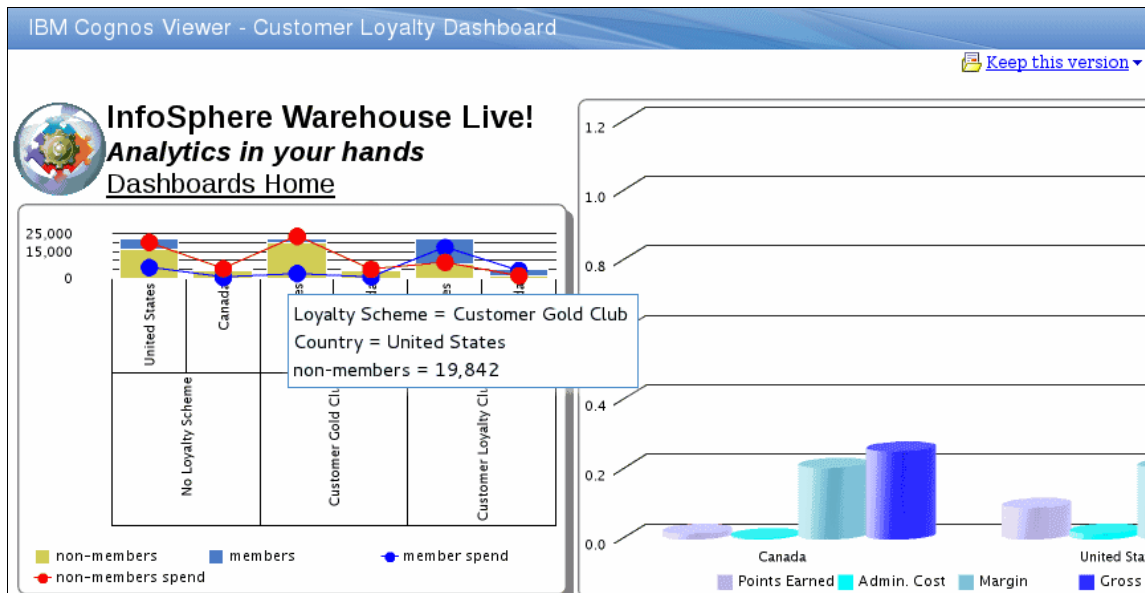


Figure 3-3 The Customer Loyalty Dashboard presented by Cognos10

The dashboard displays a basic view of the three loyalty programs implemented by the company. It compares the membership rates across all three programs (specifically, the US and Canada). It also shows the costs associated with the loyalty programs, the gross and net margins, and profits associated to client transactions within the loyalty programs.

This type of reporting allows the vast amount of data held within the data warehouse to be viewed in a method that is easier and faster to digest. By selecting elements of this window, we can drill down to granular detail through an additional reporting window.

5. As an example, note the "Gold Club" members' contribution for the United States (top left chart in the middle). Clicking an area part of a chart (or, on some browsers, hovering over a column) produces a pop-up window detailing the data set value (Figure 3-4 on page 75). Now we can see that the

composition of non-members of the Gold program is disproportionate to members.

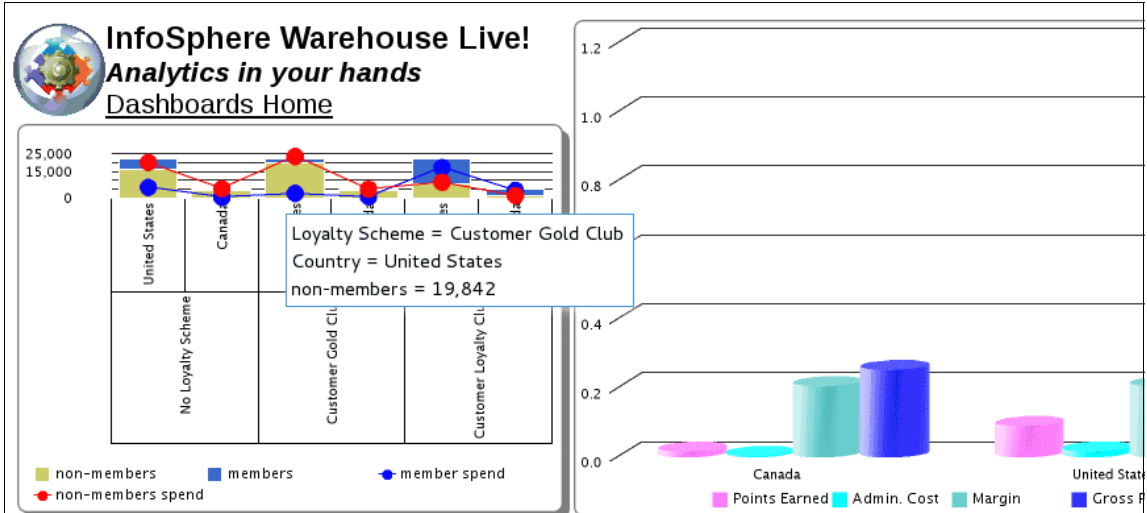


Figure 3-4 Viewing the underlying data of any chart

6. By right-clicking a data column (or, on some mobile platforms, selecting the question mark (?) symbol), a new menu with more options is displayed (Figure 3-5).

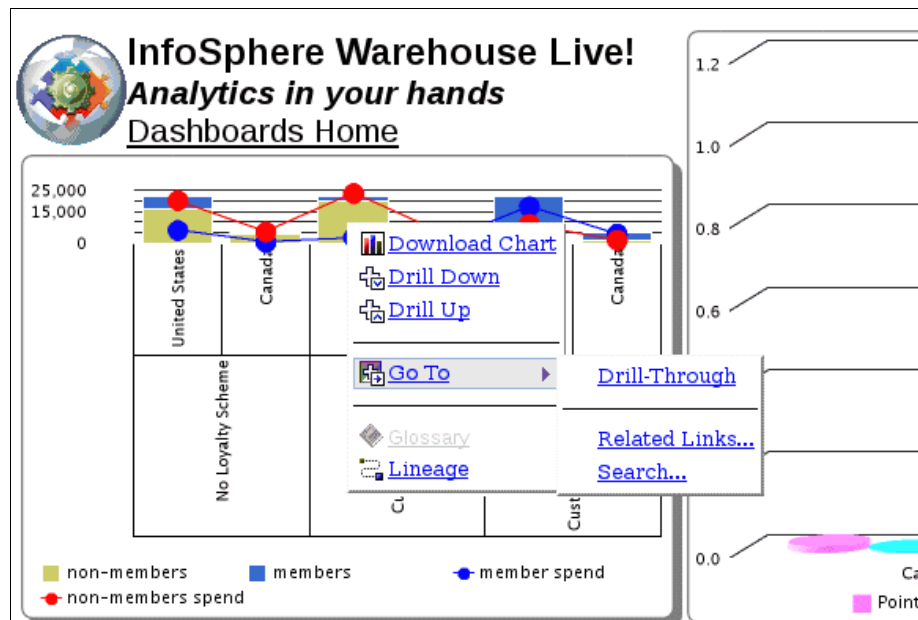


Figure 3-5 Additional details menu resulting from selecting the question (?) mark

7. From this menu, we can choose the “Drill Down” option on the data to provide a more detailed chart of the element selected. In our example, this results in a breakdown by state for the Gold Loyalty Program. Thus, we may select from

any number of the elements on the window to drill down to a more detailed view of that element (that is, those not in any of the Loyalty programs).

We also can select the “Drill Through” option, which goes directly into a more detailed report supporting the data contained in this chart on the dashboard (Figure 3-6).

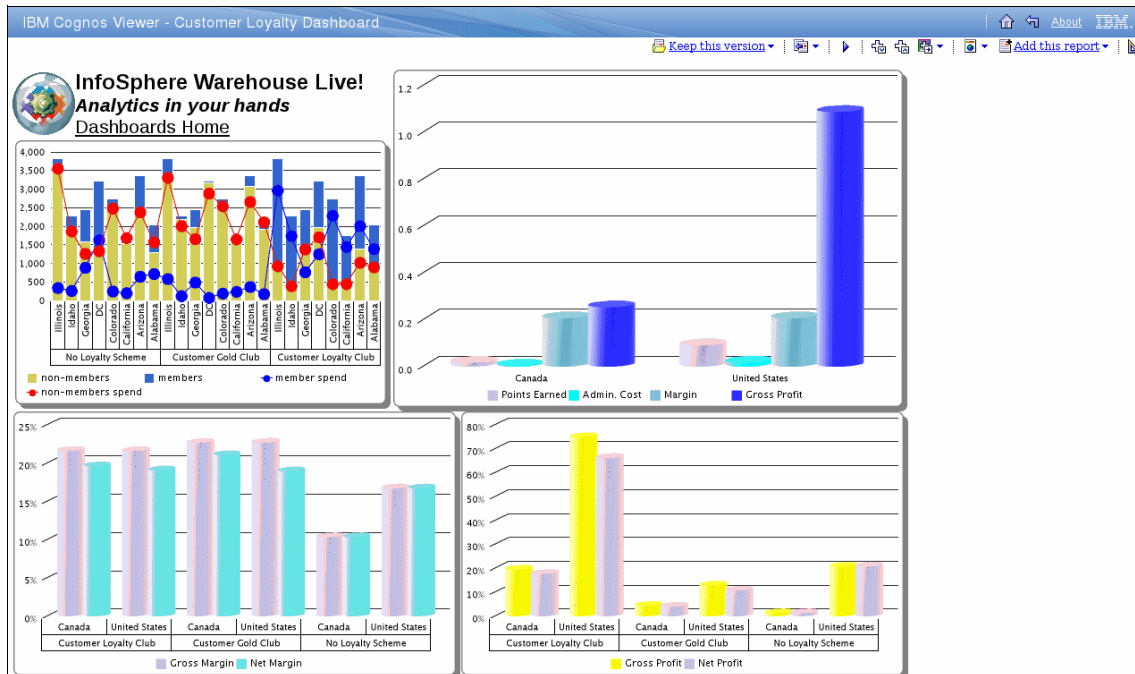


Figure 3-6 A detailed report presented after choosing to drill through from an overview report

8. Clicking the Back icon (🔙) of the report returns the viewer to the previous window.

So, as with the CEO of the retail company, we are able to view the details of the three loyalty programs implemented by the company including membership rates, the gross and net margins, and profits associated to client transactions.

3.3 Example architecture used in this book

In this section, we describe the architecture of the environment used to demonstrate our examples throughout this book. Our environment used VMware

images, as is common for many demos and examples. In our lab environment, we had three VMware images in an ESX server:

► System One

This “main” system VMware image is based on the standard IBM Smart Analytics System 5710 that IBM offers as a method of purchasing a solution for data warehousing that is ready for immediate use. This VMware image makes a perfect starting base for those in need of a rapid implementation for development and testing, or even as a small company solution.

The integrated solution for our lab included the following software:

- Novell SUSE Linux Enterprise Server 11 (SLES) SP2 64-bit operating system
- InfoSphere Warehouse Departmental Edition, 10.1
- DB2 for Linux, UNIX, and Windows, Version 10.1
- Cognos10 BI 10.1.1.0 and Cognos Mobile 10.1
- WebSphere Application Server 8.0.0.2

Our DB2 database had two database partitions in two physical nodes. System One was the coordinate node of the partitioned DB2 database. WebSphere Application Server hosted our InfoSphere Warehouse Instance.

This system was also the IBM Cognos 10 BI server. During the initial configuration, we selected the option to install the “Cognos Live Demo” example set. The Cognos Live Demo provides a preconfigured and populated execution database containing a defined table and schema structure along with a retail product and client data set.

This example set also contained a populated Cognos metadata database in support of the already created Cognos reports. We used these reports in the demonstration provided in 3.2, “Warehouse Live! Demo” on page 71 in support of our scenario.

This system was configured with 5 GB of RAM and 1 processor.

Note: When installing and configuring the Cognos Live Demo VMware image, select the option **Cogos Live Demo** to include the demo database and reports.

► System Two

The second VMware image was also based on IBM Smart Analytics System 5710 with the same software installed as in System One. This was the second DB2 database physical node. Both the InfoSphere warehouse and Cognos BI servers were deactivated.

This system was configured with 5 GB of RAM and 1 processor.

► System Three

This was a Windows-based VMware image that contained the IBM Optim products including IBM Optim Performance Manager and Optim Configuration Manager.

This system was configured with 4 GB of RAM and 1 processor.

IBM Optim tools also support Linux, AIX, and z/OS operating systems.

Figure 3-7 depicts our lab environment.

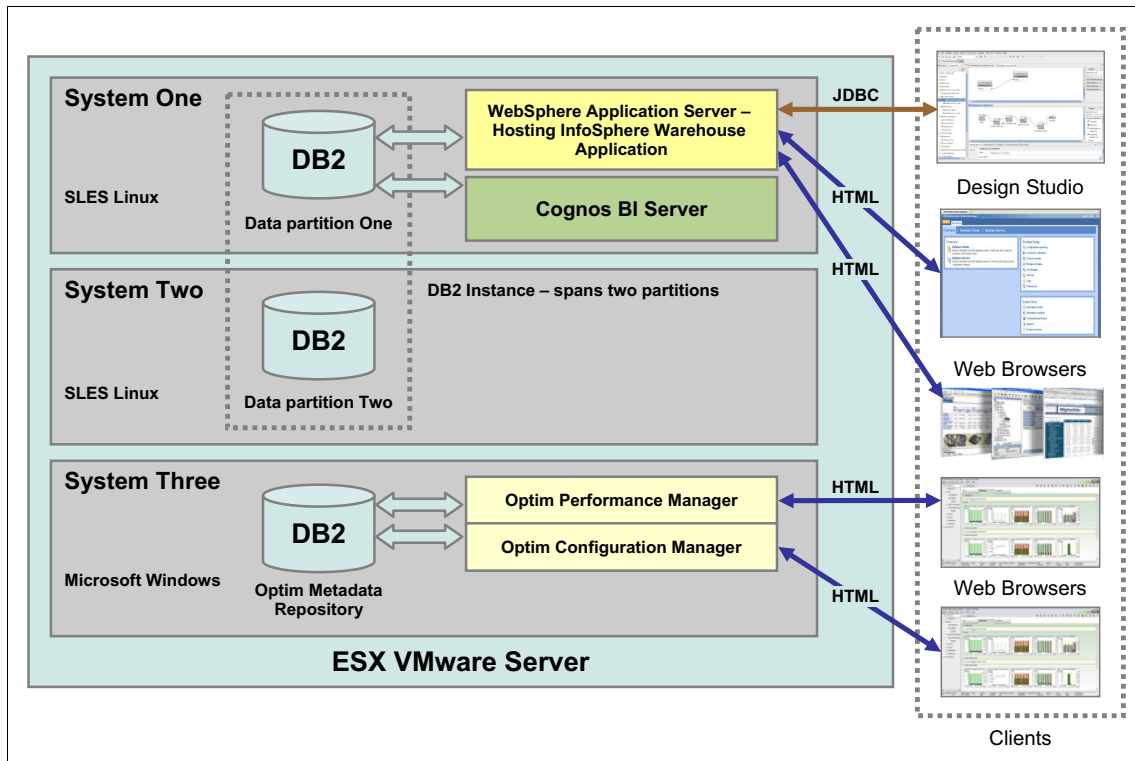


Figure 3-7 Lab environment

Because the majority of the IBM products offer Try and Buy licences, all the examples contained within this book can be attempted by our readers.



Data modeling: End to end

Modeling touches all elements of a data warehouse implementation, from the logical model representing the business logic, the dimensional, and the OLAP models imparting order, to the physical data model simulating a real database.

In this chapter we cover this modelling evolution, starting with our business questions, and ending with the implementation of an InfoSphere Warehouse solution. This process touches upon the tools required to develop and manage these models and the interfaces between them.

It is not the intention within this chapter to dig deep into the details of the tools utilized within this end-to-end modelling process, but rather to give an overall example of their role in a model's journey, and the interfaces between these tools. There are many excellent IBM Redbooks that cover the individual tools in depth; these are referenced within the chapter.

The Redbooks publication *Dimensional Modeling: In a Business Intelligence Environment*, SG24-7138, is a suggested companion to this chapter.

4.1 Start with the business problem

In a true business intelligence environment, the objective of a data warehouse is not to provide the means of holding large amounts of data, or even to provide an efficient platform for processing queries. These are, in reality, the requirements needed by a data warehouse to achieve its main goal of providing the means to answer questions such as the following about your business:

- ▶ Did our business grow from last quarter or last year?
- ▶ Will we have sufficient stock in key regions?
- ▶ Which stores are making a profit or have a loss?

It is these kinds of questions that are important to the business. When you start to plan your data warehouse solution, they become the first step in your modeling process.

4.1.1 Online analytical processing

Normally, any company will contain at least one, if not many, operational production databases that are used in the running of the core business. Traditionally, these types of relational databases are implemented as part of a transaction processing application. For this reason, they are known as online transaction processing (OLTP) databases.

Such databases are aimed at high volume, high velocity, and low complexity transaction workloads, including those generated during a sales transaction either at a shop till or an online shopping site. The details of what items customers purchased, whether they used cash or credit cards, along with what time the transactions took place are all stored within the OLTP database.

In these cases, the database design will be focused on a single or low number of table queries and updates. A database design best suited to this type of environment is based on an entity relationship schema, and is typically modeled to a level of normalization called third normal form (3NF). This type of model can be implemented within IBM database development tool called InfoSphere Data Architect, as one of the following data models:

- ▶ A logical data model, which is a model that represents the entity or table structures needed to perform a business operation
- ▶ A physical data model, which is a simulation of the tables and other database objects that constitute a real database

Figure 4-1 illustrates both a logical model and physical model within InfoSphere Data Architect.

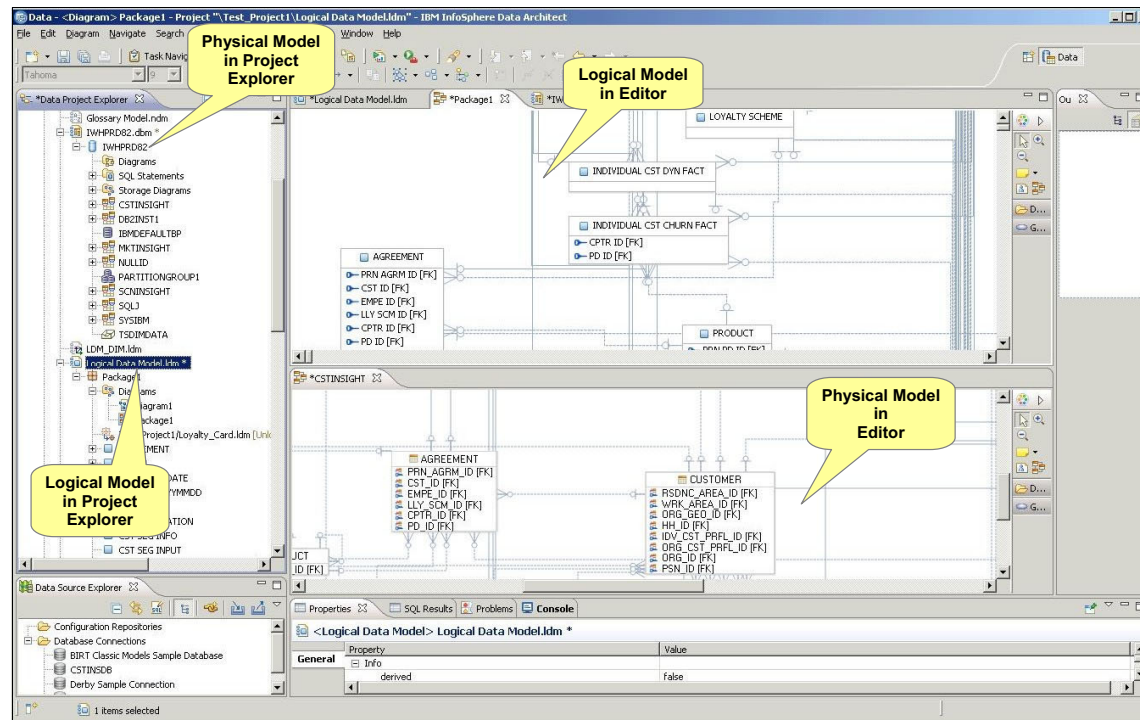


Figure 4-1 Logical and physical models in InfoSphere Data Architect

Levels of normalization: For those not familiar with normal form (NF), the full list is:

- 1NF - No repeating elements or groups of elements within a table.
- 2NF - No partial dependencies on a concatenated key within a table.
- 3NF - No dependencies on non-key attributes within a table

By its nature, a database implemented for online transaction processing is inefficient when used in performing the types of analytical queries required in answering the range of questions demanded by the company's business analysts. The reason is because these analysts, in performing their work of producing business intelligence reports, must create highly complex, multidimensional queries that often must engage with large quantities of historical data.

However, a database designed to handle OLTP transactions is developed to process short, simple queries. Thus, it will be inefficient when handling these complex queries, often resulting in many calls to the database to return the required information. Normally termed online analytical processing (OLAP), these types of complex queries therefore require a completely different schema structure to that implemented for OLTP.

A solution to this problem can be found in a format called a *star schema*. A star schema contains a structure where the data is divided into both Fact and Dimensional tables as shown in Figure 4-2.

This star schema allows complex queries to be processed with a far smaller number of interactions with the database. For an efficient OLAP solution, develop each star schema to represent a single business process that is at the center of your related business questions.

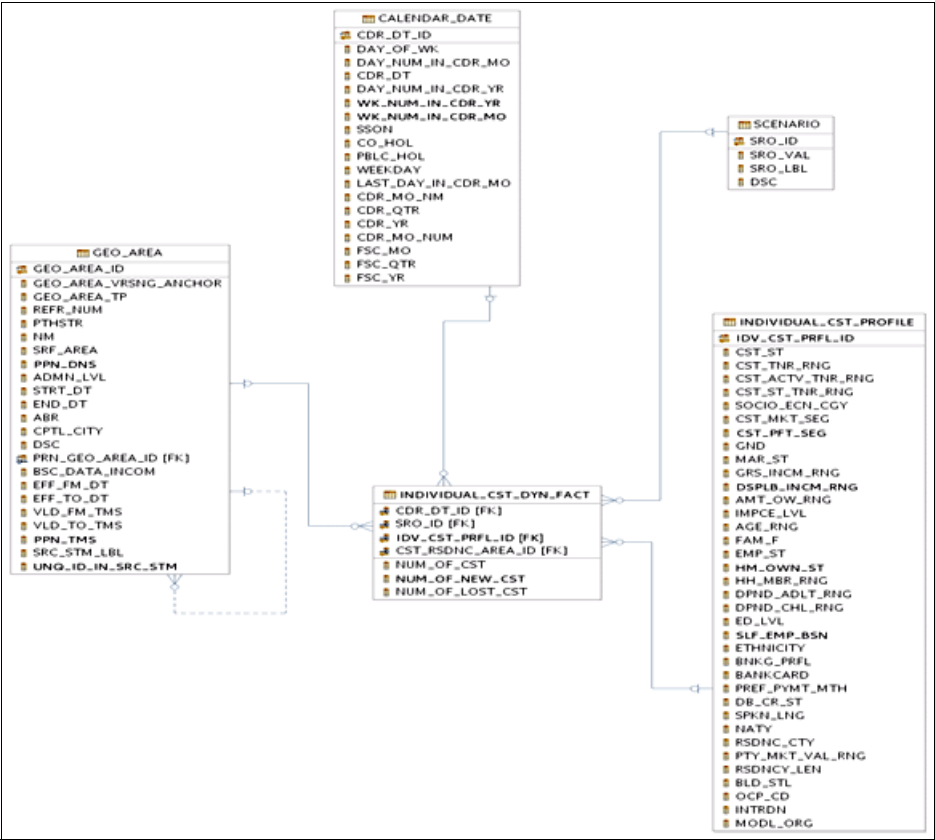


Figure 4-2 Tables organized as a star schema

As shown in Figure 4-2 on page 84, the simplest and most efficient implementation of a star schema consists of simply one fact table related to multiple dimension tables, with the fact table containing the foreign keys to the respective primary keys within the dimensional tables.

However, there can be two additional types of star schema, as shown in Figure 4-3. The first of these is the *snowflake*, in which the dimensional tables related to the fact table are themselves connected to additional dimensional tables. This type of star schema is more complex and, if possible, not recommended.

The final star schema is that of the *multi-star* model, which consists of multiple fact tables joined together through the dimensional tables. It is this multi-star model that most common for mapping multiple business process, for example, sales and inventory which are often interrelated and share common elements such as products and stock levels.

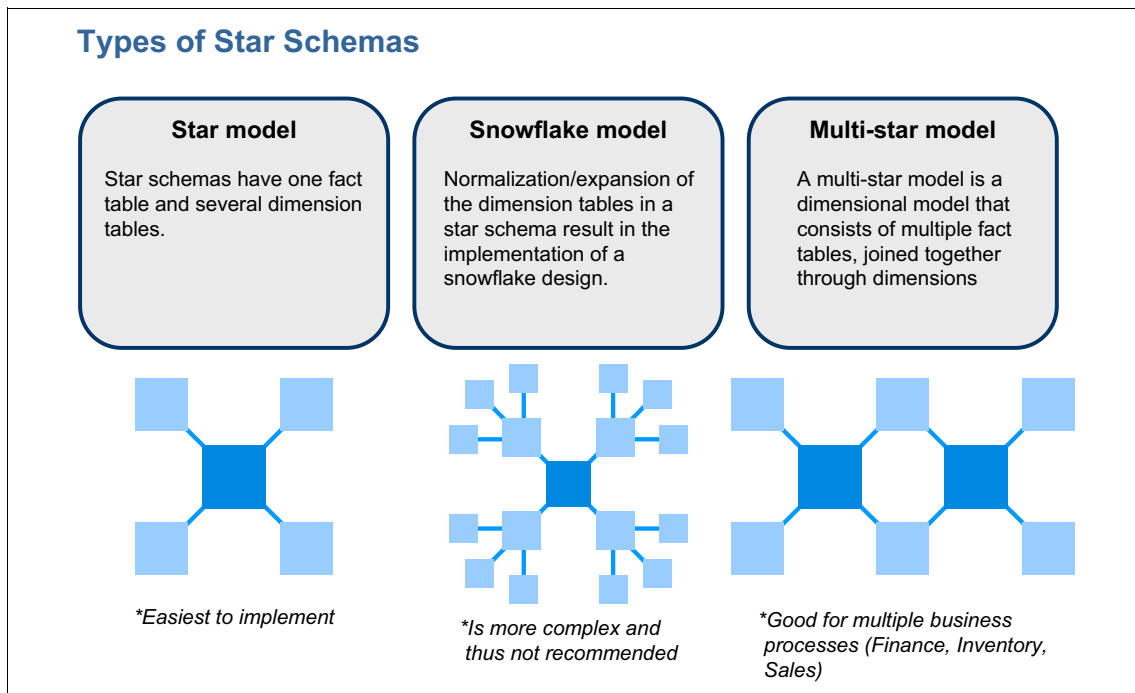


Figure 4-3 Different types of star schemas

4.2 The logical modeling process

As discussed in the previous section, to provide the most efficient database design with which to answer your business questions, you need to implement a star schema rather than the 3NF schema currently employed within the OLTP database. However, creating a model in the form of a star schema that represents a business operation is not a simple process. To aid in this development process, IBM has provided tools such as InfoSphere Data Architect which can create, as a starting point, a basic logical data model from a converted reverse engineered physical data model of your OLTP database. This base logical data model can then be developed, still utilizing InfoSphere Data Architect, into a form representing a star schema.

This process of developing a model representing a star schema is known as dimensional modeling, and is most successful when implemented as a four-stage process as described by Dr. Ralph Kimball¹. These stages are:

- ▶ Identify the business process.
- ▶ Define the granularity.
- ▶ Define the dimensions.
- ▶ Identify the facts.

Alternative modeling technique: Within this book, we focus on dimensional modeling as a technique for modeling a data warehouse.

There is, however, an alternative in the form of entity relationship (E/R) or entity attribute relationship (EAR) modeling, which is related and aligned to 3NF and OLTP modeling.

4.2.1 Identify the business process

The 3NF model implemented as the production database within a company normally covers a wide range business functions, both for efficiency and performance. A useful example is a production database that catered for the needs of sales, order management, inventory, and procurement.

However, to best answer your business questions, you have to separate these business processes into multiple dimensional models. Thus, the first stage is to identify these business processes and their relative components within the database.

¹ Ralph Kimball et al., *The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, 2008.

4.2.2 Define the granularity

Stage two of the process is to determine the right level of detail that must be contained within the model and ultimately, within the fact table. As an example, a model might contain a complete record of a sales transaction as one entry within the fact table, with all the products sold contained within the one entry or maybe not even listed. In this case, the lowest level is be the individual sales transaction.

An alternative can be to set the product as the lowest level, thus requiring multiple entries within the fact table to represent just one sales transaction. The *grain* (detail) of the fact table has a significant impact on the data warehouse from business, technical, and development points of view.

From a business perspective, a dimensional model designed to the lowest level of grain is easy to change, with new dimensions or facts being easily added. Thus, the model can maintain future capability and be flexible enough to answer questions at a lower level of detail.

From a technical perspective, the grain of a table has a major impact on the size of the star schema and thus the size of the tables. This has an effect on table implementation and the performance of transform and query operations.

From a development perspective, the development team might have to consider more dimensions and have a greater understanding of the underlying production tables. They might also have to perform a greater number of, and more complex, transforms for getting the data into the warehouse.

At this point in the modelling process it is worth investigating documents and account forms from within the business; items such as order forms, invoices, and sales receipts can help determine an acceptable grain level. You also must consider time- and date-dependent information, because many of the business questions that you aim to answer can be time related. An example might be “What was the number of sales of a particular item on a particular date.” In this scenario, you have to model time as one of your dimensions. You do, however, have to check that the source data does contain information at the day, month, or year level.

4.2.3 Identify the dimension tables

Dimensional tables are a major factor in the construction of a Star Schema dimensional model. As illustrated in Figure 4-2 on page 84, one or more dimensional tables are connected to the fact table, with each normally containing detailed and relevant information about a section or element of the complete fact table.

Therefore, the dimensional model has to create a relationship between the fact table and its required dimensional tables. This is achieved by having individual columns within the fact table assigned to hold the foreign key of each primary key within the dimensional tables.

A dimensional table might, within your model, also be designed to contain additional information than that normally included within the associated production tables related to the business process being modeled. A useful example is the inclusion of extra columns within the dimension table for storing a “product type” or “suppliers details. This additional information will not normally be part of a sales receipt, upon which you have based your fact table. But by adding this extra, related information to the dimension table, query performance can be improved because fewer less table joins are required in answering business questions related to this additional information.

In stage three of developing the dimensional model, you must determine which tables currently implemented within the production 3NF model are actually required as dimension tables within the star schema dimensional model. However, this is not a simple process of only copying complete sets of tables from the 3NF model that represents your production database into your dimensional model as dimensional tables. Doing so does not produce an effective and efficient star schema. For performance reasons, the 3NF model, being based on the OLTP production database, might contain multiple related tables, with each table only containing a small number of columns from the complete related subject.

As an example, consider that the 3NF model of your production database has been implemented to contain a set of product-related tables. Your model contains a table holding the product names, a separate table for the type of product, and a final table of the supplier’s details. The reason is that this information is not critical to the core function of storing a sales transaction, and in all probability, will only be updated within the production database at infrequent intervals.

However, if this complete set of tables is copied into your dimensional model as the basis for the “Product” dimension, the resulting model will d have more in common with a snowflake schema than the more efficient star schema. It is at this point that the dimensional model has to move away from that of the 3NF model and its related set of tables.

To create the desired star schema, it might be necessary to collapse many of the individual tables in the 3NF logical model into one single table with a larger set of columns within the dimensional model. A side effect of this process is that the now larger dimensional table will contain a greater amount of repeating information and require a larger amount of storage than that given to the original

base tables. Figure 4-4 illustrates the principle of collapsing multiple tables from the production database into a single dimensional table.

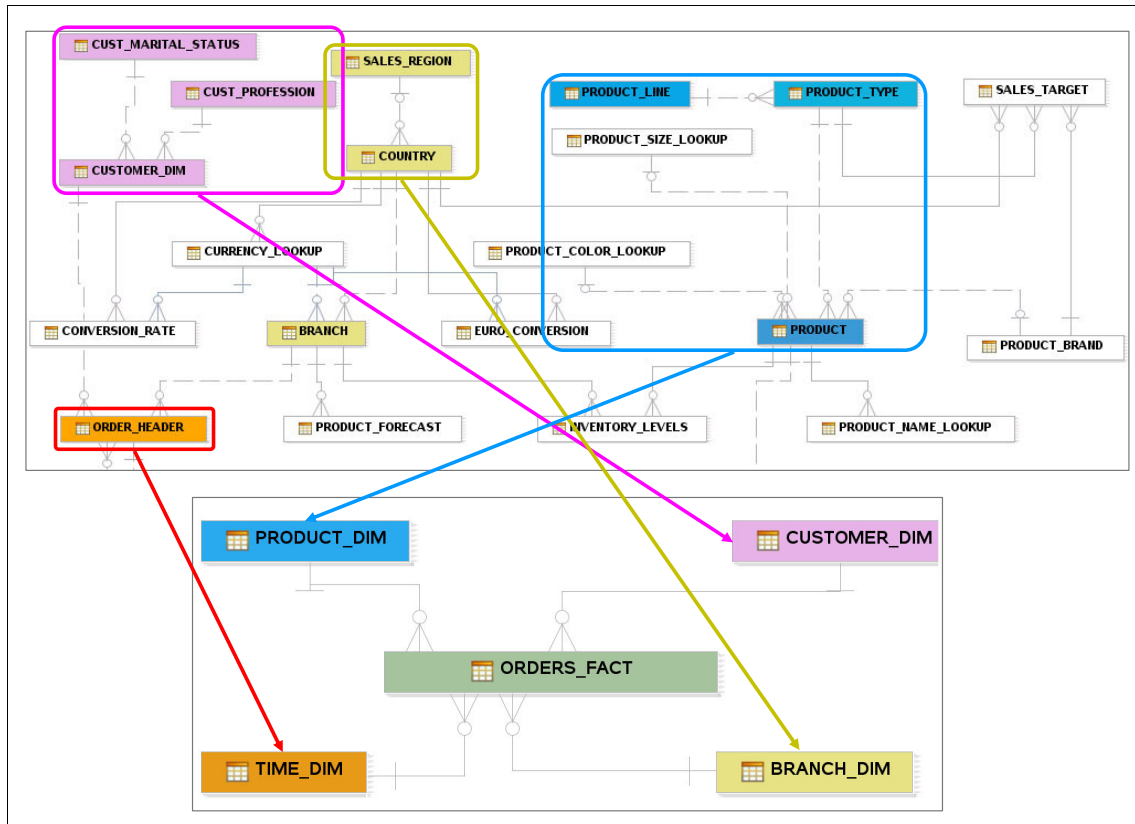


Figure 4-4 Collapse multiple tables into one dimensional table

Figure 4-5 illustrates the multiple tables within a production database that can be collapsed into a single dimensional table.

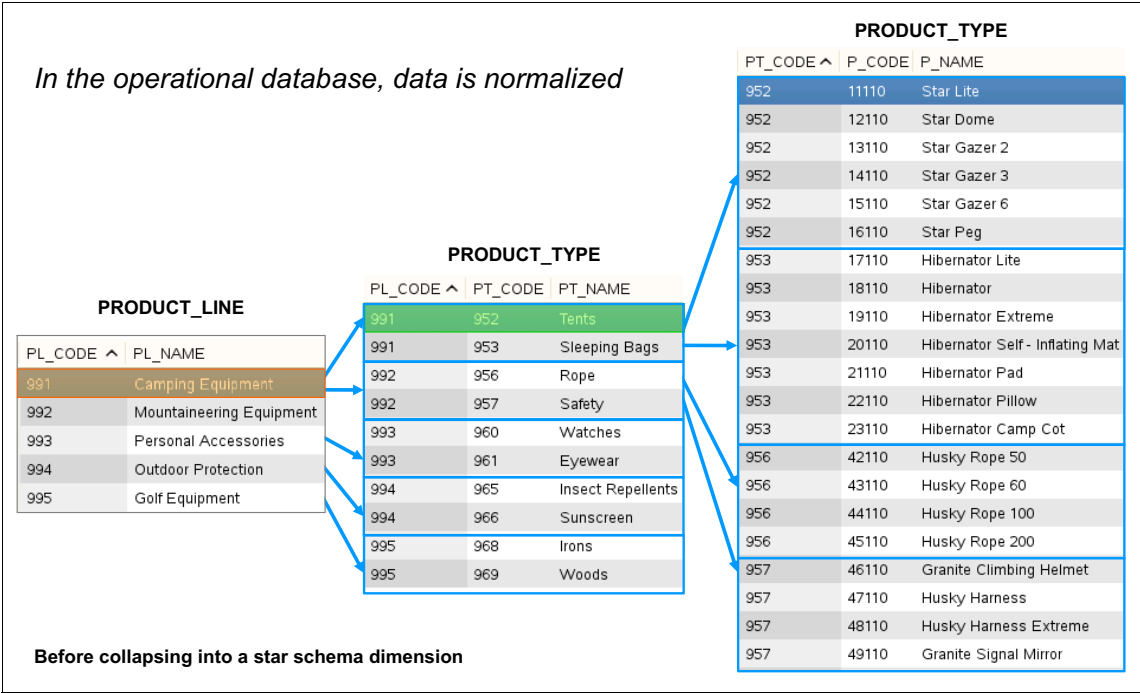


Figure 4-5 Multiple tables within a production database

The process of collapsing or merging tables together to create a single dimensional table is often referred to as a table being *de-normalized*. Figure 4-6 on page 91 illustrates the logical de-normalized dimensional model of the table in Figure 4-4 on page 89.

In the warehouse database, data is de-normalized

PRODUCT_DIM	PL_CODE	PL_NAME	PT_CODE	PT_NAME	P_CODE	P_NAME
	991	Camping Equipment	952	Tents	11110	Star Lite
	991	Camping Equipment	952	Tents	12110	Star Dome
	991	Camping Equipment	952	Tents	13110	Star Gazer 2
	991	Camping Equipment	952	Tents	14110	Star Gazer 3
	991	Camping Equipment	952	Tents	15110	Star Gazer 6
	991	Camping Equipment	952	Tents	16110	Star Peg
	991	Camping Equipment	953	Sleeping Bags	17110	Hibernator Lite
	991	Camping Equipment	953	Sleeping Bags	18110	Hibernator
	991	Camping Equipment	953	Sleeping Bags	19110	Hibernator Extreme
	991	Camping Equipment	953	Sleeping Bags	20110	Hibernator Self- Inflating Mat
	991	Camping Equipment	953	Sleeping Bags	21110	Hibernator Pad
	991	Camping Equipment	953	Sleeping Bags	22110	Hibernator Pillow
	991	Camping Equipment	953	Sleeping Bags	23110	Hibernator Camp Cot
	992	Mountaineering Equipment	956	Rope	42110	Husky Rope 50
	992	Mountaineering Equipment	956	Rope	43110	Husky Rope 60
	992	Mountaineering Equipment	956	Rope	44110	Husky Rope 100
	992	Mountaineering Equipment	956	Rope	45110	Husky Rope 200
	992	Mountaineering Equipment	957	Safety	46110	Granite Climbing Helmet
	992	Mountaineering Equipment	957	Safety	47110	Husky Harness
	992	Mountaineering Equipment	957	Safety	48110	Husky Harness Extreme
	992	Mountaineering Equipment	957	Safety	49110	Granite Signal Mirror

After collapsing into a star schema dimension

Figure 4-6 Example of a de-normalized dimensional table

Dimension tables contain attributes that describe fact records in the fact table. Some of these attributes provide descriptive information. Other attributes are used to specify how fact table data is to be summarized to provide useful information within the business report. Dimension tables contain hierarchies of attributes that aid in summarization.

Note the following key points to consider about dimension tables:

- ▶ Each dimension table has one and only one lowest level element, or lowest level of detail, which is the granularity of the dimension.
- ▶ Have each non-key element (other than the surrogate key) appear in only one dimension table.
- ▶ If the data for the dimension tables is drawn from different sources, it is advisable to use surrogate keys as the primary keys in all the dimension tables.

- ▶ Nearly all dimensional models have one or more date and time dimensions. These types of dimensions must be handled differently from other dimensions such as geography or products. This is due to the additional attributes that can be applied to date and time dimensions.
- ▶ If a dimension table includes a code then, where possible, include the code description as well. For example, if a product is identified by a product code, and each code represents a product name, include both the code and the product name in the dimension table.
- ▶ Normally, avoid having dimensional models include more than 10-15 dimension tables. If your model has more, look for dimension tables that can be merged.
- ▶ Include the primary keys (surrogate keys) of the dimension tables in the fact table as foreign keys. In this context, the surrogate keys are a unique identifier for each row within a table.

If a unique key or identifier is required for each row within a table, and the application data in the table does not provide within a single column, a new column containing a derived or calculated unique identifier value is added to the table. For performance reasons, the data type of this column is to be a numeric of type INTEGER.

- ▶ A dimension table may have one or more dummy or unrelated records. This is because a foreign key in a fact table can never be null; that by definition violates referential integrity.
- ▶ The rows in a dimension table establish a one-to-many relationship with the fact table. For example, there might be a number of sales to a single customer, or a number of sales of a single product.
- ▶ The dimension table contains attributes associated with the dimension entry; these attributes are rich and business-oriented textual details, such as product name or customer name. These attributes serve as report labels and query constraints.
- ▶ Have each dimension attribute aim to take on a single value in the context of each measurement inside the fact table.

It is worth noting that the quality of a good dimensional model is directly related to the quality of attributes present inside these dimension tables. The dimension table attributes appear as report labels within the business reports generated by BI tools such as IBM Cognos.

4.2.4 Identify the fact tables

The final process of developing a dimensional model is to identify tables within the 3NF model that have the potential to become the required fact table at the

center of the star schema. Different from the dimensional tables, the fact table must align to a business process or action and provide the necessary means to answer questions related to this process or action. Similar to the process of de-normalizing a dimensional table, the final fact table might have to be a combination of one or multiple tables from the 3NF model.

A fact table also must contain, along with the links to the dimensional tables, a set of columns that provide the data on which to measure the business process on which the fact table is based. These data fields are often called the “facts” or “measures” of the fact table.

Consider the example of modeling a sales transaction. The fact table might represent the business process of a customer buying a set of goods. In this case, the number of items sold, either individually or as a whole, become the set of measures or facts. Thus, in answer to the business questions of “How many of product Y did a customer buy?”, you can use the measures within the fact table to answer “They bought X amount of product Y.”

Figure 4-7 illustrates an example fact table within a dimensional model.

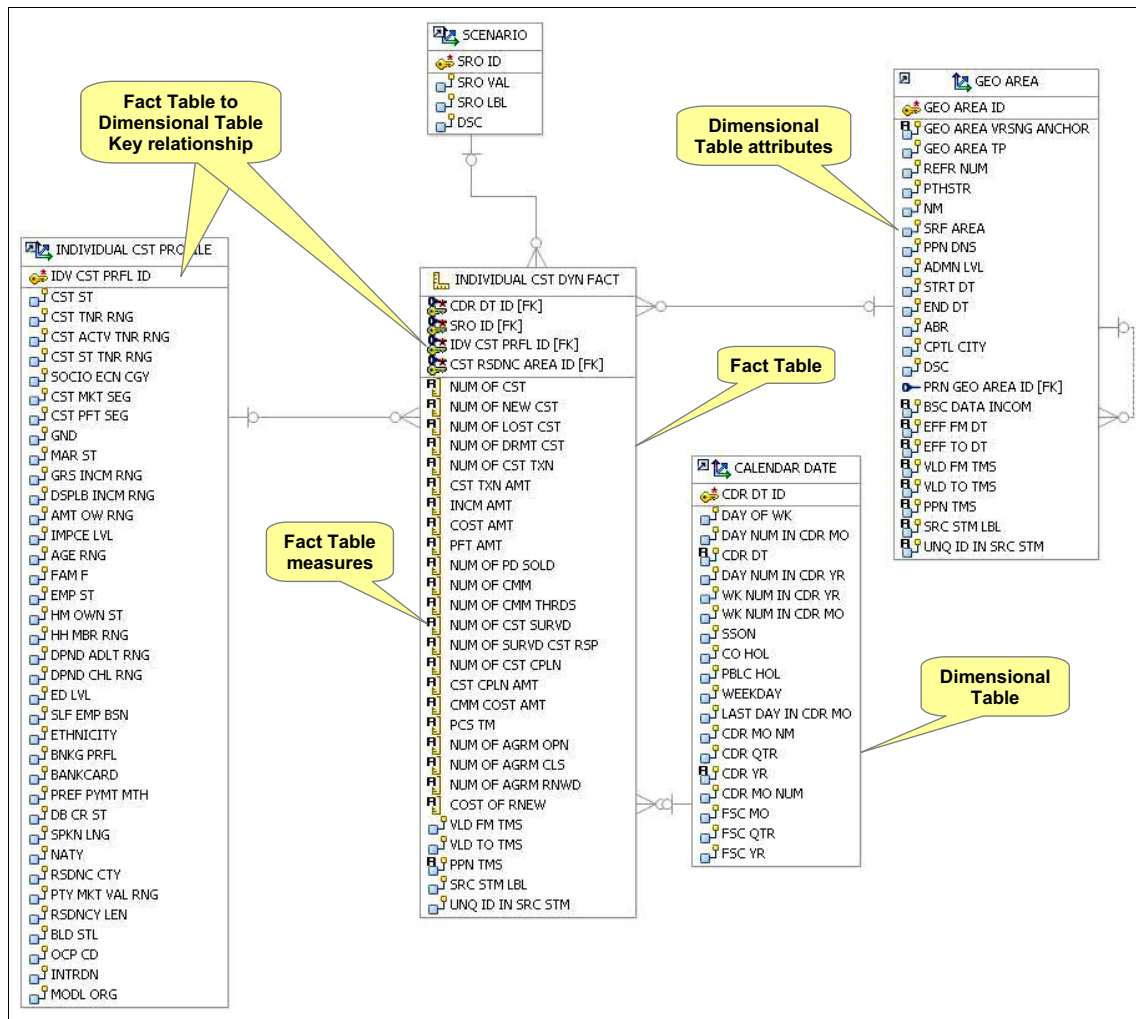


Figure 4-7 A fact table within a dimensional model

When developing a fact table, keep the following points in mind, because they have considerations for the implementation of the model:

- ▶ For best performance, use an INTEGER data type for the link columns that are used to join the fact table to the dimensions tables. Use a surrogate key if the natural key is of a different data type.
- ▶ Link columns are created as NOT NULL to ensure that the connections to the dimensions tables can be performed during queries.

- ▶ Define an index for each of the link columns.
- ▶ Define a link column as a foreign key to the related dimension table, either as enforced or as an informational constraint.

4.3 InfoSphere Data Architect and logical dimensional modeling

IBM InfoSphere Data Architect is an advanced collaborative data design client tool for the development of data-related projects. It allows developers to discover, visualize, create, and edit different types of data elements including both physical and logical data models. Based on the Eclipse Workbench and sharing many similarities with IBM Data Studio and IBM Design Studio, developers can quickly become productive.

IBM InfoSphere Data Architect can either be purchased from IBM as an individual product or obtained as part of a suite of products, combined under the umbrella of single product licence, as in the case of the InfoSphere Warehouse Advanced Enterprise edition.

Figure 4-8 shows the working sections of InfoSphere Data Architect.

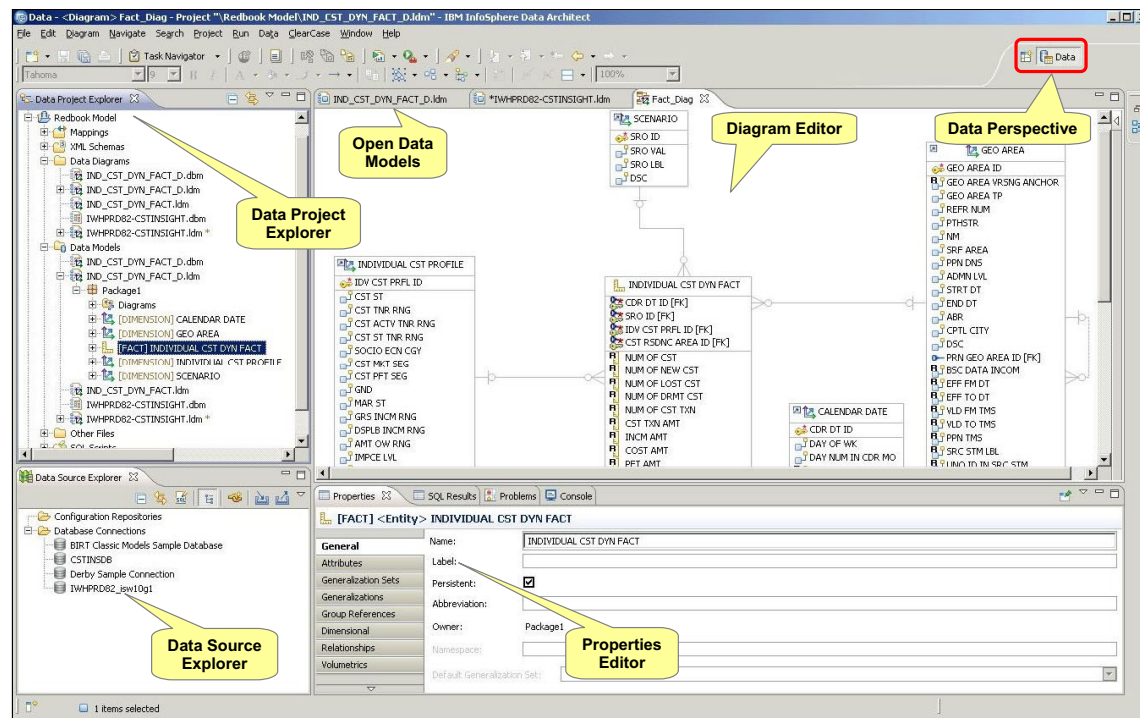


Figure 4-8 Layout of InfoSphere Data Architect

The aim of this section is to demonstrate how to use InfoSphere Data Architect as a development platform in the creation of dimensional models. These models then become the basis upon which to implement the InfoSphere Warehouse in support of its role at the center of the business analytics.

Our examples: The examples detailed in this section are all based on the schemas and tables contained in the IWHPRD82 database hosted within the InfoSphere Warehouse VMware image detailed in 3.3, “Example architecture used in this book” on page 77. This is a demonstration database that is created from the models and schemas supplied as part of the InfoSphere Warehouse Customer Insight Pack.

As such, the implementation of the database has already been developed from a dimensioned logical star schema model. Therefore, for our exercises in this chapter, we have to apply a small element of artistic license. As an example, the modeling process normally starts with a physical model of a OLTP production database that we did not have, so we started with a physical model of our OLAP database.

4.3.1 A starting point: The physical data model

Here we aim to create a logical data model of the target OLAP data warehouse by first creating a physical data model of the OLTP production database. In our example, this is a physical data model of the IWHPRD82 database on our lab systems. It is worth noting that you can, as an alternative, always start with a new blank logical data model in InfoSphere Data Architect.

To start this process, perform these steps:

1. Start InfoSphere Data Architect and create a workbench storage area, if one has not already been assigned.
2. Create a new Data Project to store your work by selecting from the main menu **File** → **New** → **Data Design Project**.
3. Within the wizard, give the new project a name, for example, Redbooks Model, then select **Finish**. This creates a new project within the Data Project Explorer window.

Note that a number of subsections under the main project name have been created automatically. These are directories on the file system within the workbench area. Those of interest to us are the Data Diagram and Data Model directories, which currently are empty.

4. Within the Project Explorer area, right-click the **Data Model** directory and select **New** → **Physical Data Model**.
5. In the opened wizard, edit the file name “Physical Data Model” to be something more meaningful. We use IWHPRD82-CSTINSIGHT, which is the

name of our database plus the name of the schema of most interest to us. Then select **Create from reverse engineering**. Click **Next**.

In our case, we reverse-engineer our model from a live database. However, it is also possible to use a DDL file of the source database produced by the db2look utility.

6. Within the next wizard panel, select **Database**. Click **Next**.
7. From the list of databases shown, select the source that is going to act as the base for your model. In our example, it is IWHPRD82. Click **Next**.
8. The next wizard panel lists all the schemas discovered from the selected database. Any of those listed can be selected to be part of the model. In our example, we are interested in CSTINSIGHT. Click **Next**.
9. The wizard then lists the database elements that are required to be pulled into the physical model. At this point the defaults are sufficient for our needs, so click **Next**.
10. At this point we are asked if we want to produce an Overview diagram, and whether it is to contain the relationships. Select **Yes** to the Overview diagram, and click **Finish**.

The final process of actually creating the physical data model might take a few minutes while the tool interrogates the live database and records those elements of the database needed in the model. On completion, the model is added under the Data Model directory of the Project Explorer panel and opened to allow browsing of the database elements.

Notice, as shown in Figure 4-9 under the Data Diagrams directory, a file is also named the same as the data model. This file is the overview diagram of the physical data model.

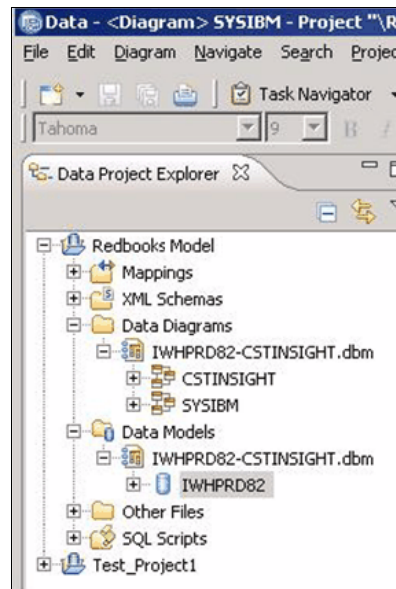


Figure 4-9 View of the Data Project Explorer panel within InfoSphere Data Architect

Opening the overview diagram (using a double-click) at this stage displays a busy and confusing view because it contains all the tables within the selected schema. See Figure 4-10.

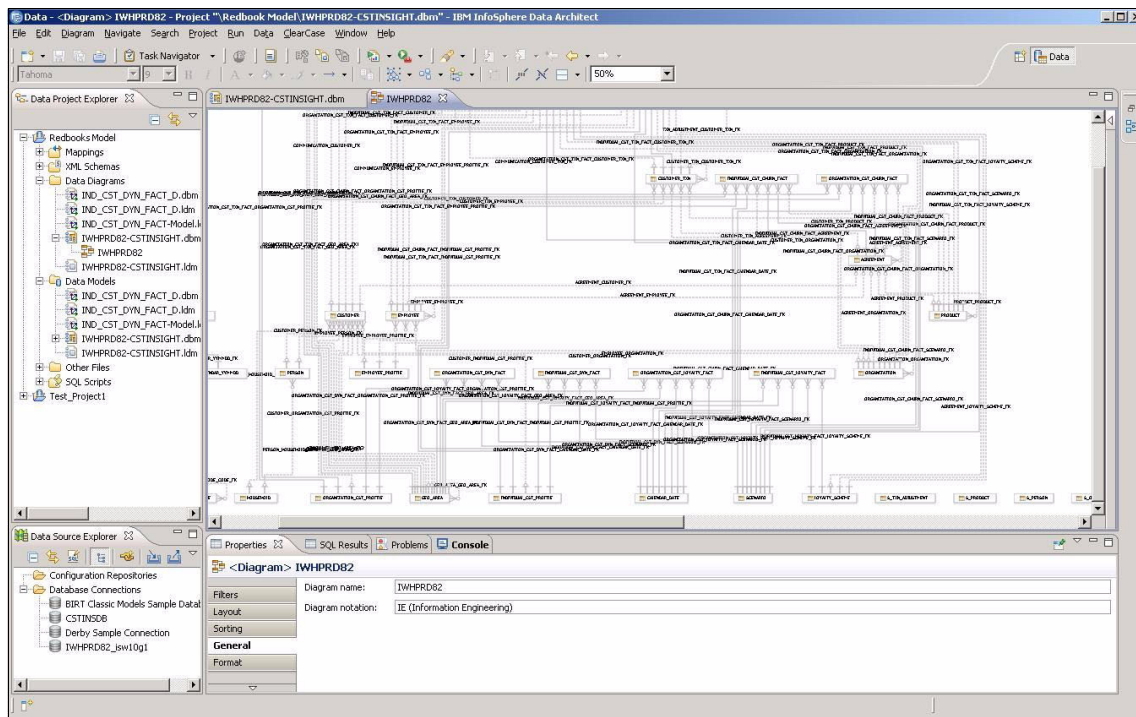


Figure 4-10 Physical model within InfoSphere Data Architect

Before moving forward to develop our logical data model, it is worth taking the time to explain what a “physical data model” actually is.

A physical data model is central to all the development tasks required to implement an InfoSphere Warehouse. It contains the metadata for the representation of an actual physical database. This include schemas, tables, columns, views, stored procedures, table spaces, users, and all the elements that make up a relational database. The physical data models are not limited to simply the representation of DB2 databases; they can also be used to represent a number of relational databases of different vendors such as Informix, Derby, MySQL, and so on.

A physical data model can also be used during the modeling process for OLAP objects to represent items such as cubes, fact, and dimensional tables, as is demonstrated later in this chapter.

Within IBM InfoSphere Data Architect, a physical data model is implemented as an entity relationship model, with either information engineering (IE) or unified modeling language (UML) notation. Figure 4-11 demonstrates the database elements represented within a physical data model.

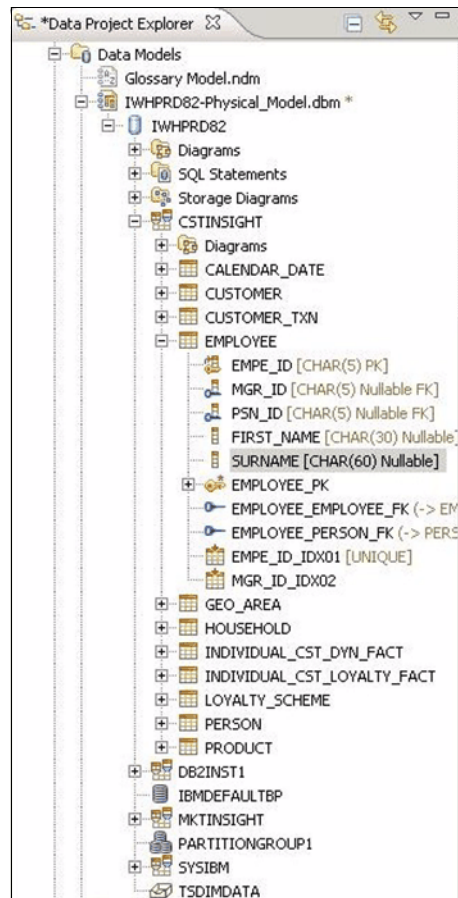


Figure 4-11 Relational database elements represented within a physical data model

Physical data model files can be exchanged between IBM InfoSphere Data Architect and other client development tools such as IBM Data Studio and IBM Design Studio.

For more details about the development, and implementations around physical data models, refer to *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

4.3.2 Create a logical data model

As previously mentioned, a small amount of allowance is needed regarding our example because our physical data model has been engineered from our OLAP database, rather than as done traditionally from an OLTP database. However, the steps demonstrated in our example to create a logical data model remain the same.

Follow these steps to create a logical data model:

1. Within IBM InfoSphere Data Architect, highlight the physical data model created previously, in our case IWHPRD82-CSINSIGHT.dbm. Right-click to display a menu. Select **Transform to Logical Data Model**.
2. Select the **Create new model** option, then click **Next**.
3. Keep the default values by clicking **Next** for the next two panels.
4. At this point, the wizard reports that the transform has completed. Select **Finish** to save the new logical model.

Figure 4-12 shows our newly created logical data model IWHPRD82-CSINSIGHT.Idm under the Data Model directory on the Data Project Explorer. Notice that all the tables have a blue square by their name, and the reason for this is explained in 4.3.3, “Add dimensional notation to the logical data model” on page 104.

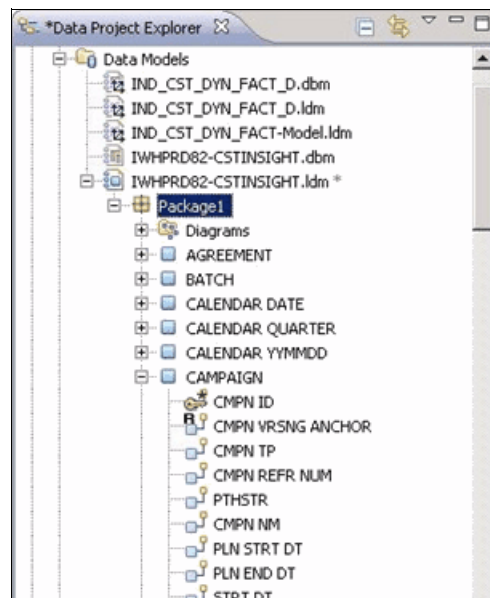


Figure 4-12 Logical data model within IBM InfoSphere Data Architect

At the moment our new logical data model contains a set of entities that are the representation of all the tables within the selected schema of the physical data model. To ease the modeling process, you might want to create a number of smaller logical data models that contain what amounts to a single fact entity and its associated dimensional entities. These individual logical data models can always be joined together at a later stage.

To keep our model simple for the purposes of demonstration, we create a new logical data model that focuses on the single fact entity `INDIVIDUAL_CST_DYN_FACT` and its associated dimensional entities. This fact entity represents individual customer dynamics. We perform the following steps:

1. Create an empty logical data model.

Select and right-click the Data Models directory from the open project within Data Project Explorer. Select **New** → **Logical Data Model**. Within the wizard, edit the file name. We use `IND_CST_DYN_FACT`. Click **Finish**. This has now created an empty logical data model, into which you can copy and paste entities that are of interest to you.

2. Copy entities into the empty model.

We copy from our original large logical model `IWHPRD82-CSTINSIGHT.Idm` the following entities into our new logical model `IND_CST_DYN_FACT.Idm`:

- `CALENDAR_DATE`
- `GEO_AREA`
- `INDIVIDUAL_CST_PROFILE`
- `SCENARIO`

and the fact table:

- `INDIVIDUAL_CST_DYN_FACT`

3. Create an overview diagram.

In the new logical data model, select and right-click the fact entity (`INDIVIDUAL_CST_DYN_FACT`, in our example) to bring up the option menu. Select **Add to Overview Diagram**. This opens a wizard listing the dimensional entities within this new logical data model. Select **Automatically select referenced elements**. Click **OK**.

A new overview diagram is created and opened within the main working area, as shown in Figure 4-13.

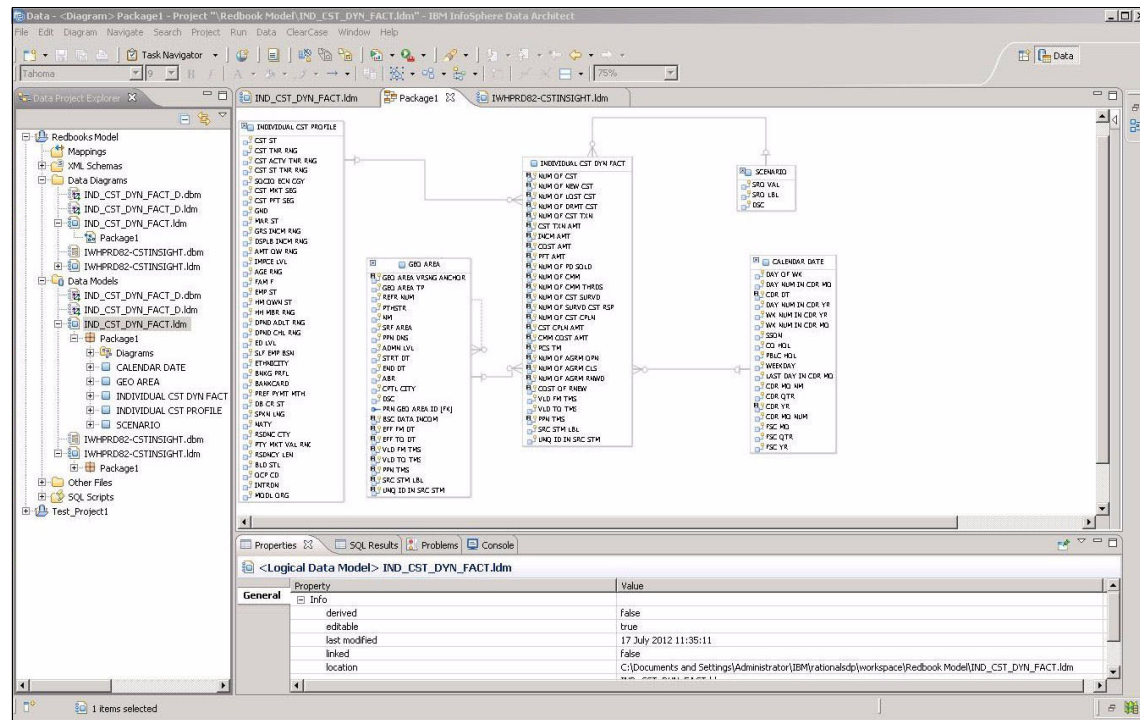


Figure 4-13 Reduced logical data model showing our star schema

It is at this point in the development process that you look to de-normalize your fact and dimensional entities within your logical data model, with the aim of creating a set of entities that represent a star schema model.



4.3.3 Add dimensional notation to the logical data model

The next stage in the process of developing our OLAP database is to add dimensional notation to our logical data model, thus moving it closer to the goal of a logical dimensional model.

We perform these steps to add dimensional notation to the logical data model with IBM InfoSphere Data Architect:

1. Open our Redbooks Model project and select the new logical data model IND_CST_DYN_FACT.Idm.

2. Right-click the model to display the option menu. Select **Use Dimensional Notation**.

As shown in Figure 4-14 on page 106, the icon depicting our logical data model in the Data Project Explorer area has now changed from a type of book icon  to a three arrow book icon  representing three dimensions. However, as shown within this figure, the icons representing the entities listed beneath these changed book icons still remain as blue squares.

3. Right-click the model again to display the option menu. Select the new menu option **Discover Facts and Dimensions**. The tool automatically discovers and annotates which entities within the model are either dimensional or fact.
4. During the discover process, a wizard window appears and asks if to generate the hierarchy for each discovered dimension. For our example, we select **Yes**.

At the end of this discover process, the blue square entity icons of the dimensional logical model changed to represent their associated function of either being a fact, dimensional, or outrigger entity type. Entities that the tool cannot determine an appropriate entity type remain a blue square. These particular entities can be set manually to an appropriate entity type using the Set Dimensional Notation To option on the right-click menu.

Note also that an additional menu option Generate Hierarchy appears that can be used to create a dimension hierarchy. Using the same menu, you also can add additional levels to the hierarchy.

Figure 4-14 shows our new dimensional logical model.

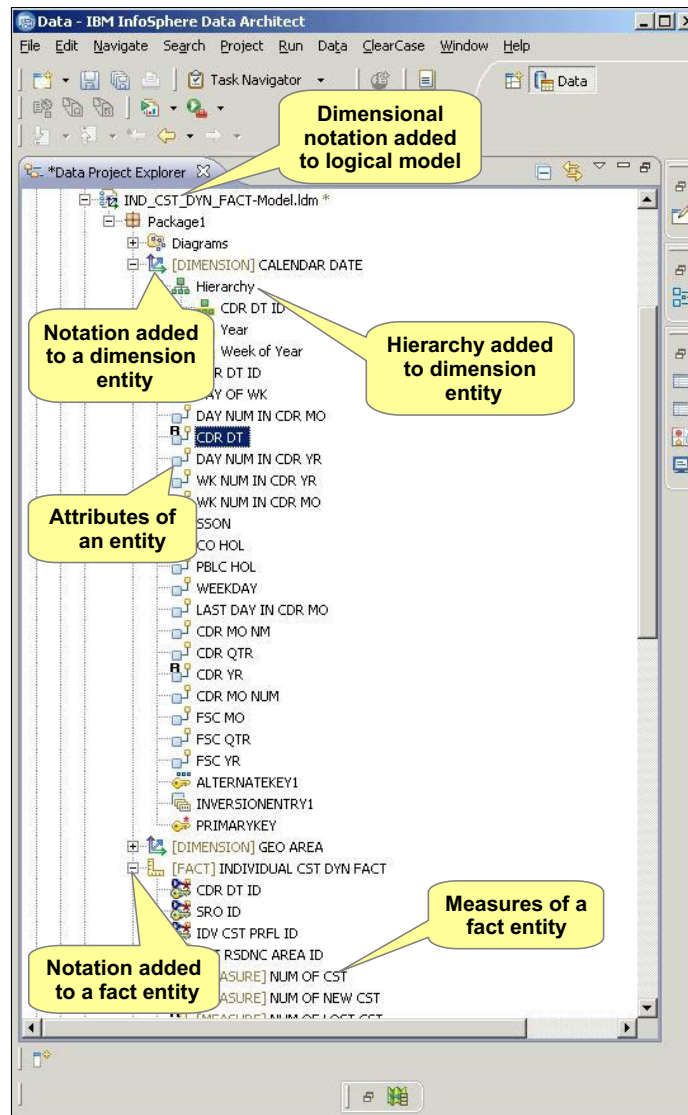


Figure 4-14 Dimensional notation added to a logical model

4.3.4 Create a dimensional physical data model to implement OLAP Data Warehouse

Having developed the star based de-normalized dimensional logical model, now you can implement this model in your InfoSphere data warehouse. An efficient

process for performing this action is to create a new physical data model, which in turn can be utilized to create a DDL file that can be used to build the new database.

Perform these steps to create the DDL file that represents the dimensional logical model:

1. From within InfoSphere Data Architect, select and right-click our dimensional logical model **IND_CST_DYN_FACT.Idm** to bring up the option menu. Select **Transform to Physical Data Model**.

In the wizard, make sure the **Create new model** option is selected, then click **Next**.

2. In the next panel, specify the version of DB2 or other relational database, that will host this database structure. In our example, we use DB2 10.1. Click **Next**.
3. In the final panel, you can change the schema name to an appropriate name. We use **CSTINSIGHT_NEW**. Click **Finish** to save the new model **IND_CST_DYN_FACT_D.dbm**.

From this new physical model you have to create a DDL file, which can be used to create the new schema within your InfoSphere data warehouse. Perform these steps:

1. Select and right-click the new physical model **IND_CST_DYN_FACT_D.dbm** to bring up the option menu. Select **Generate DDL**.
2. Within the wizard, accept the default database elements that must be created. Click **Next** until at the last panel, then **Finish**. The wizard then creates the DDL file for you.

This DDL file can then be implemented on your database server to create the database structure as defined within your model.

4.4 OLAP modeling and cubes

We have demonstrated that you can create an efficient database structure best geared toward OLAP queries, and ultimately, toward answering business intelligence questions by using one of the following methods:

- ▶ Using the derived logical dimensional model and its associated physical data model
- ▶ Taking advantage of the InfoSphere Warehouse Insight Packs included within InfoSphere Warehouse Advanced Enterprise edition

After the data is populated, this new structure is ready for use by the BI applications, such as IBM Cognos. Although these BI applications can directly access the data within the database, there is still a more efficient and effective mechanism that can be introduced to perform this task. OLAP modelling provides a method by which these BI applications can communicate with the users in a dimensional language closer to the way humans think, while at the same time providing a business abstraction layer between the data warehouse and these BI applications.

Within InfoSphere Warehouse, OLAP modeling is defined as a cube model, which can then be implemented as a cubing service. Although a star schema can begin to provide an abstraction to a relational database, it lacks the depth and richness of OLAP objects such as hierarchies, levels, and derived measures and attributes. OLAP cubes also provide an enhanced ability to capture business rules and logic, which is not as easy or intuitive within a relational database.

You might ask, why is this called a cube model?

Questions asked by business analysis such as “what is the sum of all sales” or “how many stores are there” can easily be answered by a normal SQL query of a relational database. When taken in context of your logical data model, these types of questions only relate to one dimension within the model. Business questions, however, are generally of the form that ask about something (a dimension) *by* something (another dimension), and might be even *by* something again. This form of multidimensional analysis is best suited to a cube model, where data can be analyzed from multiple perspectives.

Consider that you have to analyze the sales data for a number of products in a range of countries that you have stores, for all periods of time. To help visualize the data from these multiple perspectives, think of a cube with each of the axes representing a dimension. For example, consider country, product, and time. There is also an implicit fourth dimension measure, which is the sales amount.

Figure 4-15 on page 109 depicts this cube, which itself has then been divided into a number of mini-cubes. Each mini-cube contains a value (or measure) for that specific intersection of the country, product, and time. As an example, the darker mini-cube (shown in blue) represents the sales for the first quarter of 2008

(time dimension) for the grocery item Milk (Product dimension) that was sold in the United Kingdom (country dimension).

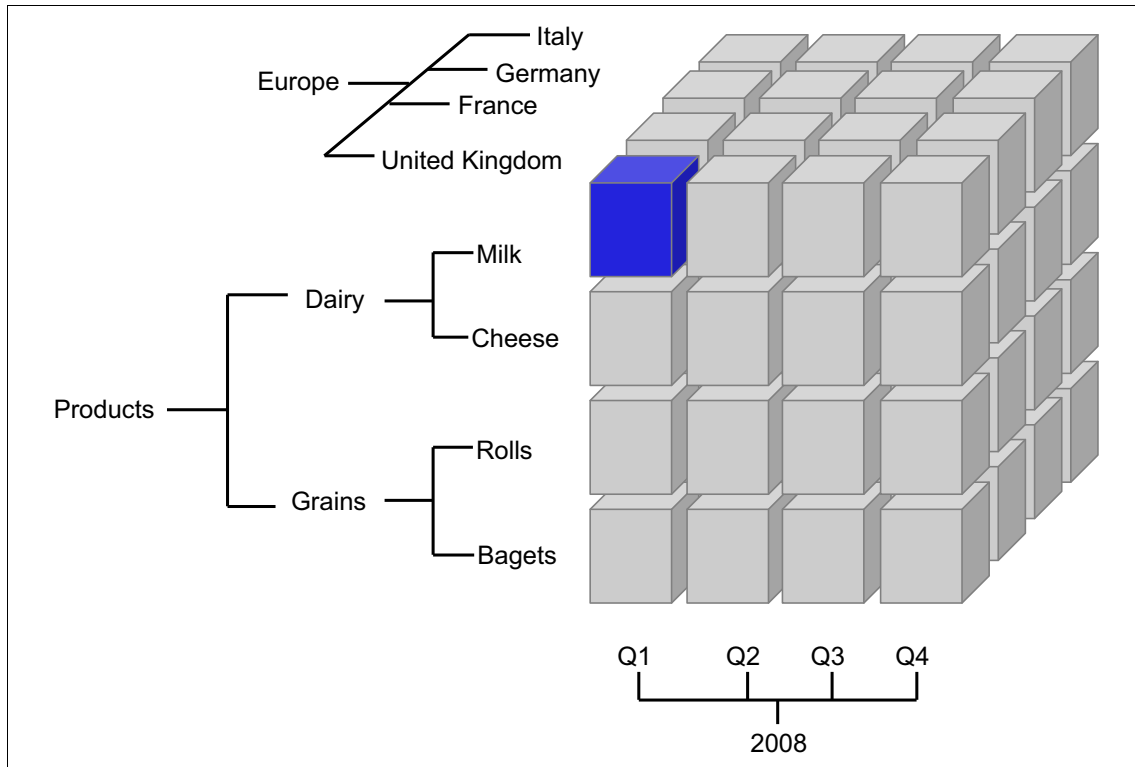


Figure 4-15 Example cube model

Even if a query only contains two of the dimensions, for example “what is the sales of milk in Q1”, there is still an implicit referral to the third dimension. Consider the previous example, if a country was not part of the question. In this case, all the measures for the two specified dimensions, Product (Milk) and time (Q1), are related to the highest level of aggregation of the country dimension, which in this case is Europe. So the answer is the total of Milk sold across Europe in Q1.

Do not create a cube containing too many dimensions with the idea of trying to answer all the many different business questions. It is preferable to create two or three smaller cubes with fewer dimensions, with the aim of answering many of those business questions separately. Even though there is no limit on how many dimensions you can add, a cube with 10 or 12 dimensions becomes difficult to manage and even more difficult to use. You have too many variables to deal with when analyzing data. A useful range is from four to eight dimensions. Create a

cube with as few dimensions as possible to respond to a specific group of related business questions.

A cube model, when implemented as a cubing service for OLAP, provides a substantial performance boost to analytic queries and greatly augments the high performance characteristics already supplied by DB2, especially when combined with elements of DB2 such as MDCs, MQTs, and DB2 powerful SQL optimizer. This combination can deliver high speed responses to high level summary and aggregate data queries, while providing access to large volumes of data.

A cubing service also adds in layers of intelligent caching of both data and metadata to efficiently manage and reuse previous query results and further reduce I/O and processing on the database. While pushing the simpler bulk OLAP operations such as selects and table joins onto DB2, which is highly suited to those operations, a cubing service offloads most of the complex multidimensional calculations and processing into its own server, thus reducing the workload on the database server.

In addition to the performance boost that cubing services provide, they can also be configured to implement a high level of user access and security. These levels of security are defined more inline with business rules, that is, restricting access to regions of a cube based on the role of the user. For example, a sales manager for a Northern region might not be allowed to access to the data from a Southern region's sales team.

OLAP database architecture

It is worth mentioning that the term “OLAP database” is a general term, and that over the years a number of approaches have been developed for implementing the most efficient OLAP database. The three most common approaches to an OLAP database are called MOLAP, ROLAP, and HOLAP. These three approaches are briefly described here:

- MOLAP is the term for multidimensional OLAP. This type of database is stored in a special, typically proprietary, structure that is optimized (often through some pre-calculation of data values) for fast query response time and multidimensional analysis. The database usually has a proprietary OLAP API that BI tools must support. Various OLAP APIs often offer several OLAP formulas, not found in standard SQL, that are frequently used when performing financial analysis.

MOLAP databases require that data be extracted from the data warehouse and loaded into the MOLAP database, duplicating the data. They also require both time and performance to perform the pre-calculations along all the dimensions, which often results in large data sets.

- ROLAP is the term for relational OLAP. The database is a standard relational database and the database model is often a star or snowflake schema. A

primary benefit of this implementation is the significant scalability capabilities offered by relational databases. An additional benefit is that the original data does not need to be copied or duplicated. Access is provided through standard SQL, which enables access to the data by a broader set of tools and applications.

ROLAP performance is typically considered slower than MOLAP because each query is an SQL query (or multiple SQL queries) and the query time is largely governed by the complexity of the SQL used, the number and size of the tables that must be joined, and the level of aggregation required to satisfy the query. To improve performance ROLAP databases often use summary tables to hold the aggregated data.

- HOLAP is the term for hybrid OLAP. As the name implies, it is a hybrid of ROLAP and MOLAP. A HOLAP database typically stores high level aggregates in a MOLAP database. Lower level detail data is stored in a relational database. By storing the lower levels of a dimension in relational format instead of in MOLAP, the MOLAP database size is reduced and the time required to load and perform the pre-calculation of the aggregate data is also reduced.

Queries that request data from the MOLAP section of the database will benefit from the fast performance that is expected from having pre-calculated data. Moreover, by storing the lower levels of the database in relational, the database as a whole (MOLAP and relational combined) can be extended to take advantage of the scalability benefits in the relational database.

The HOLAP database developer determines what level of granularity is in which database. HOLAP databases use the proprietary OLAP API of the MOLAP database.

In response to the need for an efficient OLAP database while still keeping the data within a relational database with all the associated benefits, IBM implements a number of features and additions to IBM DB2 that address these issues. As an example, IBM DB2 has Materialized Query Tables (MQTs). MQTs are summary tables that can be used in multidimensional or OLAP analysis. Costly table joins and complex calculations can be computed in advance and stored in an MQT.

MQTs also provide the advantage of being 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 whether it costs less to satisfy the query by using the base tables or by using the available MQTs, which thus improves performance. 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 an application viewpoint, it appears to be a MOLAP database using

multidimensional expressions (MDXs) as the query language. However, the data is not copied or persisted in another storage mechanism, but remains in the DB2 relational database.

Cubing Services thus is what we refer to as 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 a 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 while 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.

4.4.1 Cube model dimensional concepts

Before creating any cube models, it is important to understand the components that make up such a model, and to understand their association to the fact and dimensional tables that make up the dimensional model based on a star schema.

Cube models have four main core ingredients, which themselves have a number of additional elements. These four ingredients are listed here.

- ▶ Fact: derived from the fact table within the dimensional model, and which contain:
 - Measures: derived from the columns or calculated.
- ▶ Dimension: derived from the dimension table within the dimensional model, and which contain:
 - Hierarchies: aggregates of data.
 - Levels: the granularity of the data with uniquely identifiable attributes.
 - Attributes: identifiable entities.
- ▶ Joins: derived from the relationship between tables:
 - Joins between fact and dimension tables, in a star schema.

- Joins between multiple dimension tables, in a snow flake schema.
- Cube: an instance of parts or a complete cube model implemented as a cube service.

Figure 4-16 illustrates the components of a cube model and their relationship to the tables in a warehouse database.

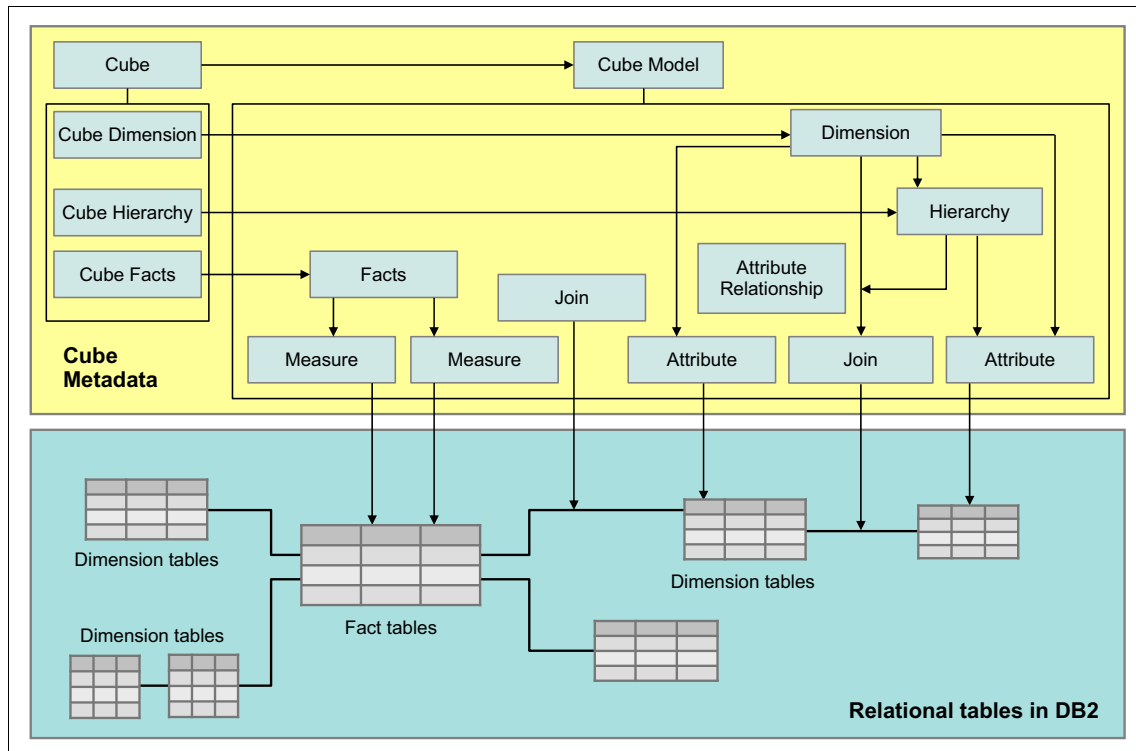


Figure 4-16 Relationship between cube model and tables in a relational database

Fact element

In a dimensional model, the fact table holds the central position within the star schema, with all the dimensional tables linked to this one table. In a cube model, the *fact element* maintains a similar central position, with the dimensional elements connected, this time as a defined join. This means, within the definition of the cube model, you have to define a set of SQL queries that join the fact table to the individual dimensional tables within the relational database.

A fact object also contains a set of measures or metrics that best describes how to aggregate data from the fact table, across dimensions. These measures or metrics can be drawn directly from fact table columns, or derived through

calculations. It is these measures or metrics that form the basis of what is to be analyzed within the cube model. As an example, they might be sales amounts, counts of product failures, or the number of admitted patients.

Within the cube model, understand that measures in themselves only become meaningful when used in the context of a set of dimensions. For instance, when used on its own, a sale of 300 units is not meaningful. However, when combined with dimensions for location and product line, the resulting measure takes on more meaning, that is, “A clothing line in the eastern sales region sold 300 units.” It is these measures or metrics in connection with the dimension objects, such as time or location, that allow you to manage your business, analyze options, and ask your BI questions.

The following types of fact measures can be implemented:

- | | |
|---------------------------|--|
| Calculated facts | These are calculations that are based on existing measures or column values in the relational database. For example, the sales amount might be calculated from the unit price of a item against the number of items sold.

These calculations can be performed by two methods within a cube service in InfoSphere Warehouse: SQL and MDX. For SQL calculations, the work is performed by DB2 and SQL functions. For MDX calculations, the work is performed within the Java application that is the cube service and which is better suited to complex calculations. |
| Additive facts | This refers to where a measure is rolled up across all dimensions and levels using a consistent aggregation function. An example of an additive fact is the sum of sales or cost of goods sold. Cubing Services also support average, count, standard deviation, max, min, and variance as aggregation types. |
| Non-additive facts | Non-additive facts cannot be rolled up through any of the dimensions in the cube model, which is typically the case for any measure calculations that result in ratios, averages, or variance percentages, for example. Cubing Services supports non-additive facts for calculated measures by setting the aggregation function to None, which results in that calculation being calculated after the values are aggregated |

Dimension element

In a star schema-based dimensional model, each dimension table contains a specific set of related columns, biased towards a particular topic such as time, location, or product range. Because a cube model is an abstraction of the star

schema, it too has dimension elements. And these in turn are defined as a set of related attributes that together describe one partition or aspect of the model, along the lines of its metrics.

Similar to the dimensions within the star schema, these related dimensions within a cube model organize the data in the fact object according to common logical entities, such as by time, by location, by person. These dimensions are then subdivided into additional components that are known as attributes, levels, and hierarchies, as described here:

► Hierarchies

Generally, for each dimension defined within the cube model, one or multiple hierarchies are defined. These hierarchies describe the relationship and structure of the referenced attributes, and provide a navigational and computational way to traverse the dimension. Individual hierarchies are divided into an ordered list of levels that represent the scope of dimension granularity from the lowest to highest.

During the creation of the hierarchy, you must also specify the hierarchy type. The following are hierarchy types that can be implemented within a dimension:

– Balanced hierarchy

All the branches in a balanced hierarchy descend to the same level and have the same corresponding number of levels. A good example of a balanced hierarchy is time, where the meaning and depth of each level, such as year, month, and week, are constant.

– Unbalanced hierarchy

In this context, although all the branches within an unbalanced hierarchy have a parent-child relationship, they have an inconsistent number of levels and might not all reach to the same depth. An example of this type of hierarchy might be that of an organizational chart where a large department has a sub-manager, but a smaller department does not.

– Ragged hierarchy

Similar to an unbalanced hierarchy, the branches of a ragged hierarchy can descend to different levels. However, an hierarchy becomes ragged when the parent of a member is not in the level immediately above that member and thus, one branch does not have an entity at all the levels. An good example might be a set of countries in which a larger country might be divided into states, then a city level. However, a smaller country just has a city level.

- Network hierarchy

In this situation, the levels do not have an inherent parent-child relationship, so their order is not important. But the network hierarchy still has attributes such as color, size, or product type.

In a relational database, the different levels of a hierarchy can be stored in a single table (as in a star schema) or in separate tables (as in a snowflake schema).

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.

- ▶ Attributes

These are the basic building blocks of the dimension element and are the identifiable aspects in the model with which to describe, organize, group and ask questions about entities with the cube model. In the simplest terms, attributes represent the basic abstraction of a database column. They can be expressed as a simple SQL expression that maps to a table column within a dimensional table, or as a more complex expression that can combine multiple columns within one or more tables.

Attributes often serve as headings for the columns of a report, so make them descriptive and easy to understand.

- ▶ Levels

Levels define the specific unique “slices” of each hierarchy or the specific granular layers within the hierarchy for each dimension definition. If a hierarchy contains more than one level, then these levels have a defined order within hierarchy structure. As an example, the natural levels of a time dimension are days, months, years. They are placed in a specific levels order to form a Calendar Time hierarchy; that is, the highest level will be years, followed by months, and then at the lowest level by days. Each member, a specific day or month, of the level is uniquely identifiable. In the context of a dimension table in a logical model or within the database, this property maps to the key column or a group of columns that uniquely identifies every row in the table.

Figure 4-17 shows hierarchy examples and the related levels.

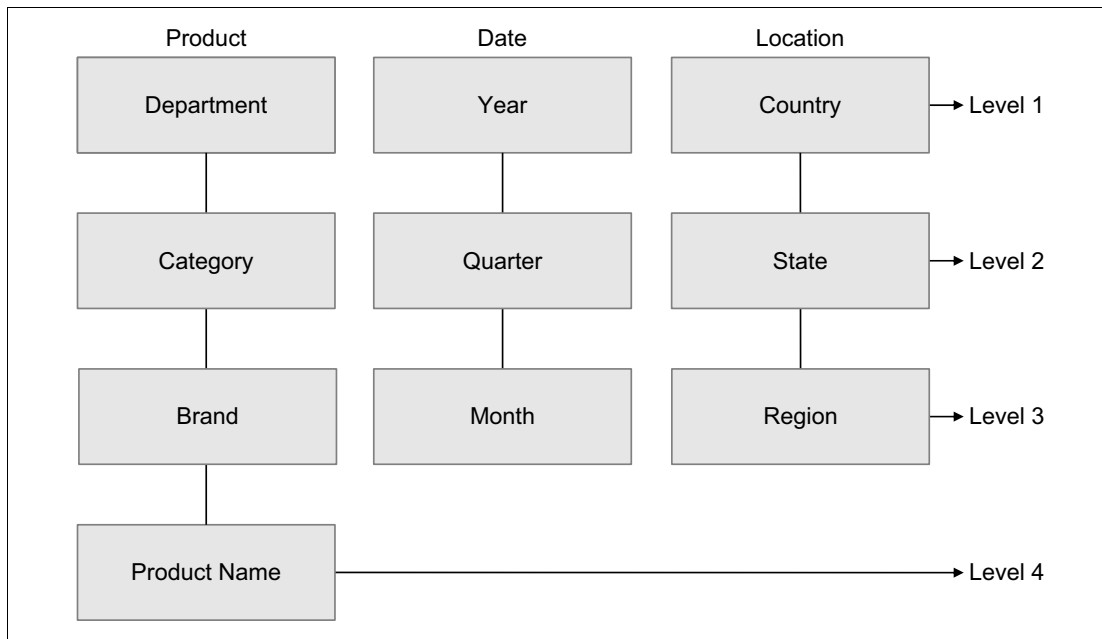


Figure 4-17 Example of Levels within a a group of hierarchies

You define a level by specifying both a level key and default attributes, which then must be placed in a defined order. It might be necessary to include related attributes to the level if the key attribute or default attribute are not understandable in terms of any reports needing to be produced. For example, the product name “Milk” is meaningful but a key value of “10” makes no sense in a report. There are three types of attributes that can be defined within a level:

- Level key attributes

These key attributes are one or more attributes that uniquely identify each of the items in the level. This must be the key column or a group of columns 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 item in the level. The default-related attribute is a required field. A example is a product name as the default-level attribute for a product dimension.

- Related attributes

Related attributes are all the optional attributes that provide additional information about the items in a level. These attributes are functionally determined by the level key attributes.

After that, you must add the level to a already created or new hierarchy along with the correct rollup order.

With regard to the dimensional elements of a cube model, those related to TIME might have to be considered as a special case. When defining a model for use as a cubing service within InfoSphere Warehouse, time calculations related to a time dimension, such as year-to-date (YTD), month-to-date (MTD), and others, use special time-related MDX calculations. For these MDX calculations to work, the type of the dimension must be set to Time versus Regular within the cube model. All other dimensions are normally of the type Regular.

Join element

The relationships between both the fact and dimension tables in a star schema-based dimensional model are defined by the use of primary and foreign keys. In an OLAP cube model, these relationships have to be linked to the actual tables within the relational database, and thus become defined as the method by which these tables are joined. Within a cube model, there are three places where the join elements can be defined:

- ▶ Fact-to-dimension join - Stored as part of the fact element within a cube model.
- ▶ Dimension-to-dimension join - Stored as part of the dimension element, this type of join is related to a snowflake schema, where not all the attributes of a cube model are within one dimension table.
- ▶ Fact-to-fact join - Not commonly found within a cube model, but it might be needed if not all the measures required can be found within one fact table.

Cube element

A cube that is referred to as a cube service is part of a cube model and can be considered as the implementation aspect of the OLAP model. A single cube model can contain several cube services that each get defined with a subset combination of the defined facts, dimensions, hierarchies, and levels from the OLAP cube model. A cube is the closest to an OLAP conceptual cube.

A cube service definition, unlike a cube model, can have only one hierarchy defined for each cube dimension.

In InfoSphere Warehouse, a cube service is implemented as an independent Java process, which is then managed by the InfoSphere Administration Console.

4.4.2 Define a cube model within Design Studio

As outlined in “Design Studio” on page 54, Design Studio is an Eclipse-based development workbench included as one of the features of the IBM InfoSphere Data Warehouse Client. Its appearance, as shown in Figure 4-18, is similar to that of InfoSphere Data Architect. As the main development environment for building a Data Warehouse solution, Design Studio contains a number of tools to ease this development process. One example is the development tool for the creation and deployment of cubing models and the resulting cubing services.

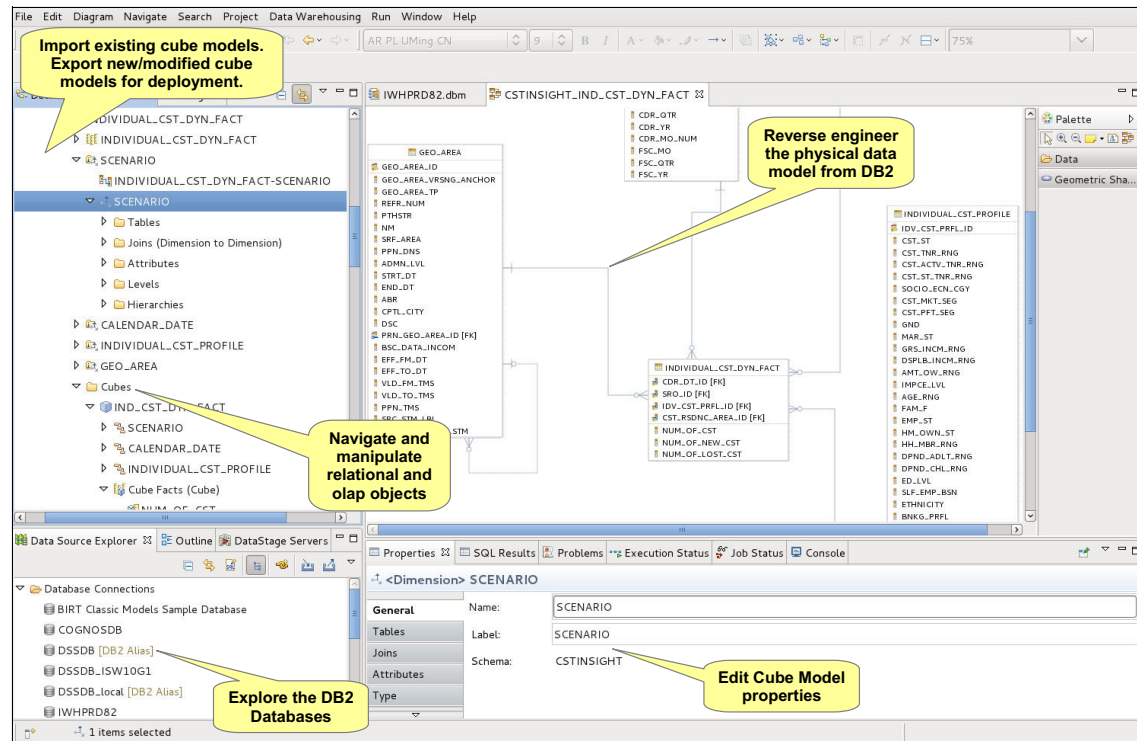


Figure 4-18 Use of Design Studio to develop cube models

Continuing on from our previous modeling section within this chapter, our example demonstrated here continues to be based on our Retail Store example.

Defining a new cube model

Perform these steps to define a new cube model using Design Studio:

1. Create a new project in Design Studio by selecting **File** → **new** → **Data Design Project (OLAP)**.

2. Provide the project a new name. We use **Redbooks_Cube_Model**. Click **Finish**. You can see the new created project within the Data Project Explorer, as shown in Figure 4-19.

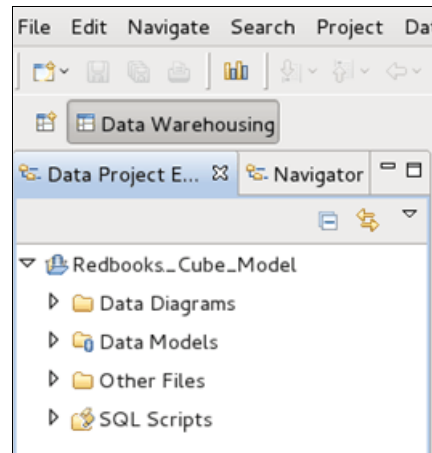


Figure 4-19 New OLAP data project within Design Studio

3. Create a physical data model.

A physical data model is required as the starting point to the OLAP cube model. Similar to those in Data Architect, there are a number of ways to create a physical data model: from new; by reverse engineering from a DDL file or live database; or by loading from Data Architect.

In our example we copy the physical data model from Data Architect. You can copy and paste the physical data model file either from an open Data Architect workbench, or from the native operating file system.

By right-clicking the menu, we copy and paste the IWHPRD82-CSTINSIGHT.dbm file from the native file system into the Design Studio under the subsection "Physical Model" within the project explorer panel. We also copy and paste the IND_CST_DYN_FACT_D.dbm file into the same area. Figure 4-20 on page 121 shows the Data Project area.

This second model file only has the tables from our star schema tables, which makes our example easier to work with.

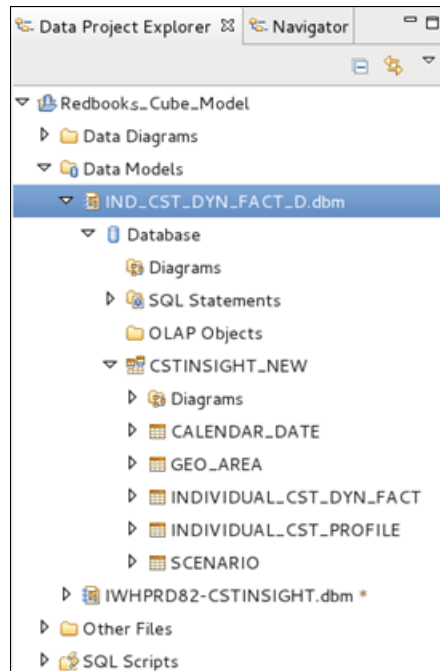


Figure 4-20 Physical model loaded into Design Studio

Within Design Studio, we only work with physical data models. This is different from the Data Architect, where we can develop both physical and logical data models. However, this physical data model within Design Studio is extended to include a directory for the definition of OLAP objects that make up the cube model.

4. Add the cube model.

Expand the new physical data model (in our case, IND_CST_DYN_FACT_D.dbm) to reveal the new section labeled OLAP Objects. Right-click **OLAP Objects** and select **Add a Cube Model**. See Figure 4-21.

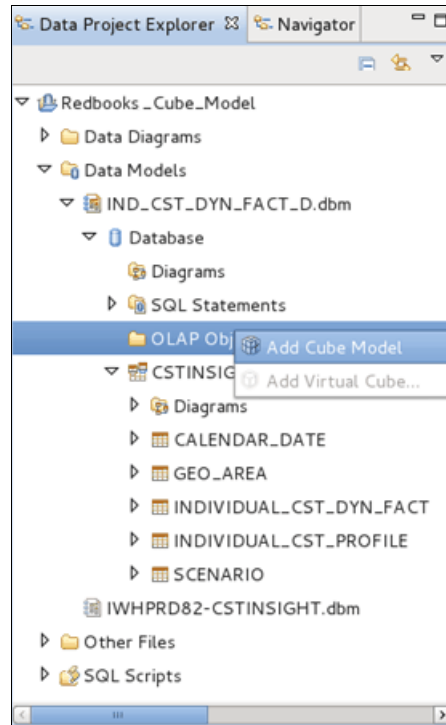


Figure 4-21 Creating a new cube model in Design Studio

5. A new panel appears, listing all the tables within the model. Select the table that will become the fact table within the new cube. In our case, this is **INDIVIDUAL_CST_DYN_FACT**. Click **Next**.
6. Design Studio pulls in the required dimension tables automatically and creates both the cube model and the elements that make up a cube service. The wizard displays both the new cube model and under the folder called Cubes the definition of the new cube service, as shown in Figure 4-22 on

page 123. At this point you can edit these object names, or select **Finish** to complete the initial model definition.

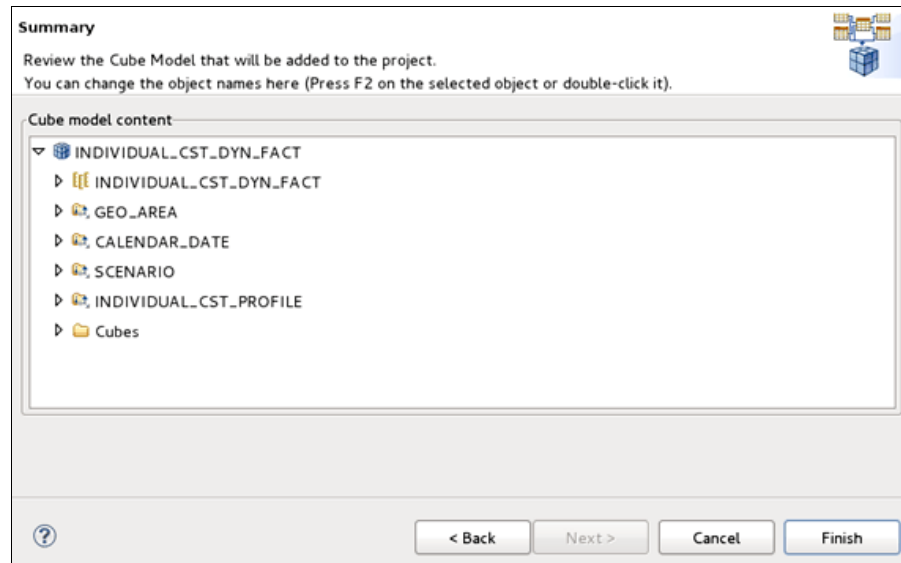


Figure 4-22 Design Studio Wizard creates a new cube model

Add attributes to a cube model

Design Studio has now created both a cube model and the definition of a basic cube service. However, this default cube might not contain the correct elements or all of the information that you want to include within the final reports. At this stage you can either create a new cube model, or edit the current definition.

As an example, our fact table `INDIVIDUAL_CST_DYN_FACT` has a relationship with more than three dimension tables. In our basic cube service, we have selected `CALENDAR_DATE`, `SCENARIO`, and `INDIVIDUAL_CST_PROFILE` as the dimensions of our default cube. However, we want to replace `SCENARIO` with the dimension for `GEO_AREA`. We also want to alter a few of the other dimension definitions within our model.

Follow these steps to add attributes to the cube model just created:

1. Within the Data Project Explorer panel, expand the cube model definition and the `CALENDAR_DATE` dimension down to the Attributes directory. Right-click and select **Add Attributes from Columns** from the menu.
2. From within the pop-up panel, select and rename the following columns of the `CALENDAR_DATE` table displayed:
 - Rename `CDR_YR` YEAR

- Rename CDR_MO_NM to MONTH
- Rename CDR_MO_NUM MONTH_OF_YEAR
- Rename DAY_NUM_CDR_YR to DAY_OF_YEAR

Click **OK** to add these new attributes to the model, as shown in Figure 4-23.

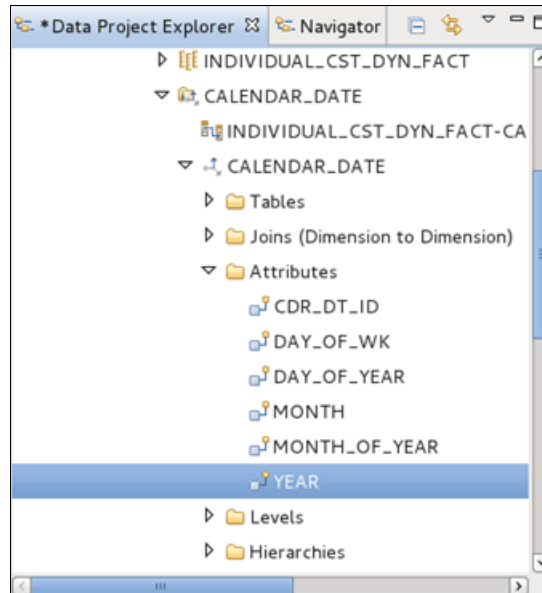


Figure 4-23 New calendar attributes added to dimension

3. Expand the **Levels** directory and right-click the Levels heading. Select **Add Level**. Give the new level a name; in our case, we start with YEAR. Select the

Time or Date type option, and then **Years** as shown in Figure 4-24. Click **Next**.

Properties
Specify the name, label and type of the level.

Name:

Label:

Type

☐ Unknown

☐ Regular

☒ Time or Date

☐ Complete date ☐ Quarters ☐ Hours

☐ Undefined time ☐ Months ☐ Minutes

☒ Years ☐ Weeks ☐ Seconds

☐ Half years ☐ Days

? < Back Next > Cancel Finish

Figure 4-24 Define new level within our model


4. In the next panel, select the **YEAR** attribute from the list shown. Use the right arrow button () to move the attribute over to the selected panel; see

Figure 4-25. Make sure that you select the **Key**, **Default**, and **Related** options. Click **Finish** to define the new level.

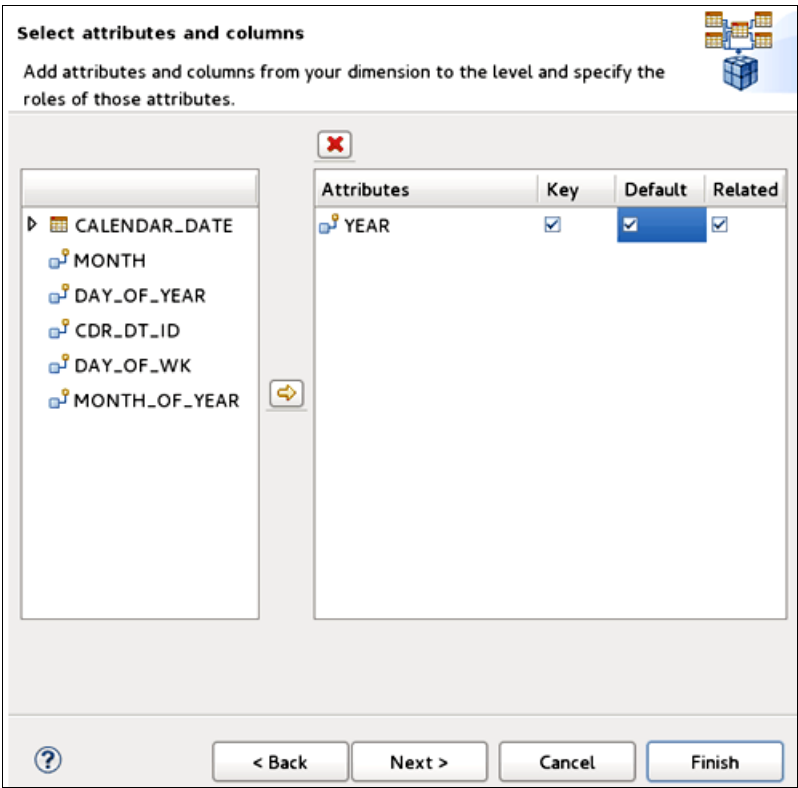


Figure 4-25 Defining the key for the new levels

5. Repeat this process to define new levels for all the new attributes created.

Table 4-1 shows the values to set for the remaining new levels.

Table 4-1 Details of the new levels for Dimension CALENDAR_DATE

Level name	MONTH_OF_YEAR	DAY_OF_YEAR
Level type	Time - Month	Time - Day
Key attributes	MONTH_OF_YEAR, YEAR	DAY_OF_YEAR, YEAR
Default attribute	MONTH	DAY_OF_YEAR

Notice that for these two new levels, we include two attributes to create the unique key, that is, MONTH_OF_YEAR plus YEAR, and DAY_OF YEAR plus YEAR.

Also, to obtain the name of the month name rather than just a number in the report, we add the attribute MONTH as the default attribute for the name of the month.

6. To complete the edits to our CALENDAR_DATE dimension, we add a new level to our hierarchy. Select and right-click the **CALENDAR_DATE** hierarchy to bring up the option menu. Select **Add Data Object** → **Existing Levels**.
7. Within the new panel, select all the new levels just created and click **OK**. These new levels now are added to our hierarchy; see Figure 4-26.



Figure 4-26 Complete defines for Calendar date dimension

We can now test the new model, but instead we continue to make changes to the other dimensions. We define additional attributes and new levels for the dimension INDIVIDUAL_CUST_PROFILE.

8. Using the same procedure, we add the following attributes to the INDIVIDUAL_CUST_PROFILE dimension:
 - Rename CST_ACTV_TNR_RNG to CUSTOMER_TYPE.
 - Rename CST_MKT_SEG to MARKET_SEGMENT.
9. Next, we add attributes as new levels. Select and right-click the Levels directory under the INDIVIDUAL_CUST_PROFILE dimension to bring up the option menu. Select **Add Level** and add the levels listed in Table 4-2.

Table 4-2 Level details for INDIVIDUAL_CUST_PROFILE dimension

Level name	CUSTOMER_TYPE	MARKET_SEGMENT
Level type	Regular	Regular
Key attributes	IDV_CST_PRFL_ID	IDV_CST_PRFL_ID
Default attribute	CUSTOMER_TYPE	MARKET_SEGMENT

10. For the hierarchy, we delete the current IDV_CST_PRFL_ID level and replace it with one of the new levels. Select and right-click **INDIVIDUAL_CUST_PROFILE** under the hierarchy directory. From the

option menu, select **ADD Data Object** → **Existing Levels** and then select the new level **MARKET_SEGMENT**.

Figure 4-27 shows the edited dimension.

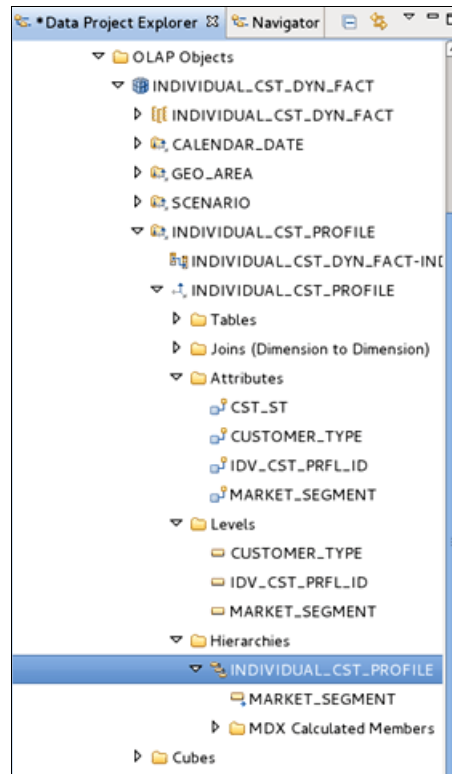


Figure 4-27 Complete edited Customer Profile Dimension

We now have to edit what becomes the final dimension of our cube service. Currently the GEO_AREA dimension is not seen as a valid dimension and was not added to our original default cube service. The reason for this is because in our current model, this dimension does not have any hierarchies defined. Thus, we have to create a hierarchy within the model that can be used within the cube service.

Before we can add a new hierarchy, we have to make sure that we have the right attributes and current levels. Navigate to the GEO_AREA dimension and investigate the current attributes and levels by selecting these items. View their properties within the properties window at the bottom of the Design Studio window. The current attributes do not provide a great deal of information, so we add a new attribute and level.

11. Under the Attributes directory, select **Add Attribute from Columns** from the option menu. Within the new panel that appears, open the items until all the columns within the table are displayed. For our example, we select the REFER_NUM column and rename it to AREA_REFERENCE.
12. Add a new level with the details shown in Table 4-3.

Table 4-3 Level details for AREA_REFERENCE

Level name	AREA_REFERENCE
Level type	Regular
Key attributes	GEO_AREA_ID
Default attribute	AREA_REFERENCE

13. Select and right-click the **Hierarchies** directory within the GEO_AREA dimension to bring up the option menu. Select **Add Hierarchy**. The newly created hierarchy at this point is labeled as “Hierarchy”, which can be renamed to something more meaningful. We named it LOCATION REFERENCE.

14. Finally, using the same process, we add the existing level AREA_REFERENCE to this newly created hierarchy. Figure 4-28 shows the final view.

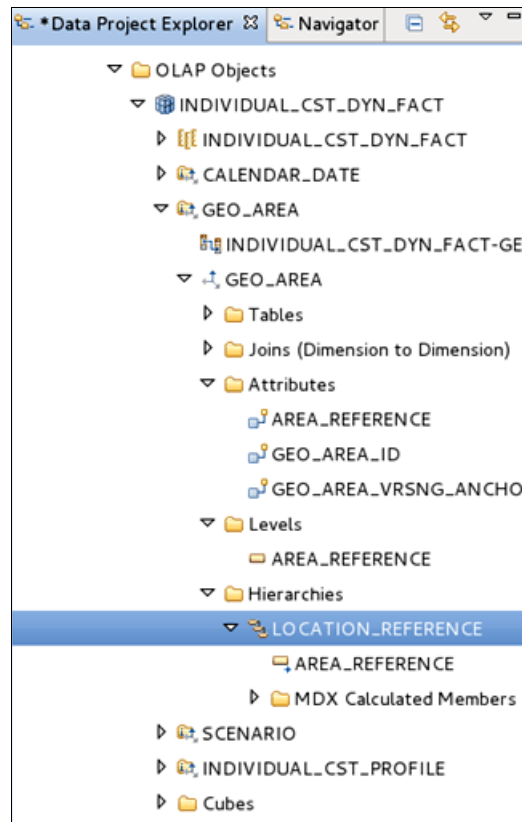


Figure 4-28 Third dimension being defined within our cube model

We have now finished defining the alterations to our cube model. However, the definition of the current basic cube service that was also created during the automatic cube definition does not contain any of the changes that we have made to our cube model. So, we can either alter the definition of our basic cube service, or create a new cube definition.

In the Cubes directory of our cube model, we have the definition of the basic cube service INDIVIDUAL_CST_DYN_FACT. Considering the number of changes we have made to our new cube model, it will be easier to define a new

cube service. This new definition will still be created in the cube folder. Follow these steps:

1. Right-click the Cubes folder to view the option menu. Select **Add Cube**, and give this new definition of a cube service a valid name. In our case, we named it CUSTOMER_PROFILES. We have now created a framework for our cube service definition that at present is empty, without any elements.
2. Select and right-click the cube service **CUSTOMER_PROFILES** to bring up the option menu. Select **Add Data Object** → **Hierarchy**. From the panel that appears, select the **CALENDAR_DATE** hierarchy, then click **Finish**.
3. Repeat the same process for the remaining two hierarchies, LOCATION_REFERENCE and INDIVIDUAL_CST_PROFILE.
4. Now we have to define our fact measures to complete our cube service definition. We start by renaming the current fact section from “Cube Facts (Cube)” to “CUSTOMER_DYN_FACT”. Right-click this new label and select **Add Data Object** → **Cube Measures**. In the opened panel, select the table columns that are required for measures. We select the following columns:
 - NUM_OF_CST
 - NUM_OF_CST_SURVD
 - NUM_OF_SURVD_CST_RSP
 - NUM_OF_CST_CPLN

Figure 4-29 shows the completed cube service definition.

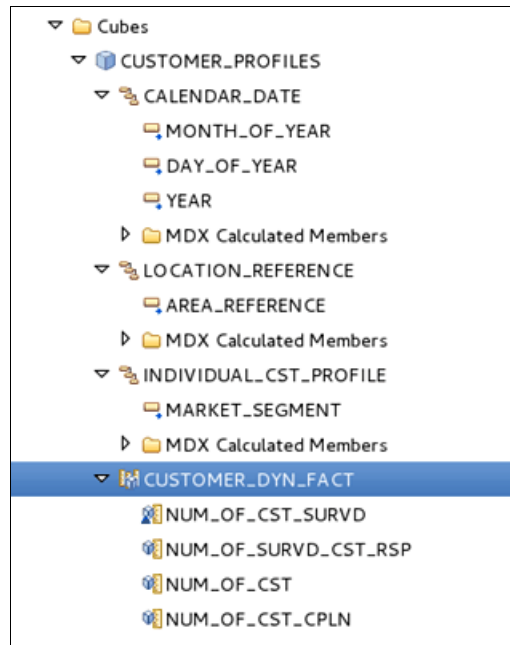


Figure 4-29 New defined cube service

Testing a cube service

Before deploying the new cube service to the InfoSphere Warehouse server application, test the cube service.

You can test the new cube service within Design Studio by following these steps:

1. Right-click the new cube service definition **CUSTOMER_PROFILES** to bring up the option menu. Select **Browse Members**.
2. In the panel, select the data source this cube relates to. In our case, it is IWHPRD82. Click **Finish**.

Design Studio then creates an actual cube service from the cube service definition and starts running this service on the client development platform. In the bottom section of Design Studio, a new tab appears labelled *Metadata View*. This new area displays the new running cube service and all its elements. You can browse and confirm that this running service provides all the required information that your reports will require. If it does not, you can continue to re-edit

your cube model and cube service definition. You can then retest it until the cube service is correct. Figure 4-30 shows what this cube service looks like.

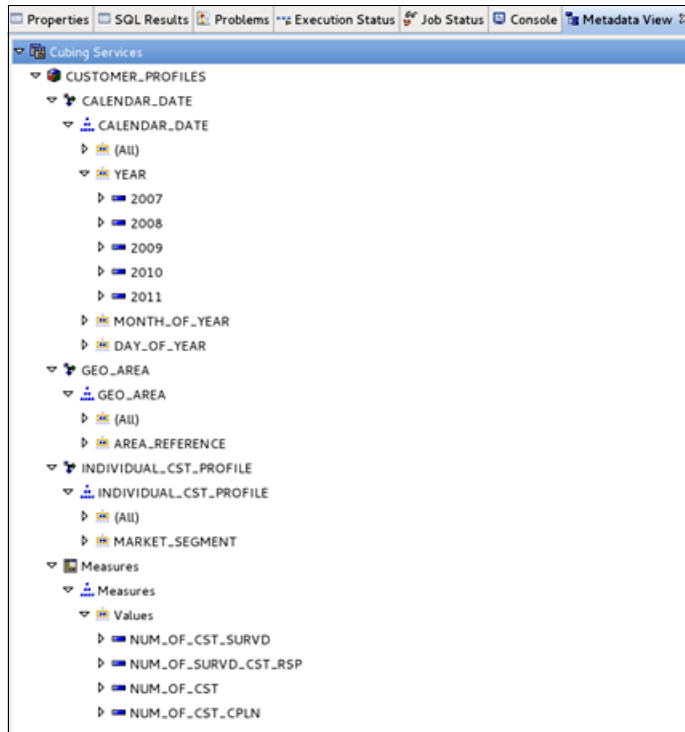


Figure 4-30 Viewing a running cube service in Design Studio

4.4.3 Deploy a cube model to InfoSphere Warehouse as a cube service

After the cube model and the test cube service are defined, you need to deploy this cube service into the runtime element of our InfoSphere Warehouse. The cubing development process has so far been about defining the OLAP objects that portray the cube within Design Studio. To deploy these OLAP objects as a cube service, perform these three main functions:

- ▶ Transfer the OLAP cube definition to the InfoSphere Warehouse metadata repository, and create an export XML file.
- ▶ Define a cube within the InfoSphere Warehouse administration console by loading the XML file.
- ▶ Start and manage the cubing service from within the InfoSphere Warehouse administration console.

Follow these steps to export the cube model within Design Studio:

1. Start Design Studio and open the warehouse project that is used to develop the cube service.
2. Go to **File -> Export** to bring up the Export wizard. Select the **Data Warehouse** folder, then **OLAP Metadata**. Click **Next**.
3. Input the details of a user who has access to the InfoSphere metadata repository. In the next panel (Figure 4-31), specify the OLAP object to be exported and give the export file a name. We use **INDIVIDUAL_CST_DYN_FACT**. Click **Finish**.

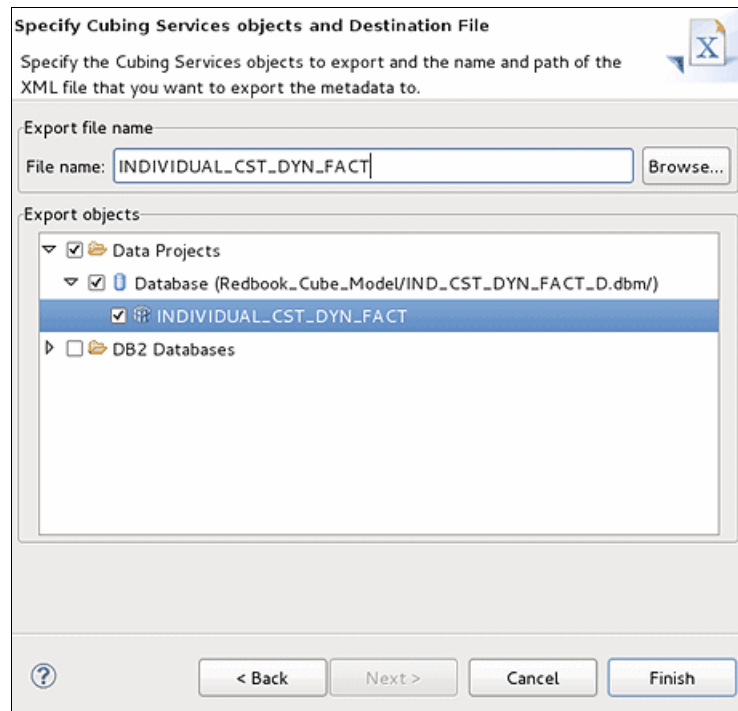


Figure 4-31 Export the cube model to the InfoSphere Administration Console

We have now exported our cube model. The next stage is to import this model into the InfoSphere Data Warehouse Administration Console and create a cube service, as follows:

1. Open the Data Warehouse Administration Console from a web browser. Log in as a valid user. Click **Open** to expand the navigation menu and select the **Cubing Services** tab.
2. To run the cube service, you need a cube server. To create a cube server, click **Manage Cube Server** to be on the right page. Next, click the create

button (🔹), which is shown as a small yellow diamond on the tools menu. A pop-up panel appears. Enter a name for the cube server you are creating, the host name on which the cube server will run, and a port number, which is 9214. For the next few panels, keep the default options by clicking **Next** until the end. Then click **Finish** to complete the creation process.

3. To start the new cube server, highlight the newly created cube server and then click the green play icon (▶) as shown in Figure 4-32.

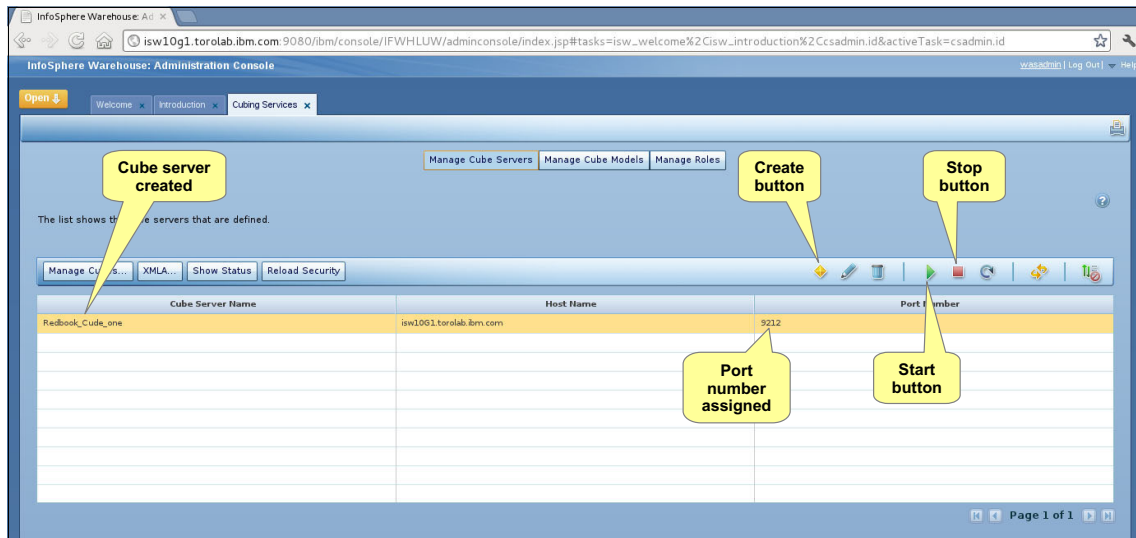


Figure 4-32 Define a cube sever in the InfoSphere Data Warehouse Administrator Console

4. Import the cube service into this cube server. Select **Manage Cube Models** to open a new panel, then select **Import Cube Model**. A wizard panel opens asking for the data source of the base data. In our case, this is IWHPRD82. Click **Next**.
5. Select the XML file created within Design Studio. Our file is `INDIVIDUAL_CST_DYN_FACT.xml`. Click **Next**. The input file is read and a

list of the included cube models and associated cubes is displayed. See Figure 4-33.

Click **Next**, then **Finish** to import the cube model. The new cube model now appears in the list.

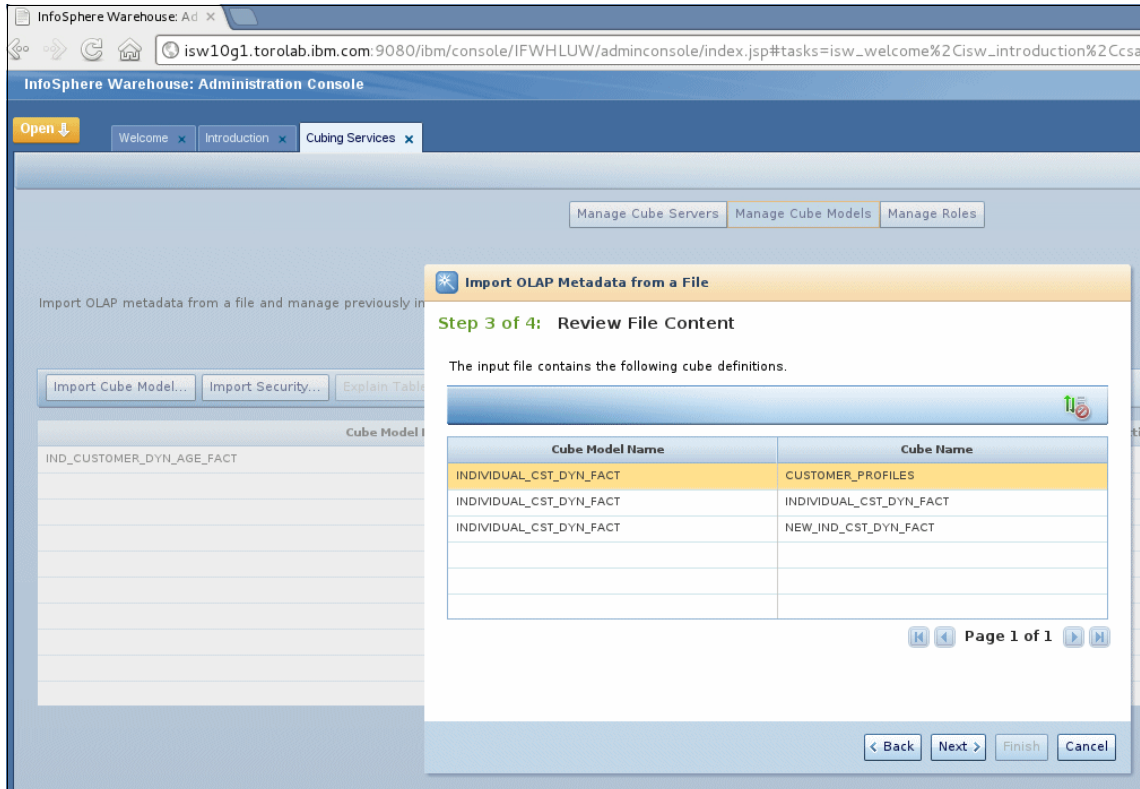


Figure 4-33 Importing the cube models into the Administration Console

6. Load the cube model into the cube server and start the new cube service. Start by returning to the “Manage Cube Servers” panel and selecting the defined cube server. Then select **Manage Cubes**. A new wizard appears.
7. From the first panel of this wizard, select the green plus (+) icon to load into this server a cube service from the loaded models; see Figure 4-34 on

page 138. Select the required cube service, in our example **CUSTOMER_PROFILES**, then click **OK**.

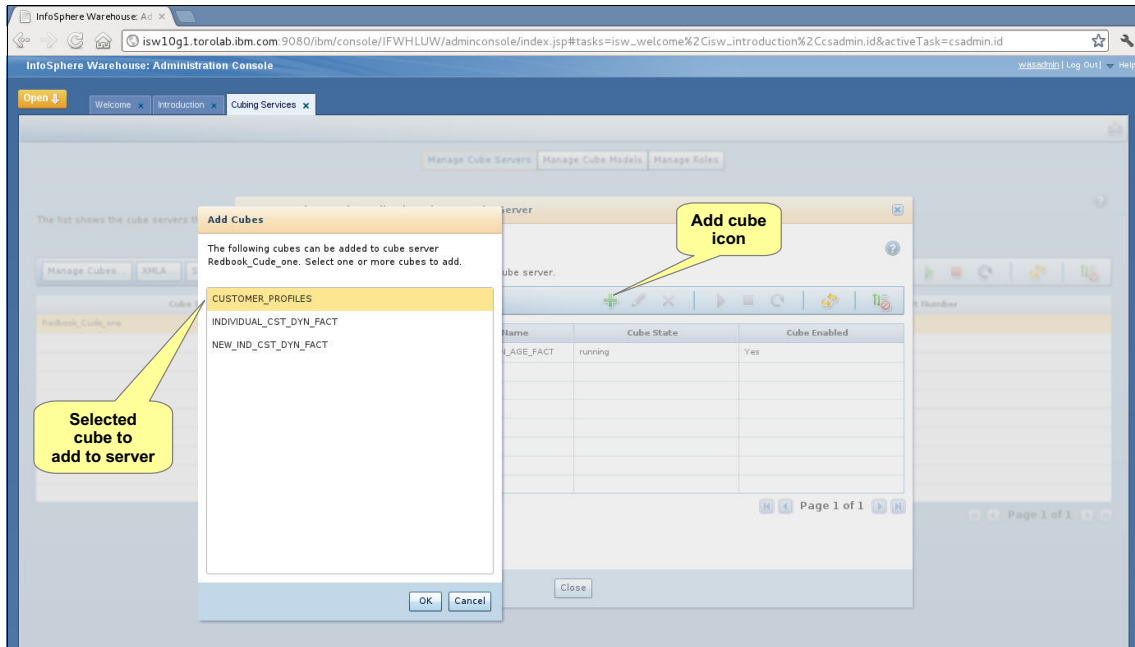


Figure 4-34 Cube service added to cube server in the Administration Console

- When returned to the wizard panel, the new loaded cube service is stopped initially. Select the cube service then click the green play icon to start this cube service within the cube server. The status will then change to running.

We now have our running cube service within our cube server on the InfoSphere Warehouse server. All we now need is an application to make use of this OLAP cube service.

4.5 Modeling and IBM Cognos BI

In the preceding sections of this chapter we show how to create and develop a range of models, from the dimensional model that represents certain business practices to the physical data model which represents the database structure, and finally to an OLAP model that allows you to provide faster answers to your business questions. However, this is not the final step in the modeling process.

In the past, only a small percentage of highly skilled employees within the business had the understanding and knowledge to work with these OLAP

models and resources. They often needed to manipulate the models or the underlying data directly to provide reports. This meant those within the business who needed to make decisions had to rely on a limited set of static reports, which were never really able to answer all their questions.

With the introduction of reporting tools such as IBM Cognos BI, these decision makers can now access dynamic reports directly and, more importantly, the underlying data in views that they can both understand and manipulate. The ability to view these reports at any location and on a multitude of devices, while still containing the latest information and data, finally allows these decision makers to make valid and informed choices.

In this section we demonstrate how you can integrate the modeling processes implemented within InfoSphere Warehouse with those contained within IBM Cognos BI, showing how you can complete the modeling lifecycle and feed back improvements to your logical, physical, and OLAP models in the process.

It is worth noting that this section of the book is not intended to cover the entire IBM Cognos BI product range, or to completely cover all aspects of producing a Cognos report. The Redbooks publication *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912, covers these topics in greater detail.

4.5.1 Cognos Framework Manager

IBM Cognos Framework Manager is a metadata modeling tool for managing access to a range of data sources and creating a common business model. These data sources can include InfoSphere Warehouse cubing services, DB2 databases, and additional data sources which are not covered in this book.

Figure 4-35 outlines the process within IBM Cognos Framework Manager of creating a Cognos package that represents a business model.

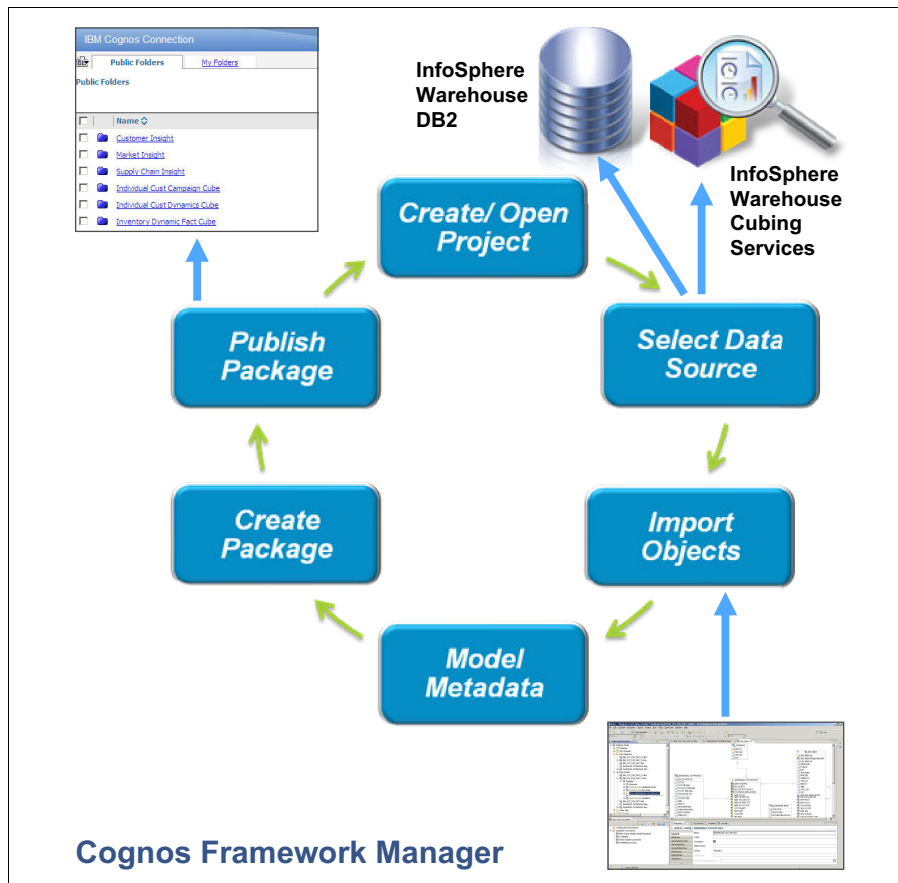


Figure 4-35 IBM Cognos Frameworks Manager

IBM Cognos BI note: InfoSphere Advanced Enterprise Edition includes a number of licenses for IBM Cognos BI. These include one administration and a number of consumer entitlements. These copies are restricted to web support only and cover the following elements of IBM Cognos BI:

- ▶ Query Studio
- ▶ Reporting Studio
- ▶ Analysis Studio
- ▶ Dashboards
- ▶ Event Management
- ▶ Framework Manager

The installation of IBM Cognos BI is separate from that for InfoSphere Warehouse, but can be installed on the same hardware server.

At the time of writing, IBM Cognos Framework Manager is available as a Microsoft Windows application only. The server components of IBM Cognos BI are available for a range of hardware and operating system platforms.

4.5.2 Create a Cognos model from cubing services

In this section, we demonstrate how to create and publish a Cognos model from an existing cubing service. This particular example follows on from 4.4.3, “Deploy a cube model to InfoSphere Warehouse as a cube service” on page 134, in that it uses our currently deployed cubing service as the starting foundation.

Follow these steps to create and publish a Cognos model from an existing cubing service:

1. Make a note of the port number for the XMLA service of the published cube service. Open the InfoSphere Administration Console with a web browser and navigate to the Manage Cube Servers section. With the chosen cube server

highlighted, select **XMLA** to view details about the XMLA services, as shown in Figure 4-36. Make a note of the network port assigned to the XMLA service.

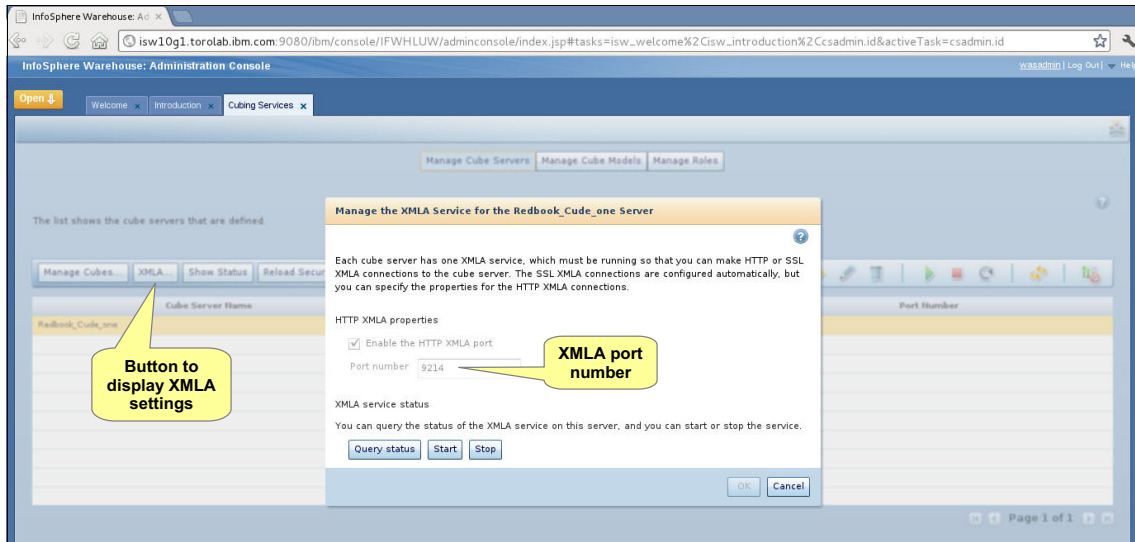


Figure 4-36 Setting for XMLA Port of a cube server

2. Check that the IBM Cognos BI server is running, because this is required to publish the model.

The start process for the Cognos BI server is particular to each operating system platform. Check the Cognos BI manual for information about starting the server.

For Microsoft Windows, as an example, the process is to select **All Programs** → **IBM Cognos 10** → **IBM Cognos Configuration**, then select the play button to start the server.

For Linux, to start the Cognos BI server, from the Cognos installed directory, run the following script:

```
./cogconfig.sh -s
```

3. On a Microsoft Windows Client, start the Cognos Framework Manager:
All Programs → **IBM Cognos 10** → **Framework Manager**

4. Create a new project. Under Projects (Figure 4-37), select the link to create a new project and give the project a name. We use **Redbooks Example**. Then select **English** as the language.

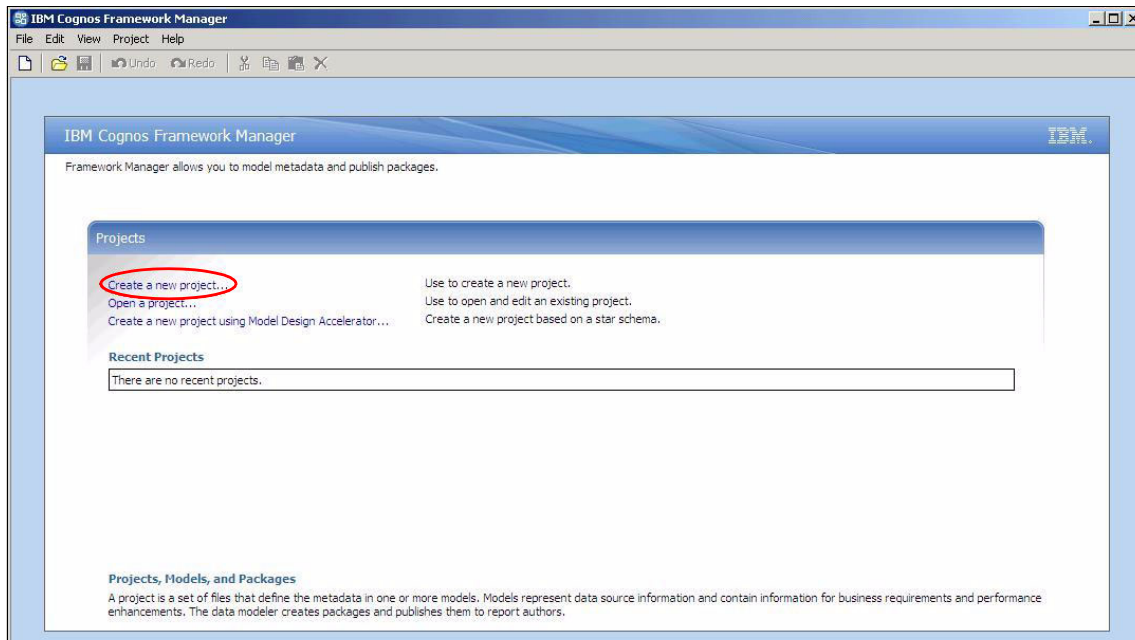


Figure 4-37 Create a new project within IBM Cognos Framework Manager

5. Within the panel that opens next, select the **Data Sources** option and click **Next**. Then select **New** to create a new data source.
6. Give the Data Source a name, for example, **ISW_Repository**, then click **Next**.

For the data source type, select **IBM InfoSphere Warehouse cubing service (XMLA)**. Keep the default isolation level, then click **Next**. The next set of questions are about the URL of the cubing server XMLA provider. This URL is composed of hostname:port/Service_name, with the port number being the value that is noted in step 1. The service name is “IBMXmlAnalysis” so that the complete URL in our example might appear as follows:

isw10g1:9214/IBMXmlAnalysis

A user name and password might be required if security has been enabled within the cube model.

Use **Test the connection** to test the connection to the cube with the details entered. See Figure 4-38 on page 144.

Having tested the connection, click **Finish** to save this data source within the Cognos metadata database.

New data source

Specify the IBM InfoSphere Warehouse cubing services (XMLE) connection string - New Data Source

Edit the parameters to build an IBM InfoSphere Warehouse cubing services (XMLE) connection string.

Server URL:
isw10g1:9214/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 signon.

☐ User ID

☐ Password

☒ Create a signon that the Everyone group can use:

User ID:
Password:
Confirm password:

Testing

Test the connection...

Figure 4-38 Create a new data source within IBM Cognos Framework Manager

7. The tool returns to the Data Source selection panel with the newly created data source "ISW_Registry" in the list. To import the cube model in to the Cognos Framework Manager, select this new data source and click **Next**.

The tool presents a list of the cubing service models as shown in Figure 4-39. We select our model **CUSTOMER_PROFILE** and click **Next**.

In the next panel, select the **Create a default package** option and then complete the import by clicking **Finish**.

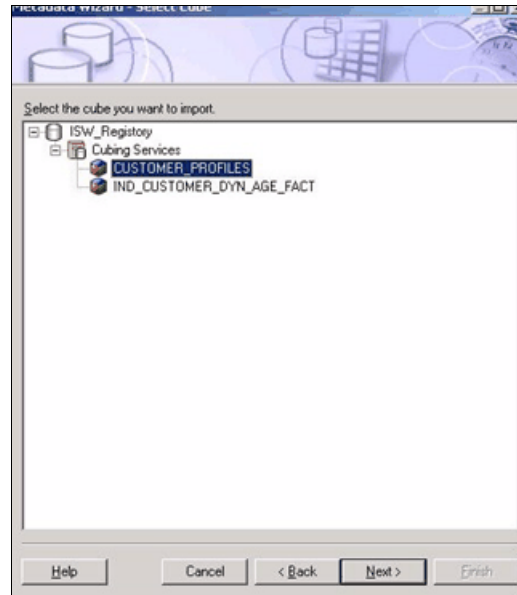


Figure 4-39 Import cube models into IBM Cognos Framework Manager

8. Publish the package by opening the **Publish Package Wizard**, then selecting **Yes**. In the next few panels, accept the default options by clicking **Next** until the final panel and then click **Publish**. The tool then publishes the package to

the Cognos BI server. Figure 4-40 shows the Cognos Frameworks Manager panel with our package.

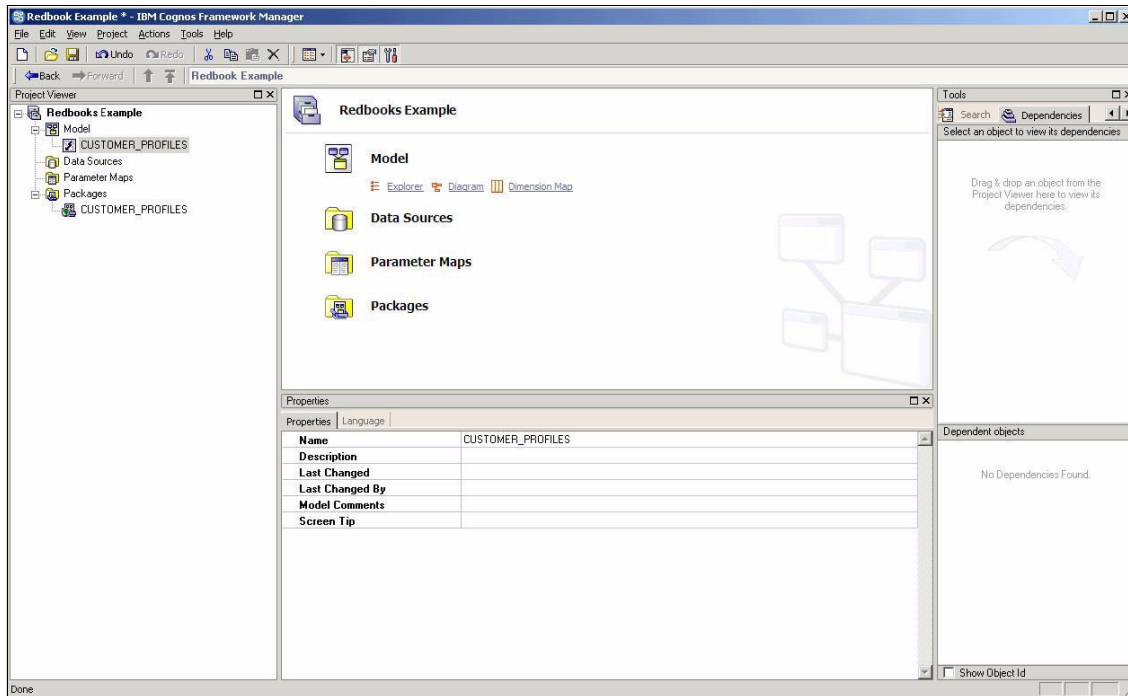


Figure 4-40 Cognos Frameworks with our new package

In Cognos terms, we have now successfully published our package to the Cognos BI server. But to make meaningful use of this package, we have to create a Cognos report that displays the information within this package, and thus within our cube service in a more user-friendly context. To achieve this goal, we use the Report Studio of Cognos BI as follows:

1. Open a Microsoft Internet Explorer browser. Enter the URL `http://<hostname of the Cognos server>/ibmcognos`, for example:

`http://isw10g1/ibmcognos`

The home window of Cognos BI will display as shown in Figure 4-41 on page 147.

Browser note: Microsoft Internet Explorer is the only browser that supports the full range of the Cognos tool set, in this case, Report Studio.

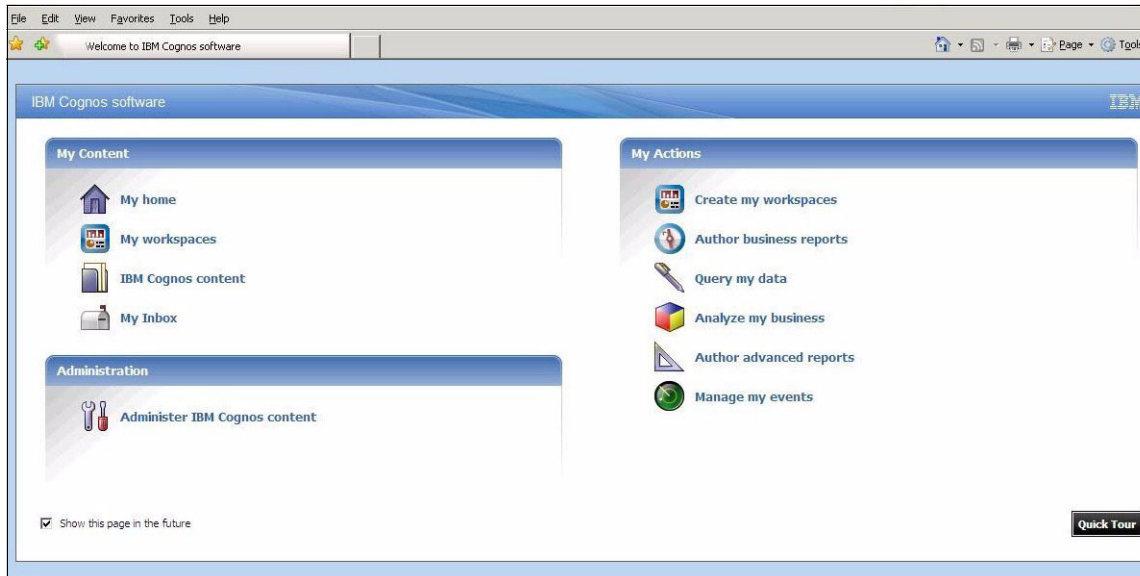


Figure 4-41 Home panel of IBM Cognos BI

2. Before starting to create a report, we check our published package from the browser. From the Welcome window, select **My Home**. This displays the view that lists the packages loaded into the IBM Cognos BI server. See Figure 4-42. Notice that currently there is nothing under our package area.

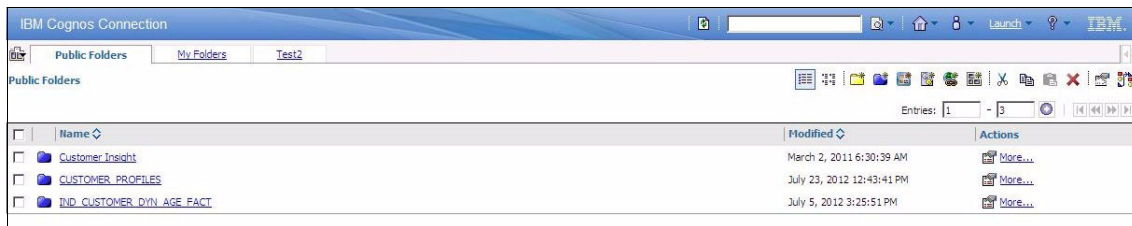


Figure 4-42 Packages loaded into the IBM Cognos BI Server

-
- IBM Cognos Connection
- Public Folders
- My Folders
- Test2
- Public Folders
- Customer Insight
- CUSTOMER_PROFILES
- IND_CUSTOMER_DYN_AGE_FACT
- Report Studio
- Launch

Figure 4-43 Create a new report in IBM Cognos

-
- The screenshot displays the IBM Cognos Report Studio interface. The top menu bar includes File, Edit, View, Structure, Table, Data, Run, Tools, and Help. Below the menu is a toolbar with various icons for report manipulation. The main workspace shows a report preview with a blue and white background. On the left, the text "IBM Cognos Report Studio" is displayed. In the center, there is a bar chart showing data for years 1999, 2000, 2001, 2002, and 2003. To the right of the chart are two data tables. The first table, titled "Sales by Product", shows sales data for various products. The second table, titled "Sales by Region", shows sales data for different regions. At the bottom left, there are two buttons: "Create new" and "Open existing". A checkbox labeled "Show this dialog in the future" is checked. A "Close" button is located at the bottom right of the dialog.

Figure 4-44 IBM Cognos BI Report Studio application

- 148 Solving Operational Business Intelligence with InfoSphere Warehouse Advanced Edition

Chart - Column option. Figure 4-45 shows the panel displayed, with our notes about the functions in this panel.

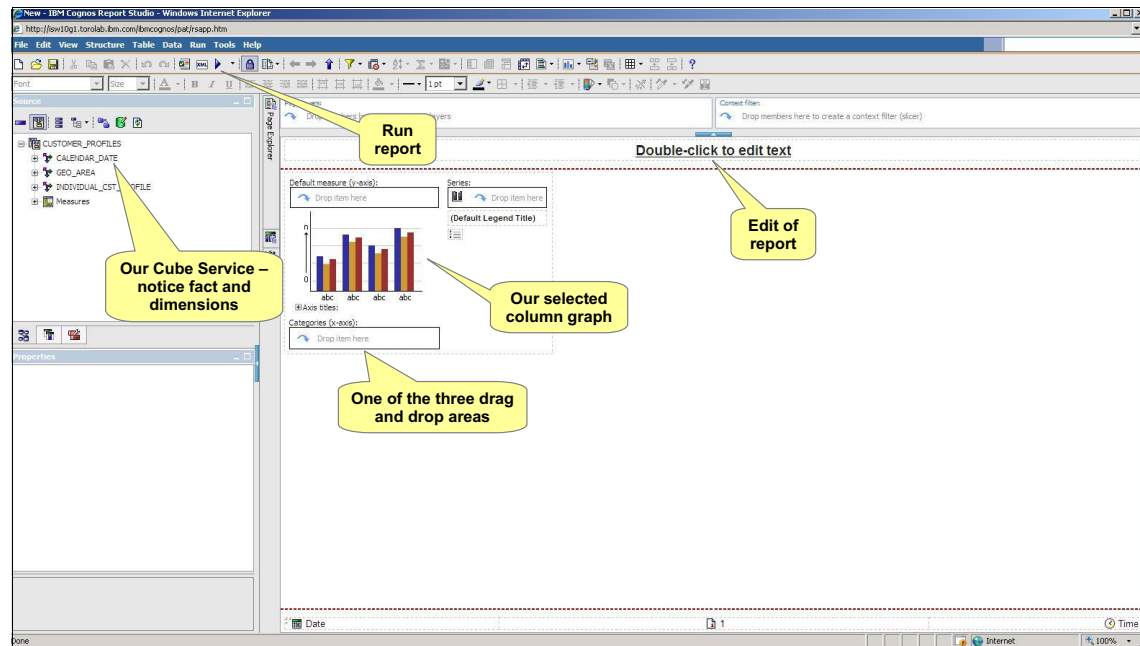


Figure 4-45 Create a new IBM Cognos report

7. Navigate into the **CALENDAR_DATE** dimension on the left side. Here you can drag and drop the elements into the panes to create your chart. In our example, we want year level as the X axis so we select the **Year** level under the hierarchy, then drag and drop it onto the center graph pane.

8. Next, we drag and drop the measures NUM_OF_CST, NUM_OF_CST_SURVD, and NUM_OF_SURVD_CST into the Series area, which produces a panel as shown in Figure 4-46.

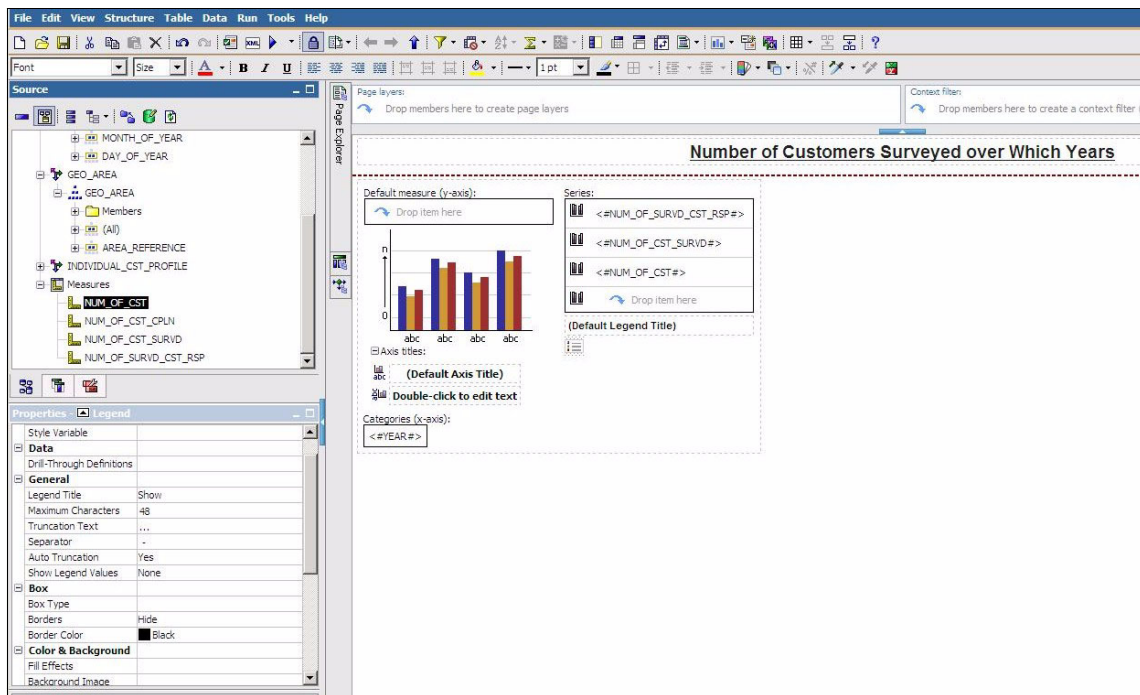


Figure 4-46 IBM Cognos report being created

9. Now we “run” this report to gather the data and produce a completed graph by clicking the blue triangle play icon on the tool bar, shown at the top in

Figure 4-45 on page 149. The Report Studio opens a new page and displays the completed report; see Figure 4-47.

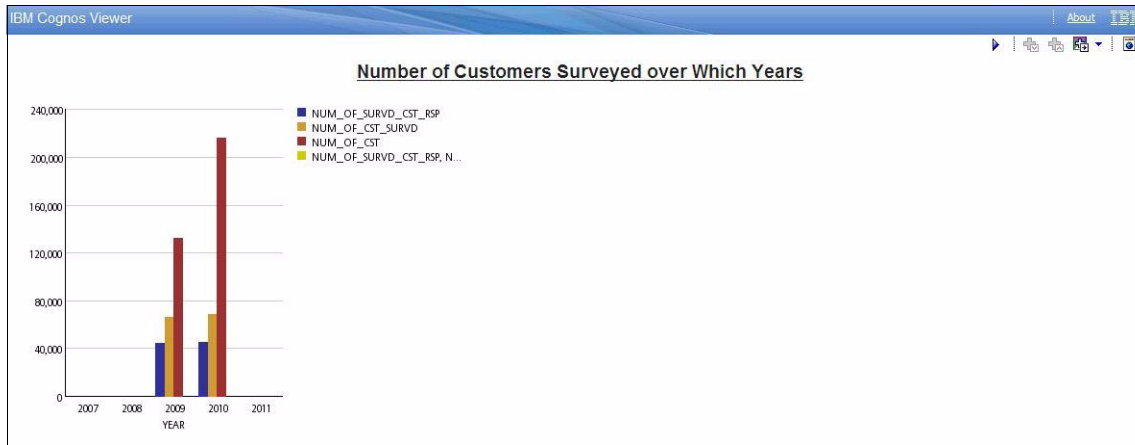


Figure 4-47 Report created

This report layout can now be saved and used later to produce this report from “My Home” of IBM Cognos BI.

4.5.3 Create a Cognos model from a logical dimensional model within Data Architect

After installing the cube model as a package into IBM Cognos BI and creating an example simple report as demonstrated in the previous section, you might notice that the model within IBM Cognos Frameworks is limited as to the information it contains. There are no dimensions, facts, attributes, levels, or measures shown within the model. This is due to the method by which we loaded this model into IBM Cognos Framework.

Our next example, however, demonstrates an alternative approach about how to transfer the model from IBM InfoSphere Data Architect into Cognos Framework, and retain the information we expect to see within our model.

We start with the dimensional model we created in 4.3.3, “Add dimensional notation to the logical data model” on page 104 and then perform the following steps to export this dimensional model out of IBM InfoSphere Data Architect:

1. Start IBM Data Architect, then open the project and navigate to find the logical dimensional model file. In our case, it is IND_CST_DYN_FACT_D.Idm. Select and right-click this file to bring up the option menu. Select the **Export** option.

2. From the Select panel, select **Data Model Export Wizard** from the Data folder as shown in Figure 4-48.

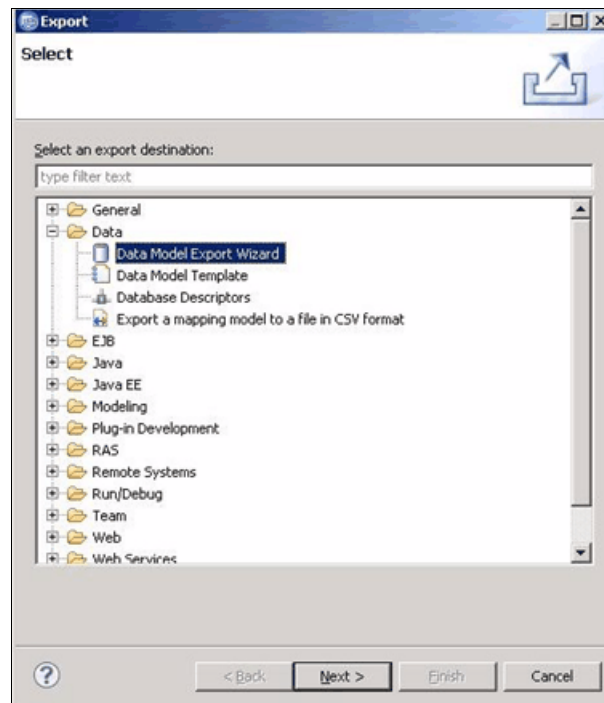


Figure 4-48 Data Model Export Wizard menu within InfoSphere Data Architect

3. Select **IBM Cognos BI Reporting - Framework Manager** as the Model Format. For Model, click **Browse** and navigate to the required model, for example, IND_CST_DYN_FACT_D.Idm. Click **OK**.

For Export to field, navigate to a suitable location for the export file. Click **Next**. See Figure 4-49.

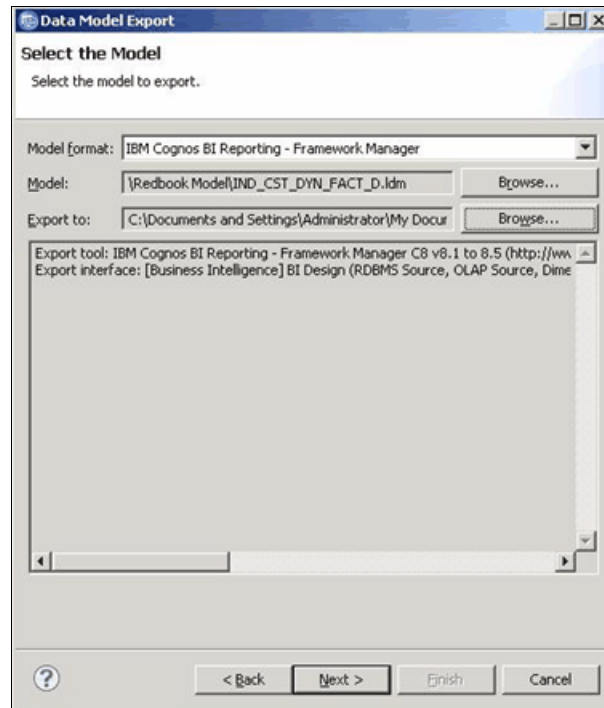


Figure 4-49 Select the Model panel of the Data Model export Wizard within IBM InfoSphere Data Architect

4. On the next panel, accept the defaults and click **Next**, and finally **Finish**.

We have now exported our logical dimensional model out of IBM InfoSphere Data Architect. Next, we import this model into IBM Cognos BI Framework Manager.

1. Start IBM Cognos BI Framework Manager to the initial home page as shown in Figure 4-37 on page 143. Select **Create a New Project** and give the new project a name. We use **Redbooks Example2**. Click **OK**, then **English**.
2. Within the wizard, select the **IBM Cognos Model** for Data Source, then click **Next**.

3. Select the file created within IBM InfoSphere Data Architect, and then click **Next**.
4. The next panel displays the model contained within the export file. For our example, we import both the logical and physical model. Select the models as shown in Figure 4-50, then click **Next**.

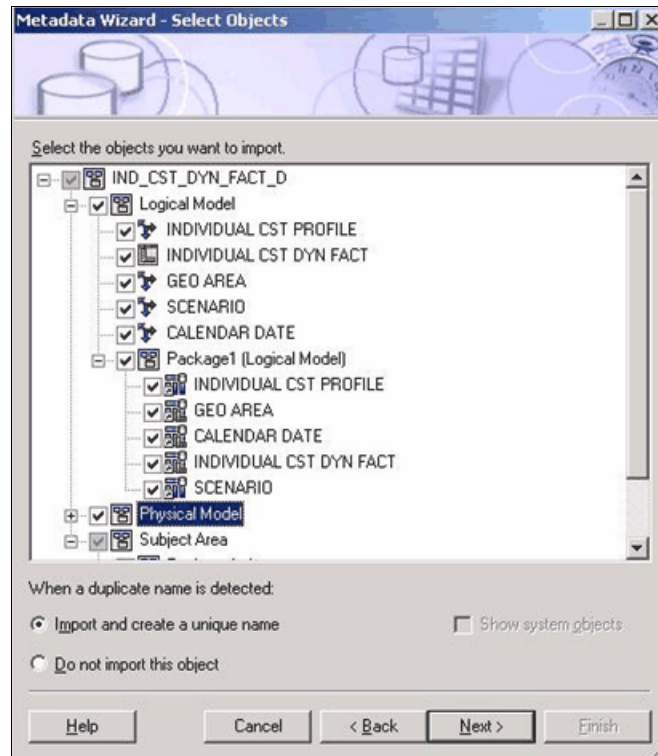


Figure 4-50 Import logical and physical models with the wizard within IBM Cognos BI Framework Manager

5. The final panel gives information about the model being imported. There is also an option within this final panel to have the model verified after import. Click **Final** to complete the import process.

Figure 4-51 on page 155 shows that the model imported here has more information about its dimensions, fact, attributes, and measures than the first model that was imported. This model can also be published to the IBM Cognos BI server in the same manner as the model in the first project through the contained package. This model can also be further developed within IBM

Cognos Frameworks Manager as the model was in both IBM InfoSphere Data Architect and IBM Design Studio.

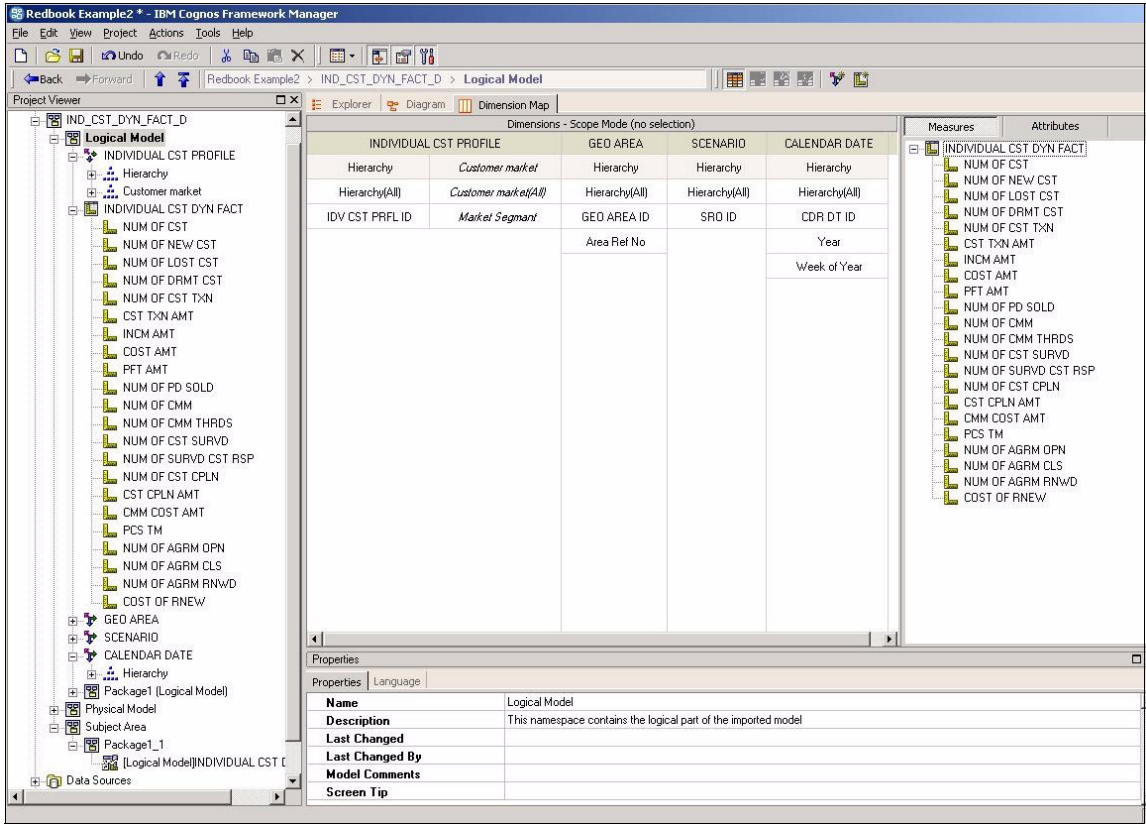


Figure 4-51 Dimensional logical and physical models from IBM InfoSphere Data Architect, loaded into IBM Cognos Frameworks manager example

There are a number of different process that can be used to transfer model information among these three client applications. Our example only demonstrated two methods, but a third possible process is for the Cognos Framework Client to import the logical and physical data models in their native 1dm and dbm format.

For further information and examples of developing and working with these models within IBM Cognos Frameworks Manager, see *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

4.6 Introduction to InfoSphere Warehouse Packs

An important element of InfoSphere Warehouse Advanced Edition is the inclusion of the InfoSphere Warehouse Insight Packs. These packs provide customers with an immediate saving in both cost and time because they provide the necessary foundation to build a data warehouse and business analytics solution. These Insight Packs are adaptable and provide a solution aimed at organizations across multiple industries.

Managers and CEOs are not interested in database structures, but rather in questions relating to their business, such as “Which customers are most profitable for us?” and “What is my market share?”. These might seem simple questions, but they require complex business analytics to provide the required answers.

For many small and medium-size companies that might not have the necessary skilled staff, time, or budget, developing a complex solution from scratch is challenging. At the same time, they cannot afford not to ask these questions or not have the right tools available to provide the answers. These companies need an affordable method to accelerate the development process and help with implementing a successful solution. It is for this reason that IBM developed these Insight packs.

Customers utilizing the Insight Packs benefit from more than 15 years of IBM experience in developing data warehouse models, along with years of successful client engagements across multiple industry sectors with more than 300 clients.

These packs are accelerator add-ons for InfoSphere Warehouse, complete with a set of prebuilt, consolidated data structures, and supporting components in the areas of:

- ▶ Customer Insight
- ▶ Market and Campaign Insight
- ▶ Supply Chain Insight

The core of these packs is a set of physical data models that can be applied to a DB2 relational database. Additionally, they include logical, OLAP, and sample cube assets along with Cognos reporting structures.

Three InfoSphere Warehouse Packs are currently available:

- ▶ IBM InfoSphere Warehouse Pack for Customer Insight

This pack addresses a range of business issues common to organizations that are trying to better understand the profitability and profiles of their typical customers. The pack includes sample Cognos reports in the areas of

communications analysis, transaction analysis, periodic analysis, and profitability analysis.

- ▶ IBM InfoSphere Warehouse Pack for Market and Campaign Insight

This pack is designed to help organizations understand the overall details of their market and determine how to acquire new customers in a competitive environment. The pack includes sample Cognos reports in the areas of market analysis and campaign analysis.

- ▶ IBM InfoSphere Warehouse Pack for Supply Chain Insight

This pack is designed to help organizations ensure they have the correct products or components in the right place at the right time, without negatively impacting cash flow by overinvesting in inventory. For many organizations, efficient supply chain management is fundamental to their success.

Figure 4-52 shows the InfoSphere Warehouse Insight Packs.

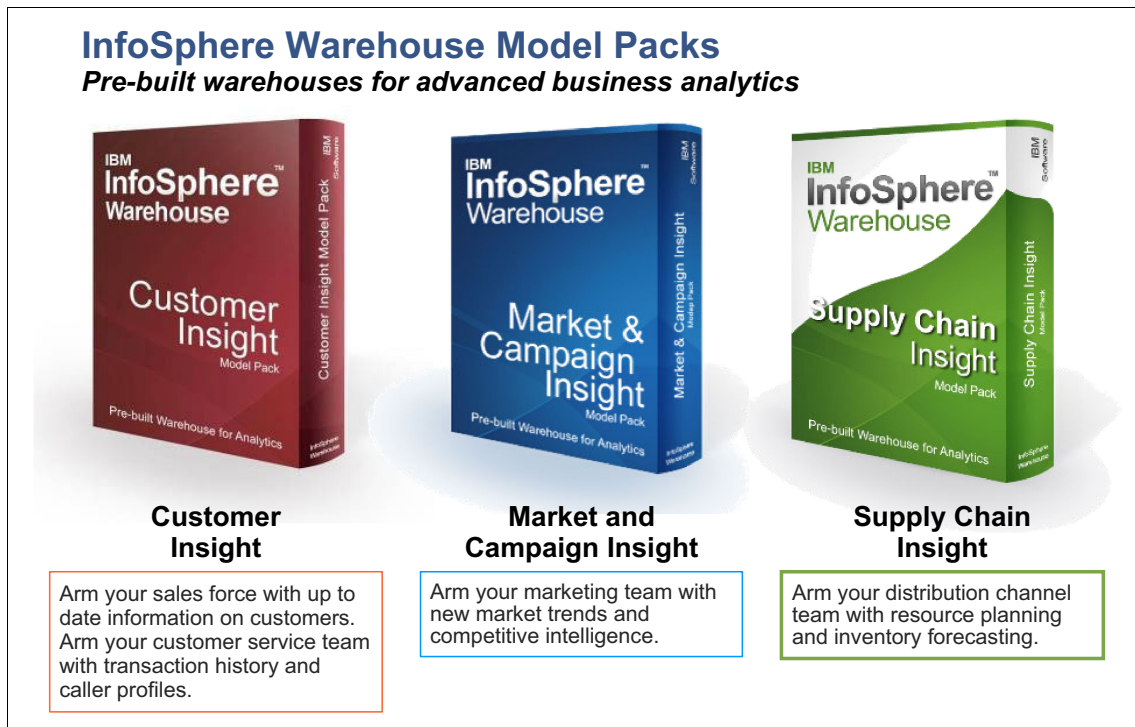


Figure 4-52 The three InfoSphere Warehouse Insight Packs

These three IBM InfoSphere Warehouse Packs consist of the following elements:

- ▶ A set of physical data models defining the necessary conformed dimensional warehouse and mart structures needed to support customer, market and campaign, and supply chain analysis. These data models are delivered as a set of .dbm files compatible for use with both InfoSphere Warehouse Design Studio and InfoSphere Data Architect.
- ▶ A set of sample physical database structures specific for an InfoSphere Warehouse environment. These sample structures consist of a dimensional warehouse and mart structures specific to the business issue being addressed. These sample physical database definitions are delivered as a set of .ddl files.
- ▶ A set of sample Cognos reports enabling visualization of how the business issues are addressed.
- ▶ A set of documentation providing guidance and assistance on the deployment and customization of the offerings. This is delivered as .pdf files.

A user might implement a single pack as a starting point and then deploy future packs to the same central data warehouse. The included warehouse data model infrastructure supports the deployment of multiple packs. This enables the growth of a single data warehouse infrastructure and simplifies the implementation and expansion of the warehouse.

Each InfoSphere Warehouse Pack contains the staging, data warehouse, and datamart definitions required for specific business issues. Instead of developing custom data models, organizations can use the industrial-strength data models of the packs. Prebuilt physical data models promote data integrity from the loading stage through the reporting stage. The prebuilt reports illustrate immediate representation of the information of an organization. The physical data model and reports can be used as is, or customized to suit the organization.

Figure 4-53 illustrates the tables included in the Staging Area model, Data Warehouse layer, and the Data Mart layer of the Customer Insight Warehouse Pack.

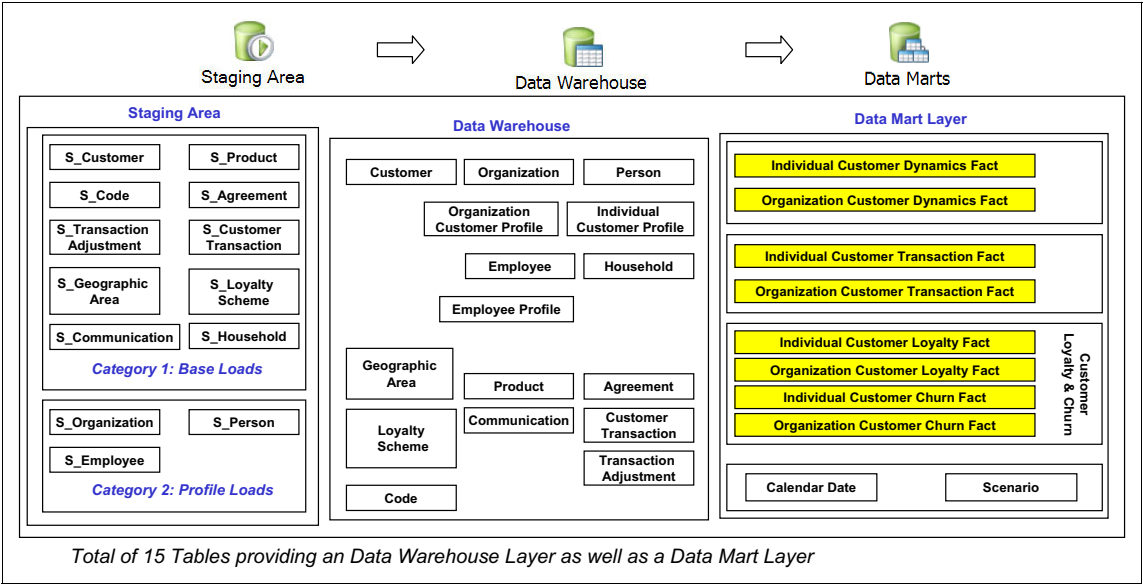


Figure 4-53 Table structure of the Customer Insight Warehouse Pack

Figure 4-54 illustrates the tables included in the Staging Area model, Atomic Data Warehouse layer, and the Data Mart layer of the Market and Campaign Insight Warehouse Pack.

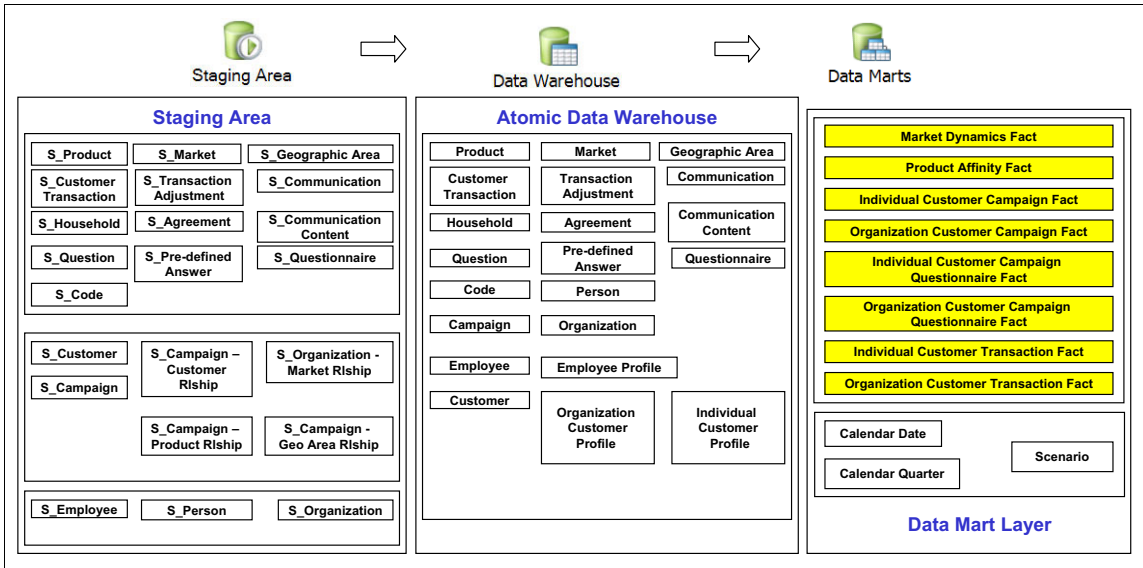


Figure 4-54 Table structure of the Market and Campaign Insight Warehouse Pack

Figure 4-55 illustrates the tables included in the Staging Area model, Atomic Data Warehouse layer and the Data Mart layer of the Supply Chain Insight Warehouse Pack.

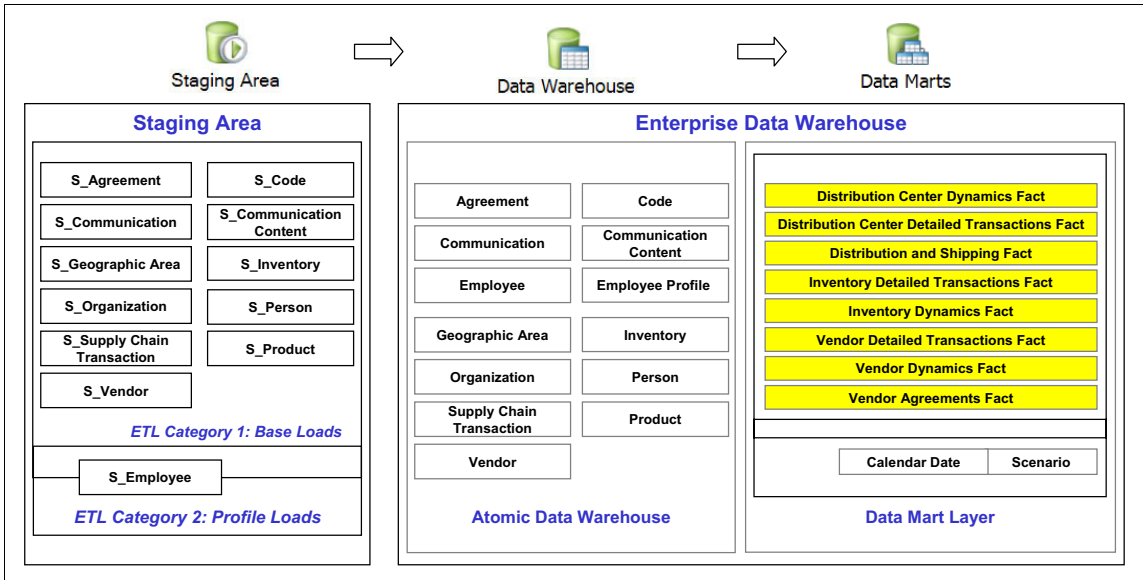


Figure 4-55 Table structure of the Supply Chain Insight Warehouse Pack

For all three packs, all tables within the models are fully attributed and all objects such as tables and columns are fully documented with definitions in the PDF files.

Figure 4-56 on page 162 shows an example of the Individual Customer Transaction Star Schema, which is one of two included Star Schemas measuring Customer Transactions. Notice the tables that make up the fact table and those that make up the dimensions. The measures in the fact table hold information

about each customer transaction that include the number of transactions, transaction amount, cost, and profit margin percentage.

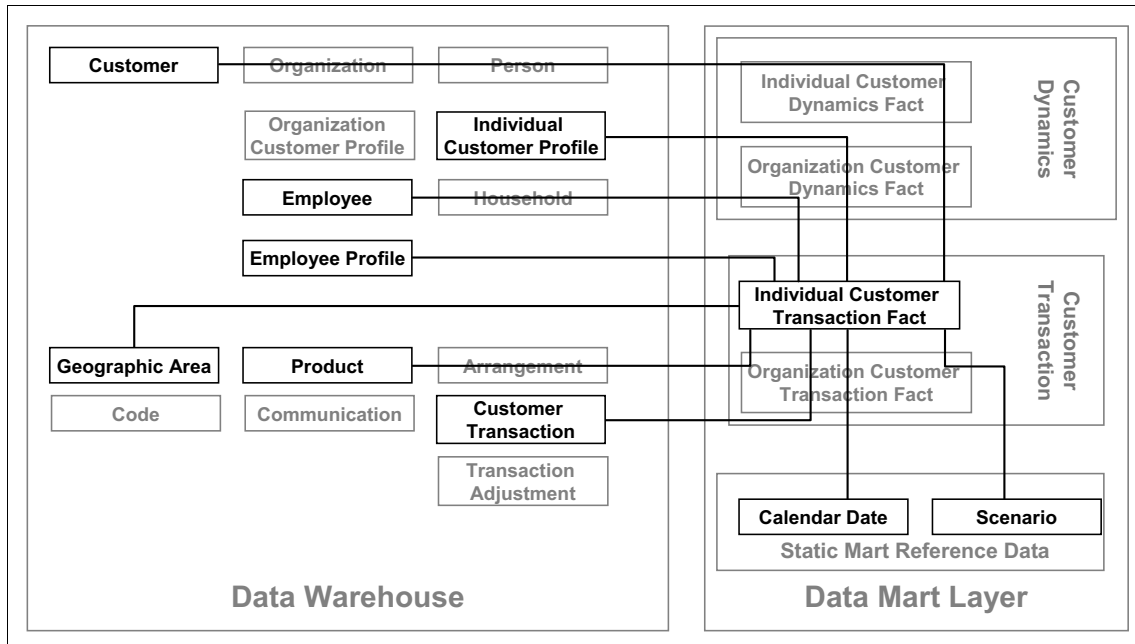


Figure 4-56 One of two Customer Transaction star schemas included in the packs

A sample OLAP cube model is provided with each pack, based on the measures and dimensions in the star schema. Figure 4-57 on page 163 shows how it looks in the Design Studio. The provided model can be easily deployed to cubing

services, and then the cubes can be used by Cognos Analysis Studio to access the data.

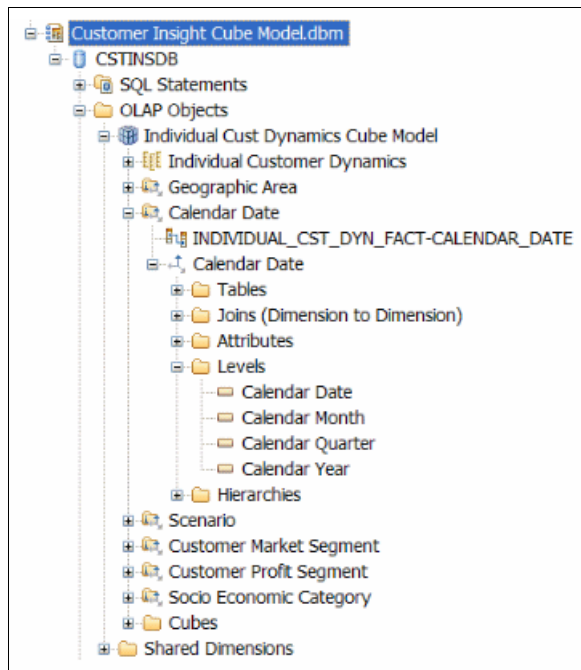


Figure 4-57 Sample OLAP cube model included in the packs

The InfoSphere Warehouse pack for Customer Insight also includes sample Cognos reports in the areas of customer loyalty, customer profiling, and customer profitability:

- Customer profiling -communications analysis

These reports help organizations understand how their customers communicate with them and why. Communications analysis reports can tell organizations which types of customers communicate with the greatest frequency, positively and negatively. Organizations can analyze reasons for complaints or positive comments, the products responsible, and the associated costs. Specific reports include activity and complaints by individual or by organization, survey responses by age group, and survey responses by socioeconomic category.

- Customer profiling - transaction analysis

Organizations can use this series of reports to analyze customer transaction activity across socioeconomic groups, age groups and channels. Specific report titles include product sales analysis, profit by socioeconomic category

and product type, top performers by income, top performers by volume, volume sales by channel and volume sales by customer age group.

- ▶ Customer profitability - profitability analysis

Using this series of reports, an organization can explore customer profitability in a number of commonly used dimensions. The reports cover income, profitability and introduce the use of budgets and actual values. Reports included in this category are average income and profit, customer profitability for geographic regions and countries, income and profit - actual versus budget by year, renewals analysis by individual, and renewals analysis by organization.

- ▶ Customer profitability - periodic analysis

Reports in this category help organizations examine customer profitability over time. Designed to measure both income and profit, the reports incorporate the following comparisons for individuals and organizations: current quarter this year versus same quarter last year, this year to date versus last year to date and year-on-year profitability comparison, along with user-defined flexible-period comparisons.

The InfoSphere Warehouse pack for Market and Campaign Insight includes sample Cognos reports in the areas of market analysis and campaign analysis:

- ▶ Campaign analysis

Organizations regularly need to answer questions such as which sales campaigns are cost-effective and whether in-store or in-branch advertising is generating leads. Measuring effectiveness means comparing actual performance in a defined time period against the targeted performance. This series includes reports on sales campaign volume performance, campaign cost analysis, and lead source analysis.

- ▶ Market analysis

To be competitive, every organization must understand its market and establish whether market share is rising or falling. The organization will want to further refine this analysis to understand if the trend is across market segments or particular to a group of customers, a geographic location or a product. Market analysis can also yield insights such as how performance in one market segment is impacting another. Reports in this category include market volume trend analysis, specific period and cumulative analysis, and like-for-like analysis.

- ▶ Market-basket analysis

The market-basket modeling technique is used to help increase sales by determining whether customers who buy a certain group of items are likely to buy another group of associated items. To ensure effectiveness, the organization must answer questions such as which product group

combinations work best and whether results are improved by including promoted products. Average market-basket analysis and target product influence are among the reports that can be generated in this category.

The InfoSphere Warehouse pack for Supply Chain Insight includes sample Cognos reports in the areas of vendor performance, inventory, distribution center analysis and distribution and shipping:

- Vendor performance

This set of reports helps organizations to achieve a deeper understanding of vendor performance to maximize service while minimizing costs.

Organizations can evaluate whether vendors are reliably fulfilling orders so that inventory levels can be kept to a minimum without jeopardizing selling opportunities. The reports can also be used to evaluate the ability of the vendor to deal with returned goods quickly and efficiently, which is another measure of the vendor's overall value to the organization. Reports in this category include vendor service-level analysis, vendor returns analysis, and vendor agreements; that is, investment buying.

- Inventory

Organizations can use this report series to help keep inventory in line with requirements. Through careful monitoring of the location and status of products and the product demand trends, organizations can ensure that the right products are in the right place at the right time. Specific reports include inventory levels (stock cover, inventory levels), stock availability, inventory location and status, inventory location by type analysis and inventory adjustments analysis.

- Distribution center analysis

Using reports in this category, an organization can evaluate the efficiency of its distribution centers (DCs). The reports offer commonly used metrics and measures to analyze everything from the performance of picking staff to the quantity, weight and volume of inventory processed at DCs. Reports available are distribution center service-level analysis, distribution center throughput, and distribution center productivity.

- Distribution and shipping

These reports are designed to analyze shipping costs and efficiency, including how well the organization is using its distribution fleet to reduce vendor delivery costs by back-hauling inventory from vendors after completing distributions. Specific report titles are load-efficiency analysis and backhaul utilization analysis.

In summary, the InfoSphere Warehouse Insight packs contain all the required artifacts to enable a customer to rapidly and in a cost-efficient manner build a Data Warehouse environment geared at helping customers to understand their

business and provide the means to address those critical questions of where and how to drive their company.

4.6.1 Installation example of InfoSphere Warehouse Insight Pack

In this section, we examine the process of installing the InfoSphere Warehouse pack for Customer Insight and investigate the included assets. Because the same installation process can be utilized for the remaining two Insight packs, they are not part of this example.

When you install an InfoSphere Warehouse Insight pack, a folder in the local file system is created into which the assets of the pack are placed. The contents of this folder are shown in Figure 4-58.

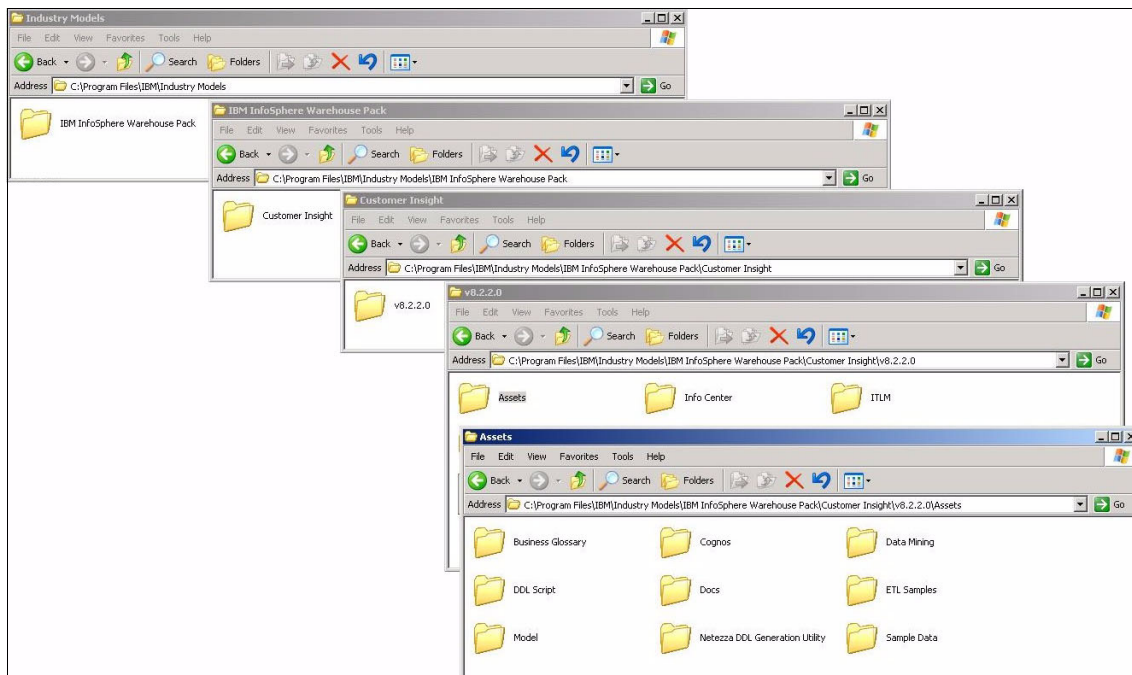


Figure 4-58 Location and assets contained in the IBM InfoSphere Warehouse Customer Insight pack

Investigate each of these folders to see all the assets that are provided. The Cognos folder contains the compressed files required to import reports into Cognos and to import the models into Framework Manager. The DDL Script and Sample Data folders contain the SQL and data to create the sample database. The Model folder contains the files required to import the data models into Design Studio. The Docs folder contains documentation about the Customer Insight pack including the document “IBM InfoSphere Warehouse Pack for

Customer Insight v8.2.2 Getting Started.pdf.” This document gives step-by-step instructions for installing the Warehouse Pack.

A default location on a Linux Operating System-based server for installing the Customer Insight Pack is /opt/IBM/Industry Models.

For our example, we install the models on our Windows client to give our development tools access to all the models. On our client the package is installed into C:\Program Files\IBM\Industry Models.

We perform these steps to install Customer Insight Pack:

1. Open IBM InfoSphere Data Architect and create a new project from **File** → **New** and give the project a name, that is **Customer_Insights**.
2. From the File menu, select **Import** → **General** → **File System** and browse to the Model directory the installation location of the Insight Pack. Select this directory as the source.

3. On the next panel, select all the required model files from within the source directory, and set a location within the new project as the target for these files. See Figure 4-59.

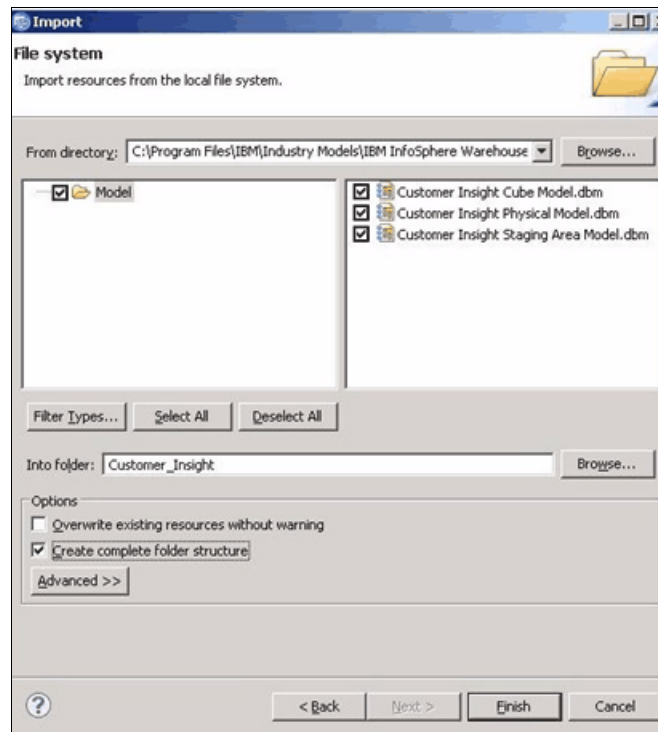


Figure 4-59 Import Model files from IBM InfoSphere Customer Insight Pack into IBM InfoSphere Data Architect

4. We can now navigate the models that have been imported and view the information contained. See Figure 4-60. The complete model is relatively large but has been broken down into a number of star schemas.

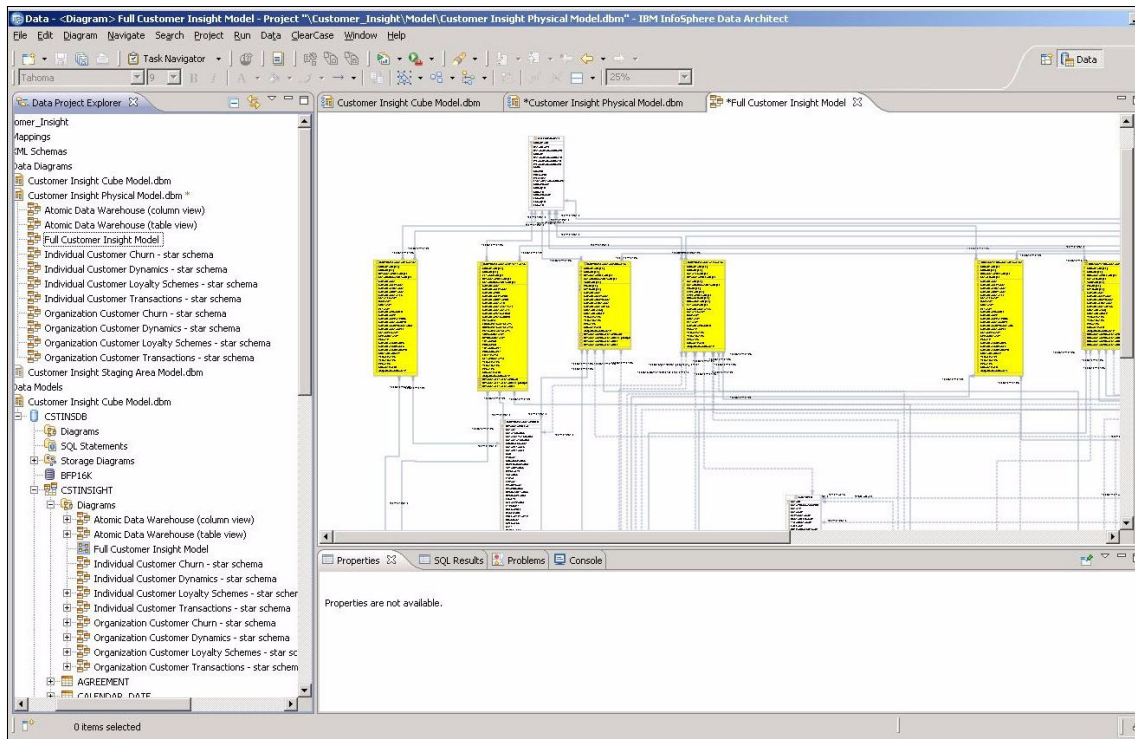


Figure 4-60 Models contained within the InfoSphere Warehouse Customer Insight Pack

5. Figure 4-61 on page 170 shows the same three model files that are loaded into a project within Design Studio. Note the OLAP cube elements within the Customer Insights Cube model file.

This model is set up to do two types of customer analysis: Customer Profiling and Customer Profitability. There are two star schemas for each type. One to deal with Individual customers and one to deal with organization customers. So there are a total of four star schemas.

There are two diagrams for each star schema: a table view and a column view.

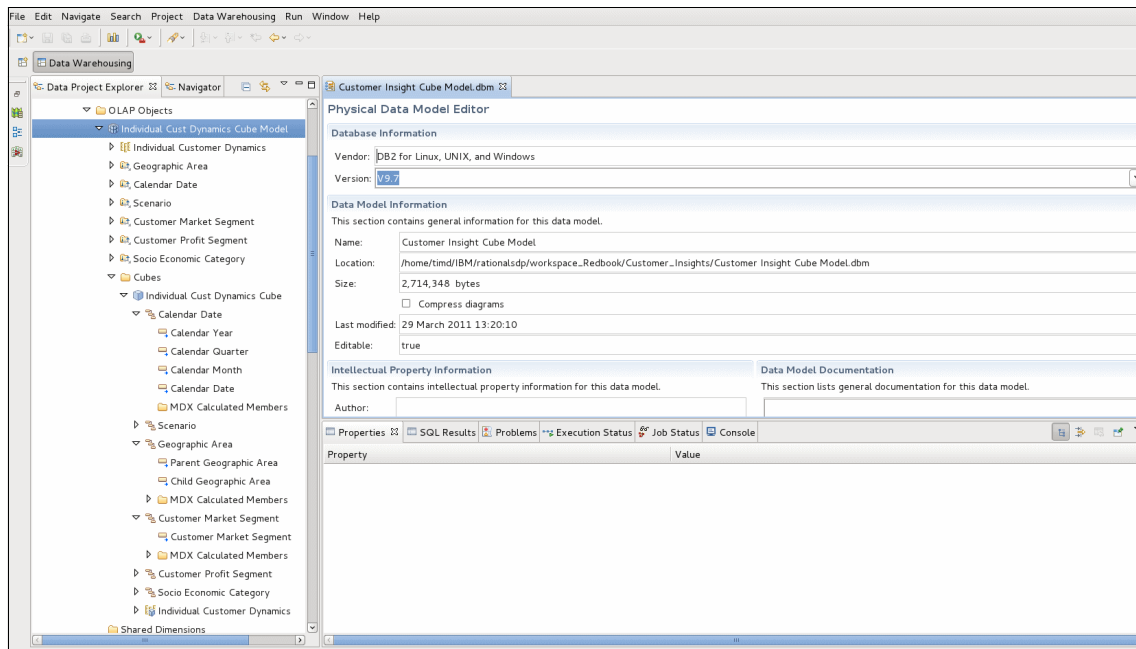


Figure 4-61 The three model files from the Customer Insight pack loaded into IBM Design Studio

6. Double-click the **Individual Customer Loyalty Schema – star schema** to show the table view (Figure 4-62 on page 171). This is a simple star schema with one fact table and six dimension tables.

Most of the dimensions used for the star schemas are derived from the data held within the staging tables where data is loaded before any ETL process

are performed. The structures for these staging tables are also included within the model supplied as part of the pack.

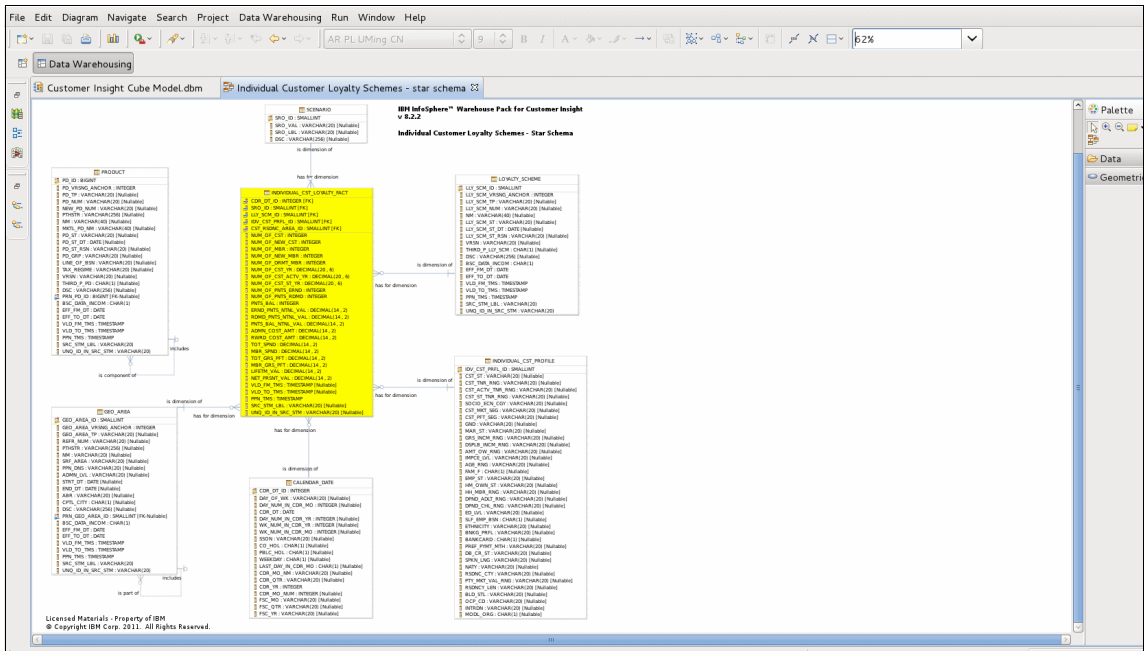


Figure 4-62 One of the star schemas included in the Customer Insight Pack

Under the OLAP objects directory of our model, we see the cube “Individual Cust Dynamics Cube” is defined and ready for either testing within Design Studio or deployment to the InfoSphere Administration Console.

7. Using the Browse Members option of the click menu, we can test and view this OLAP model as shown in Figure 4-63.

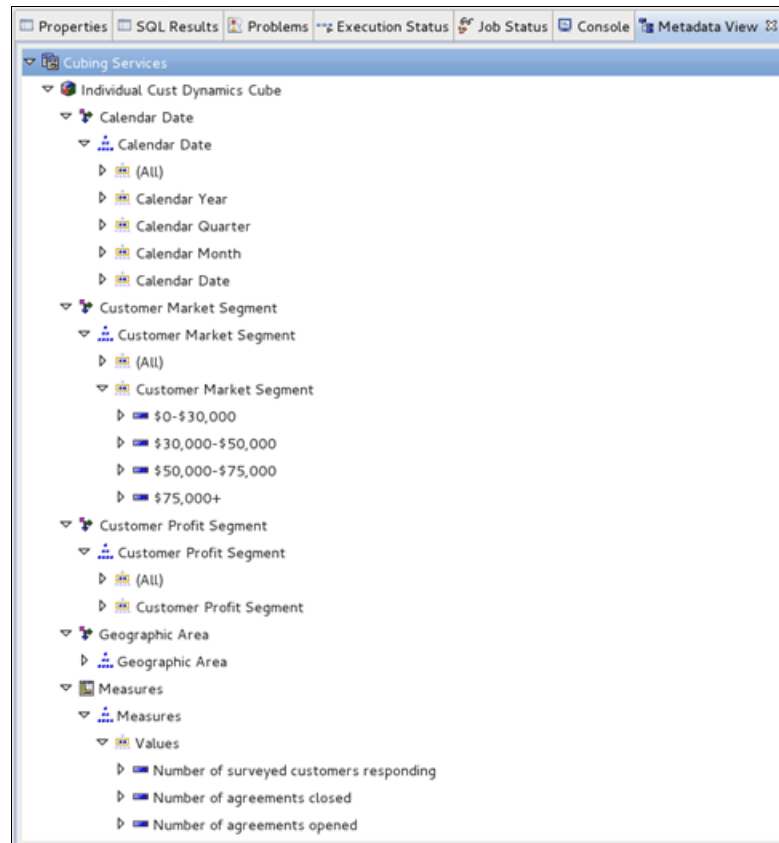


Figure 4-63 Viewing OLAP Model components within Design Studio

Finally, within the Cognos directory we have a number of example reports and dashboards that can be loaded into both Cognos Framework and the IBM Cognos BI server as starting reports by which to present the data contained within our database.

In summary and as demonstrated, the InfoSphere Warehouse Packs contain all the assets needed to help support the development process of creating a Data Warehouse environment specific to a customer's requirements.



Temporal data management and analytics in an operational warehouse

In this chapter we provide an in-depth discussion of a brand-new feature in InfoSphere Warehouse 10.1 known as temporal table support (or more popularly as “time travel query”). Temporal-awareness in the data warehouse makes it possible for business applications to employ time-sensitive logic for more robust and effective applications and business intelligence reporting.

The InfoSphere Warehouse data warehouse is now able to track and act on the notions of both system time and business time, as those terms are defined here:

- ▶ **System time:** This term refers to the start and end time stamps denoting when a record or transaction is valid in the database.
- ▶ **Business time:** This term refers to the start and end time stamps or dates when a given database record is valid from a business perspective.

The use cases for this capability are broad-based and are examined. A detailed technical description is also provided with examples

5.1 Temporal data management concepts and use cases

Considerations regarding time have been part of the concepts of data warehousing and business intelligence from their beginnings. It is hardly possible to conceive of business data or business queries that do not contain an element of time, either as the subject (“when” did something happen) or the context (tell me what happened during a certain time period) of the query or data. Even when the value of time is not explicit, for example, in a list of customers, it is still implicit, as in a list of customers “at this point in time.”

Events and transactions of interest to a business or enterprise always occur at a particular time (orders are placed, currencies change hands, applications are approved, warranties expire); they all happen at a moment in time that is recorded in some transactional system and ultimately reflected in the data warehouse.

Knowing when events occurred allows the enterprise to ask an endless number of questions and perform various analyses to understand how the business is performing, how metrics are trending relative to comparisons across time periods, how business performance might change in the future, and so on. Indeed, achieving meaningful business intelligence and reporting is not possible without the inclusion of time in the data warehouse and subsequent analyses.

This is undoubtedly apparent to those who have experience with data warehouse or business intelligence applications (or who have ever managed their own checkbooks). And, because data warehousing technology has been around for decades, it might seem likely that database management systems built for data warehousing provides built-in features to manage notions of time.

However, except for a few temporal data types and time-based functions, this has largely not been the case. Data warehouse architects, database administrators (DBAs), and application developers have generally been left to construct their own models for representing time and analyzing information relative to time. Although several industry “best practices” have arisen over the years that provide consistent techniques to solve common problems, for the most part there has been a vast proliferation of custom and ad hoc solutions to managing time in a data warehouse. Until recently, that is.

Now, IBM InfoSphere Warehouse 10 includes for the first time native temporal awareness and management features in DB2 that place IBM in the forefront of data warehouse technology providers. In this chapter, we explore how this feature works and how architects, DBAs, and application developers can use it to readily support any number of time-based use cases.

Before looking into the technology, though, we first examine the concepts involved and then consider several example use cases.

5.1.1 Temporal use cases for the operational data warehouse

There are numerous use cases in an operational data warehousing environment where temporal awareness is useful or even required to manage and query the data properly. We consider several here that illustrate key aspects of temporal analysis.

Time-based business results analysis

One of the most common and most useful activities that can be performed in any data warehouse environment is to report on some business results or metrics for some specific point in time or to make a comparison between time periods. For example, you might want to examine revenue totals for a product line for the previous quarter and to compare those with results from the previous quarter or the same quarter last year. Another example is the comparison of customer churn rates before and after a special promotion or retention campaign. A third example is reporting on trends of call failure rates in a telecommunication network over the previous six months. There are any number of well-known examples like these.

The common theme for all these examples is that the data warehouse solution requires some means of capturing and tracking when a relevant business event occurs. In many cases this is the date and time that a transaction took place, for example, a withdrawal was made at a bank ATM; a shipment delivery was confirmed at the destination; an insurance claim was paid to a provider; a user clicked on a particular URL link on a website. Simply knowing when a business-relevant event took place allows the data warehouse solution to easily support the kinds of temporally-based analyses described here. Indeed, data warehouses have been supporting these use cases for many years.

This use case gets more interesting, though, when you consider what happens when information changes over time. For example, how does the system handle changes in the dimensional information?

Changes in dimension attributes

Dimensions and dimension tables are used universally in data warehousing to manage all of the contextual information “about” the business metrics. Common dimensions include customer, product, sales organization, supplier, and so on.

For each dimension, several (or perhaps many) attributes are maintained for each member of the dimension. For example, within a customer dimension, it is common to find attributes that describe characteristics of each customer: their

home and business address, phone numbers, education level, marital status, income bracket, employment status, credit rating, buying preferences, and so on. A temporal-aware data warehouse platform requires a mechanism to handle changes to these attributes over time.

Consider what happens when an existing customer in the USA relocates to a new business address in a different state and continues to do business with your organization. Over time your data warehouse will contain history of transactions for this customer at both the old and the new address. This will cause an update to be made to the customer's home address attributes in the customer dimension table (or tables). But what about the old address? What about the transactions that occurred at the old address?

The impact of this change of address will vary based on the nature and scope of queries that involve this customer. For example, if we report on sales revenue aggregated by product line, or by quarter for the past two years, the change in the customer's business address will have no impact on the results. All of the customer's transactions will be summarized, both those before and those after the relocation, up to the quarter or to the product line level.

However, if the report shows aggregated sales revenue geographically by state, then you expect that the transactions at the old business address will appear under one state summary, and the transactions that occurred after the relocation will appear in a second state summary. That is, the customer's transaction data is now split, since the relocation.

Changes to dimensional attributes such as this are not uncommon and can involve multiple attributes for the same dimension member at different points in time. A robust data warehouse solution must provide some means to represent and manage these types of changes over time. Queries for reports and applications must be written in such a way to ensure that data that is split in time across changed attributes returns accurate results.

One common way this type of change is handled is through a modeling and implementation technique known as *slowly changing dimensions* that was first put forth and popularized by Ralph Kimball.¹ This technique is widely accepted as the standard approach to managing these types of changes, but in the following sections we describe how this can be handled in a new way with InfoSphere Warehouse 10.

Changes in dimension hierarchy

Another class of changes that must be managed in the dimensional structure over time is the hierarchy. A useful example of this in an operational retail industry scenario can be found in product hierarchies.

¹ *Ralph Kimball et al. The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, 2008

In a typical retail enterprise, individual products are organized in product lines that might then also aggregate up to product categories and brands. Product lines provide convenient summarizations of products for reporting and analysis purposes, and for support of manufacturing operations, marketing campaigns, and other areas. Products can be realigned from one product line or category to another at any time for any number of business reasons to support increased sales, more efficient distribution, manufacturing realignment, and so on. Commonly such product realignment occurs frequently, and operational data warehouse solutions must take these product realignments into account, especially when reporting is mixed with historical data.

For example, consider a manufacturer and distributor of sports shoes. Assume they market shoes for specific sports, including a product line of basketball shoes. They might sell shoes in this product line for some period of time. Then one day the company signs a big endorsement agreement with a major professional basketball star. Having a number of shoes for different sports with similar promotional deal, our vendor decides to manage these products under a distinct product line called Signature Series. So from that day onward, they are selling the exact same shoe, only now it is tracked under the Signature Series product line instead of the Basketball product line.

This is a similar situation to the changing dimensional attributes discussed, but it is different in that we are now talking about aggregation paths and parent and child relationships in the dimensional hierarchy. We are changing the logical structure of the product dimension.

Now, when the shoe company runs reports on sales of shoes by product line, there are different ways that this product realignment can be handled. If we query for sales of shoes in the Signature Series for the past 24 hours, then our aggregate will include the new addition of the basketball shoe. But what about queries that span back in time to include a period prior to the product realignment?

For example, if we were to compare aggregate sales of Signature Series shoes in the past 24 hours with daily aggregate daily sales over the past 14 days, how can we handle the fact that the Signature Series did not always include the basketball shoe?

The answer depends on what insights we are trying to learn from the query results. If we want to see how the Signature Series is performing relative to the exact same product mix over the past 14 days, then the basketball shoe will be included in the aggregate results for each daily summarization as through it had always been a part of that product line.

Alternatively, if we are more interested in seeing the impact that the addition of the new basketball shoe has had on the Signature Series results, then we want

each daily aggregate to include only products that were truly in the Signature Series for each particular day in the report. In this second scenario, the product mix in the Signature Series might be different for each daily aggregate. There are subtle and not-so-subtle nuances in the nature of each of these example business queries that have to be handled by either the data warehouse platform itself or coded into the business application logic for each application.

Information with effective start and end dates

It is quite common for activities and information of an enterprise to have validity dates or effective start and end dates (or date stamps). In fact, all of the example use cases given so far are actually specific examples of this general use case. Even when no effective begin and end time or date is stated, it is implicitly understood to being “now” and to extend “indefinitely” into the future. Many other examples fall into this category of temporal awareness, such as:

- ▶ Effective dates for an insurance policy
- ▶ Start and end dates for a marketing promotion
- ▶ Effective start and end dates for a lease agreement or any legal contract
- ▶ Grace period for a credit card monthly payment
- ▶ Period of time that a technology patent is in force
- ▶ Effective start date for a new salary or employment position
- ▶ Start and end dates for a tax filing or payment period

Many other examples can be listed, and they all have applicability across data warehousing, including operational warehousing. In an operational scenario, you might have to respond to a new sales promotion, review current account conditions or take action prior to a deadline. Comparison of current conditions must be made with previous offers or year-to-date summarizations.

What each of these examples has in common is the need for the system to understand the business logic of the effective start and end dates (or dates and times) and what values are in force during that time. The policies of the business or organization will dictate what activities, operations, values, or transactions are possible or valid within the effective start-end date range.

Handling changes in the database context

A completely different category of temporal awareness deals with changes to the data itself in the context of the database or data warehouse system. For these use cases, it is of most interest to note the following two items:

- ▶ Some piece of data changed within a data record
- ▶ The time stamp of when that change occurred

Further, it is useful to track the history of changes to data records.

This type of temporal capability typically includes the following use cases:

- ▶ **Auditability** - What records of a table have changed in the data warehouse, and when did they change?
- ▶ **Recoverability** - Can we restore a table record to a previous state of values at a particular point of time in the past?
- ▶ **History** - Related to auditability and recoverability, can we view a prior state of values of a table record at any time in the past?
- ▶ **Traceability and governance** - Did a particular change to a value in the database record occur at the correct time, or has a series of changes across tables occurred in the correct sequence?

Because operational data warehousing deals with frequent inserts and updates of the data in the system, this kind of temporal awareness is particularly useful to manage data consistency, especially for sensitive data.

5.1.2 Temporal concepts

From the preceding discussion about use cases, several repeating concepts have emerged. These concepts are defined in summary here and are further developed in the discussion of temporal awareness in InfoSphere Warehouse Advanced Enterprise Edition 10.

Business time

Business time refers to a date or time stamp that has particular value to a business process or in a business context. Validity dates and effective dates for policies, activities, or any business process are example uses of business time. Business time is more interested in when something happens, will happen, or is valid in the real or business world, rather than when something happens in the database system. For example, business time is interested in when a lease agreement is in force, not when the lease agreement was entered into the database.

System time

System time refers to a time stamp of precisely when a database record was inserted, updated, or deleted in the database system itself. System time is more concerned with the exact moment when any kind of data change occurred in the database (the database transaction) independent of the business context of that change. For example, system time is interested in when a transaction occurred requesting that a customer's mail be held while they are on vacation rather than the dates that the mail will be held.

Bitemporal

As the name implies, bitemporal indicates the ability to track and operate on both business time and system time. There are many operational warehousing scenarios where this is required, such as the ability to track when a customer requested an update to their banking account (for purposes of recovery and auditing) and to track when the change is to take effect (such as a planned change of address).

Start time

Start time applies to both business and system time. In the context of system time, it refers to the time stamp (date and time) indicating when a record change (insert, update, delete) occurred in the data warehouse system. For system time, the start time is always in the past or (for an instant, anyway) in the present. Because it tracks when a database transaction occurs, it is never in the future.

In a business time context, it refers to the time (date or time stamp) when something is valid or takes effect, such as a change to a customer account. Business start time can refer to past, present, or future time.

End time

End time applies to both business and system time. In the context of system time, it refers to the time stamp (date and time) indicating when a record in the database system is or was valid. For the record that is currently valid, the end time refers to some point in the distant future that logically means “forever.” For history records, the end time refers to the first moment that a record was no longer a current record in the database, that is, the time when some database transaction changed the record.

In a business time context, it refers to the date (or time stamp) at which point something ceases to be valid or in force, such as a promotion, policy, or temporary change of account status. In many cases it is the “expiration date” for some business process or activity, for example, the last date on which a lease agreement is in force or the last day on which a contracting bid may be accepted.

For both system and business time, end time may refer to the past, present, or future.

Inclusive-inclusive time period

Inclusive-inclusive indicates that a time period includes both its start time and end time (dates or time stamps). This is also known as a closed-closed period. The period starts at the moment of the start time and runs up to and including the date or time of the end time.

For example, if an insurance policy is valid from the start of June 1 through the month of June, then the inclusive-inclusive period will be expressed as June 1 through June 30. The end time includes all of June 30.

Inclusive-exclusive time period

Inclusive-exclusive indicates that a time period includes the first moment of the start time (date or time stamp) and runs up until the first moment of the end time (date or time stamp). This is also referred to as a closed-open period.

For instance, the example insurance policy mentioned that is effective for the month of June requires a start time of June 1 and end time of July 1. The policy is effective up to the first moment of July 1.

As discussed in the following section, DB2 manages time using inclusive-exclusive time semantics. We now turn our attention to the new temporal data management and query features in InfoSphere Warehouse 10.

5.2 Temporal data management in InfoSphere Warehouse Advanced Enterprise Edition 10

The release of InfoSphere Warehouse 10, including the Advanced Enterprise Edition, provides brand-new temporal data management and query features that fully support the concepts and use cases described in the previous section. These new features are based on the ISO SQL 2011 specifications for temporal data support². Built-in support for temporal data handling provides a consistent and automatic means of handling temporal data and logic for applications.

Prior to InfoSphere Warehouse 10, it was necessary for DBAs and application developers to develop custom programming, make use of DB2 triggers, or use combinations of these techniques to implement similar temporal logic. Now it is possible to take advantage of industry standard techniques and SQL syntax to implement temporal awareness in the database and in applications that is more seamless to applications and more easily implemented and maintained.

Temporal data management in InfoSphere Warehouse 10 covers the concepts of both system time and business time. We discuss these in turn in the following sections. We consider how the temporal tables are created, modified, queried and maintained.

² The International Organization for Standards (ISO) specifications for SQL are contained in multiple parts. The relevant specifications are designated *ISO/IEC 9075-n:2011* where *n* refers to the specification part number. All ISO SQL 2011 specifications can be found at www.iso.org.

5.2.1 Temporal data management with system time

System time refers to the time (a time stamp) at which point a data record is changed in InfoSphere Warehouse 10 (that is, in a DB2 table). This change can be one of an insert, update, or delete, and typically it is associated with some business transaction of interest. The following are examples of events for which you might want to track system time:

- ▶ A sales transaction is inserted into the database (table row insert)
- ▶ A medical insurance plan coverage amount for emergency care is increased (table row update)
- ▶ A cable services customer drops the data plan from their package including cable TV and voice services (database delete)

For the purposes of system audit, recovery or snapshot point-in-time queries, it might be useful to be able to track precisely when these changes occurred and to maintain a history of the data records in the database prior to the changes.

This is accomplished using the system time features of the temporal data management in InfoSphere Warehouse 10.

For illustration purposes in the following discussion, consider the information shown in Figure 5-1. This table captures customer profile information.³

CUSTOMER_ID	GENDER	MARITAL_STATUS	INCOME_RANGE	AGE_RANGE
1	Male	Single	\$30,000-\$50,000	35-50
10	Male	Married	\$30,000-\$50,000	20-35
12	Female	Married	\$30,000-\$50,000	20-35
14	Male	Single	\$75,000+	20-35
16	Male	Married	\$50,000-\$75,000	20-35
19	Unknown	Married	\$75,000+	Unknown

Figure 5-1 *CUSTOMER_PROFILE* table

Notice that the table contains several demographic attributes such as marital status and gender for each customer. Some sample values are shown in the table. Over the course of time, several of these attribute values might change for a customer, and you might find it useful to track those changes using system time temporal tables in InfoSphere Warehouse 10.

³ This table is derived from the `INDIVIDUAL_CST_PROFILE` table in the Customer Insight Warehouse Pack solution.

InfoSphere Warehouse 10 makes several additions to a standard table to manage it with system-period temporal awareness.

New time stamp columns in the base table

Three new time stamp columns are included in the base table (or query table) to track period of validity for each row in the table. The new columns are listed here:

- SYSTEM_TIME start This captures the time stamp at which time the new row values are valid; that is, the moment at which the changes were made in DB2.
- SYSTEM_TIME end This is the time stamp indicating the moment when the row values are no longer valid.
- TRANS_START This tracks the time when the transaction first executed a statement that changed the row’s data. This allows for application logic to determine allows across multiple tables that have been updated by the same transaction.

Each of these columns is type `TIMESTAMP(12)`, and all columns are to be created as `GENERATED ALWAYS` so that DB2 will automatically generate the values on `INSERT`, `UPDATE`, and `DELETE` operations.

Furthermore, these columns can be defined as `IMPLICITLY HIDDEN` so that they do not appear in `SELECT *` statements, but only if explicitly included in the select list. Figure 5-2 shows our `CUSTOMER_PROFILE` table modified with the new columns for system-period temporal data management. (The `TRANS_START` column omitted to simplify the example and discussion.)

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
10	Male	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
12	Female	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
14	Male	Single	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
16	Male	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
19	Unknown	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0

Figure 5-2 *CUSTOM_PROFILE* table enabled for system-period temporal data

Notice that all the rows in the table have a `SYS_END` column value of “9999-12-30”, or December 30, 9999. This effectively means that the current row value is valid indefinitely. We will see how these new columns are used as the discussion continues.

The new history table

A new table is introduced for system-period temporal data management that tracks the history of rows that have been updated or deleted. The history table is identical in structure (same columns and types) as its base table. It is used to capture the “old” or modified and deleted rows of the table, and the valid start and end times for each changed row.

Figure 5-3 shows an example of the history table after an update was made to a record in the CUSTOMER_PROFILE table. Notice the SYS_END column value is no longer set to the indefinite value of “9999-12-30” but rather to a valid time stamp in the past that indicates when the record was updated in the database.

The following section details how these start and end columns are used in the history table and in queries.

CUSTOMER_ID	GENDER	MARITAL_STATUS	AGE_RANGE	SYS_START	SYS_END
14	Male	Single	20-35	2012-07-08 22:37:05.737418	2012-07-08 23:31:40.271781

Figure 5-3 Sample history data for the CUSTOMER_PROFILE table

Creating a system-period temporal table

Now that you have seen the new objects required for managing system-period temporal tables, we examine how these tables are created. To create a new system-period temporal table from scratch, you can either take advantage of new features in the InfoSphere Warehouse Design Studio tooling, or create them with SQL statements manually. We look at both techniques here.


Using the Design Studio tooling

To create a system-period temporal table in InfoSphere Warehouse using Design Studio, simply define the table as usual and then select **System time period** under the “Temporal attributes” heading in the table Properties tab as shown in Figure 5-4.

<TemporalTable> CUSTOMER_PROFILE									
Columns									
Distribution Key	Name	Primary Key	Domain	Data Type	Length	Scale	Not Null	Generated	Default Value/Generate i Period
Data Partitions	CUSTOMER_ID	<input checked="" type="checkbox"/>		BIGINT			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Table Spaces	GENDER	<input type="checkbox"/>		VARCHAR	20		<input type="checkbox"/>	<input type="checkbox"/>	
MDC	MARITAL_STATL	<input type="checkbox"/>		VARCHAR	20		<input type="checkbox"/>	<input type="checkbox"/>	
Versioning	INCOME_RANGI	<input type="checkbox"/>		VARCHAR	20		<input type="checkbox"/>	<input type="checkbox"/>	
Column Masks	AGE_RANGE	<input type="checkbox"/>		VARCHAR	20		<input type="checkbox"/>	<input type="checkbox"/>	
Row Permissions	SYS_START	<input type="checkbox"/>		TIMESTAMP	12		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AS ROW BEGIN SYSTEM_TIME begin
Relationships	SYS_END	<input type="checkbox"/>		TIMESTAMP	12		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AS ROW END SYSTEM_TIME end
Volumetrics	TRANS_START	<input type="checkbox"/>		TIMESTAMP	12		<input type="checkbox"/>	<input checked="" type="checkbox"/>	AS TRANSACTION STA
Documentation									
Annotation	Temporal attributes:								
	<input checked="" type="checkbox"/> System time period <input type="checkbox"/> Business time period								

Figure 5-4 Design Studio properties for system-period temporal table definition

Notice several items in this figure:

- ▶ The SYS_START, SYS_END and TRANS_START columns are automatically provided. These can be renamed to any names preferred by the DBA or application developer who creates the table.
- ▶ The “Generated” flag is checked for the three TIMESTAMP columns, indicating that DB2 will automatically generate the values for these columns.
- ▶ The SYS_START column has additional notation for “AS ROW BEGIN” and “SYSTEM TIME begin” tagging it as the begin time for the system-period versioning.
- ▶ The SYS_END column has similar notation for “AS ROW END” and SYSTEM TIME end” tagging it as the end time for the system-period versioning.
- ▶ The TRANS_START column has additional notation identifying it as the “TRANSACTION START ID”.
- ▶ The table name CUSTOMER_PROFILE is tagged with the annotation “<TemporalTable>” which causes Design Studio to display a temporal table icon when listing the table in the project or database explorers, such as  CUSTOMER_PROFILE.

All of these attributes are defined automatically when you select the “System-period” check box, thus simplifying the task of creating the system-period temporal table.

Tip: You can also simplify creation of the system-period temporal table in Design Studio by right-clicking the schema in the project explorer and selecting **Add Data Object** → **System-period Temporal Table**. This creates a table skeleton with the three TIMESTAMP columns for system start, end, and transaction start already defined with all the necessary attributes.

Either of these techniques in Design Studio creates a system-period temporal table that appears in an overview diagram as shown in Figure 5-5.

CUSTOMER_PROFILE	
CUSTOMER_ID	: BIGINT
GENDER	: VARCHAR(20)
MARITAL_STATUS	: VARCHAR(20)
INCOME_RANGE	: VARCHAR(20)
AGE_RANGE	: VARCHAR(20)
SYS_START	: TIMESTAMP(12)
SYS_END	: TIMESTAMP(12)
TRANS_START	: TIMESTAMP(12)

Figure 5-5 System-period temporal table created in Design Studio

Creating the system-period temporal table manually with SQL

It is also possible to create the system-period temporal table manually using SQL, as shown in Example 5-1.

Example 5-1 SQL manually coded to create the system-period temporal table

```
CREATE TABLE "CSTINSIGHT"."CUSTOMER_PROFILE" (  
    "CUSTOMER_ID" BIGINT NOT NULL,  
    "GENDER" VARCHAR(20),  
    "MARITAL_STATUS" VARCHAR(20),  
    "INCOME_RANGE" VARCHAR(20),  
    "AGE_RANGE" VARCHAR(20),  
    "SYS_START" TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN ,  
    "SYS_END" TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END ,  
    "TRANS_START" TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID ,  
    PERIOD_SYSTEM_TIME("SYS_START", "SYS_END")  
);
```

Notice key aspects of the SQL example:

- ▶ The three **TIMESTAMP** columns must be specified as shown with the appropriate “AS ROW BEGIN”, “AS ROW END” or “AS TRANSACTION START ID” as discussed previously.
- ▶ The **TIMESTAMPS** for start and end must be “NOT NULL”.
- ▶ All three **TIMESTAMPS** are to be “GENERATED ALWAYS” by DB2.
- ▶ The **SYS_START** and **SYS_END** columns can have any user-defined name so long as they appear appropriately in the **PERIOD_SYSTEM_TIME()** clause to indicate which column is the start time and which column is the end time.

Tip: The **TIMESTAMP** columns can also be specified with **IMPLICITLY HIDDEN** so that they will not appear in **SELECT *** results.

Further manipulation in Design Studio

After the system-period temporal table has been created by any means, it can be used freely in all Design Studio activities just as any other standard table. For example, it can be used in data flows, mining flows, or cube model definitions as needed to support any applications or use cases.

Creating the history table

InfoSphere Warehouse (and DB2) 10 uses a history table to track changed table rows in system-period temporal tables. This history table must be created to enable the temporal data management. Like the main (or base) table, it can be

created using either the Design Studio tooling or by using SQL that was coded manually.

The history table is to have all of the same columns as the main table so that it can track all changes without losing any history or previous values.

Creating history tables with Design Studio

If the system-period temporal table has been created using Design Studio as discussed in “Using the Design Studio tooling” on page 184, then the history table is created automatically at the same time. The exact same column definitions are used in the history table as the base table.

By default, Design Studio appends “_HIST1” to the end of the base table name. This can be changed to any user-defined suffix that adheres to DB2 table naming and uniqueness constraints.

Creating history tables with SQL

As with base temporal tables, the history table can be created by hand-coded SQL. The easiest and shortest SQL statement uses the “LIKE” clause as shown in Example 5-2.

Example 5-2 Simple SQL to create the history table

```
CREATE TABLE CUSTOMER_PROFILE_HISTORY LIKE CUSTOMER_PROFILE;
```

The more complete SQL statement to create the history table with each column specified is shown in Example 5-3.

Example 5-3 Long-form SQL for creating a history table

```
CREATE TABLE "CSTINSIGHT"."CUSTOMER_PROFILE_HISTORY" (  
    "CUSTOMER_ID" BIGINT NOT NULL,  
    "GENDER" VARCHAR(20),  
    "MARITAL_STATUS" VARCHAR(20),  
    "INCOME_RANGE" VARCHAR(20),  
    "AGE_RANGE" VARCHAR(20),  
    "SYS_START" TIMESTAMP(12) NOT NULL,  
    "SYS_END" TIMESTAMP(12) NOT NULL,  
    "TRANS_START" TIMESTAMP(12)  
);
```

Notice several differences in the DDL between the history table and the base table for system-period temporal tables:

- ▶ The temporal table **TIMESTAMP** columns lack both the “GENERATED ALWAYS” and the other “AS ROW BEGIN”, “AS ROW END”, and “AS

TRANSACTION START ID" notation. These columns are simple
TIMESTAMP columns.

- There is no "PERIOD SYSTEM_TIME" clause in the DDL.

In general, the history table definition is merely a mirror of the base table column definitions.

Enabling system-period temporal versioning

To "connect" the base and history tables together and to enable the system-period temporal data management in InfoSphere Warehouse 10, an ALTER TABLE statement is required. This step is included automatically when Design Studio is used to create the system-period temporal table. When using manually coded SQL, the statement shown in Example 5-4 must be used.

Example 5-4 SQL statement for enabling system-period temporal data management

```
ALTER TABLE "CSTINSIGHT"."CUSTOMER_PROFILE" ADD VERSIONING USE HISTORY TABLE  
"CSTINSIGHT"."CUSTOMER_PROFILE_HISTORY";
```

First all three steps must be completed:

1. Create the base system-period temporal table.
2. Create the history table.
3. Enable versioning.

When this is done, InfoSphere Warehouse (and DB2) 10 will begin tracking changes to the defined temporal table.

If the base system-period temporal table and the history table column definitions do not match, for example, if a column is missing from or added to the history table, then an error similar to Example 5-5 is received.

Example 5-5 Typical error when the history table does not match the base temporal table

```
Table "TEMPORAL_TEST_HISTORY" was specified as a history table, but the table  
definition is not valid for a history table. Reason code "7".. SQLCODE=-20523,  
SQLSTATE=428HX, DRIVER=3.63.108
```

In this example, the history table TEMPORAL_TEST_HISTORY was defined with one of the columns in the base table TEMPORAL_TEST intentionally omitted.

Altering existing tables to enable temporal data management

It is also possible to alter existing tables to enable system-period temporal data management. This can be accomplished without having to alter any existing applications unless they have to take advantage of the new features.

The easiest way to accomplish this in InfoSphere Warehouse 10 is to use Design Studio. Simply select the table of interest in the data project explorer and view the properties. Select the “System time period” temporal attribute as shown in Figure 5-6. The necessary columns are automatically generated along with the new history table definition.

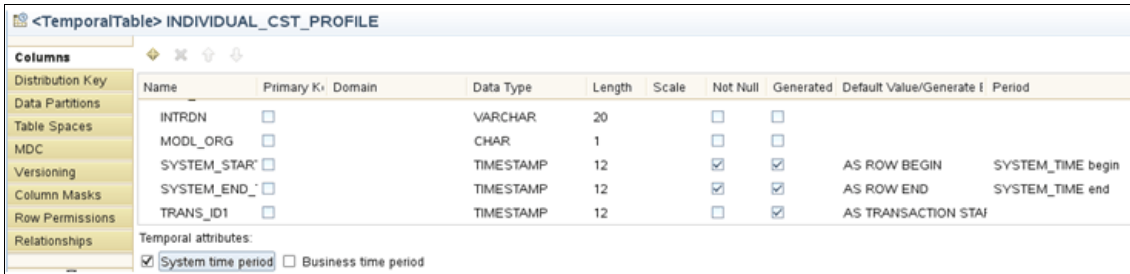


Figure 5-6 Altering a table to enable system-period temporal data management

After the tables are defined in the data project explorer, the following two-step process enables system-period temporal management:

1. Alter the existing table in the database to add the new columns.
2. Create the history table and enable versioning.

In the following section we explain how to complete these tasks in Design Studio.

Alter the existing table to make it a temporal table

After the table has been modified in the data project explorer, you must reflect the changes in the physical database table. Execute the following steps:

1. Right-click the table name in the project explorer and select **Compare with** → **Another database object**. Use the dialog box to select the corresponding table in the physical database. In the example shown in Figure 5-7 on

page 190, we modify the INDIVIDUAL_CST_PROFILE table to enable system-period temporal management.

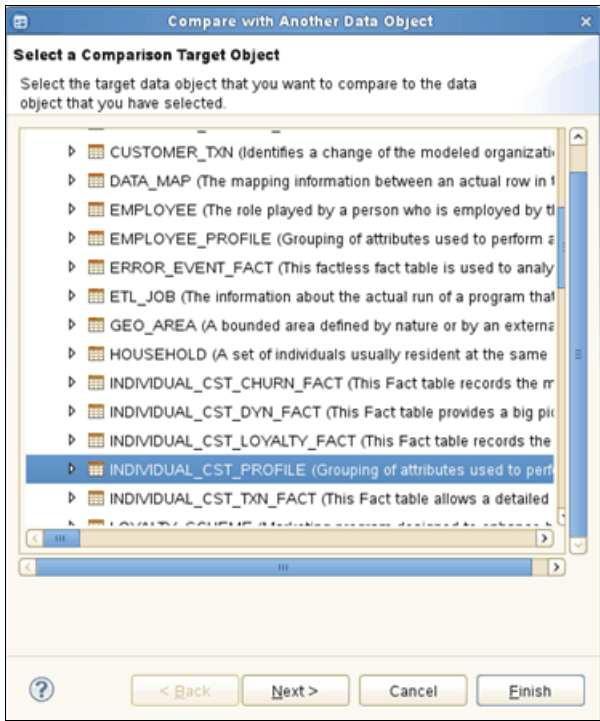


Figure 5-7 Selecting the physical table for comparison

2. In the next dialog, select to compare all object types as shown in Figure 5-8.

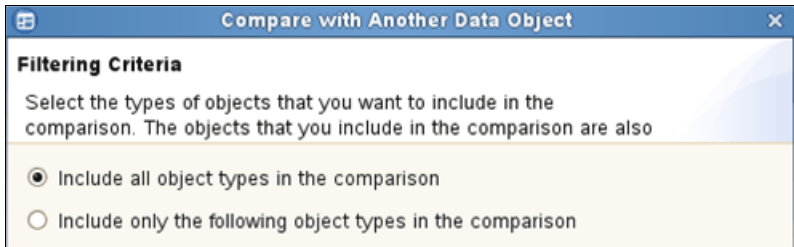




Figure 5-8 Specify the object types for comparison

3. The compare results panel then allows you to synchronize the physical database table with the new table design to enable temporal data management for system-period handling. You simply have to select the “Copy

from left to right”  icon (assuming that the new table is on the left; otherwise click the “right to left” button).

4. Select the Generate right delta DDL button  to generate the necessary SQL script to modify the table.
5. Save the generated script and execute it to modify the table to enable system-period temporal support.

These steps essentially execute SQL statements to rename the existing table, create a new table with the system-period temporal columns and statements, then populate the new table from the old table.

When to perform this task: These steps make a copy of the existing table. Be aware that if this table is significantly large, the operation can take a long time to complete. If possible, perform this task during off-peak or non-production periods of data warehouse utilization.

After these steps are complete, perform the history table and ALTER statements as described in the previous sections.

Automatic schema evolution

Regarding creating and enabling system-period temporal tables, be aware that InfoSphere Warehouse 10 support automatic schema evolution between an enabled base table and its corresponding history table for changes that do *not* result in potential data loss. Examples of operations on the base table that are automatically reflected in the history table include:

- ▶ Adding a new column to the base table. A corresponding column is simply added to the history table.
- ▶ Altering a column data type that causes no data loss, that is, types for which DB2 supports automatic conversion. For example, it is possible to change a column type from INTEGER to VARCHAR(50) because DB2 can convert the INTEGER values into corresponding VARCHAR values, and all values will fit in the 50 characters available.

Modifications to the base table that can result in data loss and are therefore blocked include these examples:

- ▶ Altering a VARCHAR typed column to a shorter length (for example, from VARCHAR(50) to a VARCHAR(2). Any column values over two characters in length will be truncated and result in data loss, so therefore the operation is blocked.
- ▶ Dropping a column. This leads to loss of all values in the column so the operation is blocked.

Dropping a column: There are cases where it is necessary to make the table modifications previously shown, but because those modifications can cause a potential loss of data, they are blocked by default. For those cases, when this action is justified and the consequences are well understood, it is possible to disable versioning and make the necessary modifications.

A useful example of dropping a column from a system-period temporal table is given in *Best Practices: Temporal Data Management with DB2* in the section “How to drop a column from a system-period temporal table”. This paper is available at the following website:

<http://www.ibm.com/developerworks/data/bestpractices/temporal/index.html>

5.2.2 System-period temporal tables

Now that you have an understanding of how to create and enable tables for system-period temporal data management, we shift our attention to how to manipulate and query these tables. We start with describing what happens when data is inserted into the base table.

Inserting data into a system-period temporal table

When data is inserted into a system-period temporal table, the new rows are added to the table just like any other regular table with the exception of how the new `TIMESTAMP` columns are assigned. The `INSERT` statement itself is 100% identical to an `INSERT` into a regular table. This feature allows the introduction of system-period temporal tables to have no impact on existing applications and operations.

If we return to our example tables from the previous discussion, the `CUSTOMER_PROFILE` table, we might have the following data already present as shown in Figure 5-9 on page 193.

Note: For all examples used in this section on system-period tables, the `SYS_START` and `SYS_END` values are shown only the `DATE` part of the `TIMESTAMP(12)` value. This is a shorthand convenience for the purposes of simplifying the illustration only.

The true value of these columns is a full `TIMESTAMP(12)` value.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08	9999-12-30
10	Male	Married	2012-07-08	9999-12-30
12	Female	Married	2012-07-08	9999-12-30
14	Male	Married	2012-07-08	9999-12-30
16	Male	Married	2012-07-08	9999-12-30
19	Female	Married	2012-07-09	9999-12-30

Figure 5-9 Example CUSTOMER_PROFILE contents before INSERT

Assume that we insert the records as shown in Example 5-6.

Example 5-6 Inserting records

```
INSERT INTO CUSTOMER_PROFILE(CUSTOMER_ID, GENDER, MARITAL_STATUS) VALUES(20,
'Male', 'Married');
```

```
INSERT INTO CUSTOMER_PROFILE(CUSTOMER_ID, GENDER, MARITAL_STATUS) VALUES(21,
'Male', 'Single');
```

The new rows are inserted as usual into the CUSTOMER_PROFILE table, and DB2 automatically generates the values for SYS_START, SYS_END and TRANS_START. Because there are no existing rows corresponding to the new rows that are updated or deleted, there is nothing added to the HISTORY table for these inserts.

Figure 5-10 shows the updated CUSTOMER_PROFILE table after the insert statements.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08	9999-12-30
10	Male	Married	2012-07-08	9999-12-30
12	Female	Married	2012-07-08	9999-12-30
14	Male	Married	2012-07-08	9999-12-30
16	Male	Married	2012-07-08	9999-12-30
19	Female	Married	2012-07-09	9999-12-30
20	Male	Married	2012-07-09	9999-12-30
21	Male	Married	2012-07-09	9999-12-30

Figure 5-10 CUSTOMER_PROFILE table after INSERTs

Figure 5-11 shows the empty HISTORY table.

Figure 5-11 CUSTOMER_PROFILE_HISTORY after INSERTs

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END

The SYS_START values for the new CUSTOMER_ID records 20 and 21 reflect the date (and time, though not shown) when the new records were inserted into the base table. Like all records in the base table, the SYS_END date and time for those records is effective indefinitely though precisely until December 30, 9999. This indicates the rows have “current” or “unexpired” data.

Updating data in the system-period temporal table

When we update existing records in the system-period temporal table, the history table begins to play a role in managing our information. For example, we update the marital status to “Married” for customer number 1. Also, we record that customer 10 was recently divorced so their status changed to “Divorced”. Example 5-7 shows the SQL for these changes.

Example 5-7 SQL statements to update the CUSTOMER_PROFILE table

```
UPDATE CUSTOMER_PROFILE SET MARITAL_STATUS='Married' WHERE CUSTOMER_ID=1;

UPDATE CUSTOMER_PROFILE SET MARITAL_STATUS='Divorced' WHERE CUSTOMER_ID=10;
```

We can see the impact of the UPDATE SQL on the CUSTOMER_PROFILE table in Figure 5-12.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Married	2012-07-09	9999-12-30
10	Male	Divorced	2012-07-09	9999-12-30
12	Female	Married	2012-07-08	9999-12-30
14	Male	Married	2012-07-08	9999-12-30
16	Male	Married	2012-07-08	9999-12-30
19	Female	Married	2012-07-09	9999-12-30
20	Male	Married	2012-07-09	9999-12-30
21	Male	Married	2012-07-09	9999-12-30

Figure 5-12 CUSTOMER_PROFILE table after UPDATES

We also see the impact on the HISTORY table in Figure 5-13.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08	2012-07-09
10	Male	Married	2012-07-08	2012-07-09

Figure 5-13 CUSTOMER_PROFILE_HISTORY after UPDATES

Reviewing the base table in Figure 5-12 on page 194, we see that in addition to the new values for MARITAL_STATUS for CUSTOMER_IDs 1 and 10, the SYS_START column values have also been updated with the current date of the UPDATE operations. The SYS_END dates are the usual December 30, 9999 for currently valid records.

In the HISTORY table list of Figure 5-13, however, we see two records inserted that were not there prior to the UPDATE statements. There is one record for each UPDATE, with a row holding the previous values for every column. The SYS_START time holds the system time that the historical record was first valid, and the SYS_END time holds the time that the historical record became invalid or expired. The SYS_END for CUSTOMER_ID 1 in the HISTORY table matches the SYS_START time for the updated record in the base table.

SYS_START is *inclusive*. SYS_END time is *exclusive*. In the discussion of queries we show how this ensures query consistency.

In plain language, CUSTOMER_ID 1 was single from July 8, 2012 (or at least that is when the data was entered into the database) until July 9, 2012. On July 9 their status changed to married and is the currently valid status. All of the SYS_START and SYS_END information in both tables was automatically inserted or updated as necessary by InfoSphere Warehouse and DB2 without any intervention by the DBA or application logic. (Although this is not shown, DB2 also updated the TRANS_START information in both tables.)

Any subsequent updates to base table records, including the same records already updated, is reflected in the HISTORY table and SYS_START and SYS_END times accordingly. For example, if CUSTOMER_ID 10 remarries, then the table will be updated again with the statement shown in Example 5-8.

Example 5-8 Updating a record for a second time

```
UPDATE CUSTOMER_PROFILE SET MARITAL_STATUS='Married' WHERE CUSTOMER_ID=10;
```

The results in Figure 5-14 on page 196 show the updated CUSTOMER_PROFILE table records with the current marital status for CUSTOMER_ID 10 ("Married").

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Married	2012-07-09 03:04:11.32564	9999-12-30 00:00:00.0
10	Male	Married	2012-07-09 03:04:11.336528	9999-12-30 00:00:00.0
12	Female	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
14	Male	Married	2012-07-08 23:31:40.271781	9999-12-30 00:00:00.0
16	Male	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0

Figure 5-14 CUSTOMER_PROFILE after second UPDATE of same customer

Figure 5-15 shows the updated HISTORY table, with a new record inserted for the previous record for that customer. Now we see (correctly) two records in the history table for the same customer.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.239378
10	Male	Married	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.249488
10	Male	Divorced	2012-07-09 02:43:31.249488	2012-07-09 03:04:11.336528

Figure 5-15 CUSTOMER_PROFILE_HISTORY after second UPDATE of same customer

Viewing the time stamps, we can reconstruct the historical sequence for CUSTOMER_ID 10 when the status progressed from married to divorced to married again.

Deleting data from a system-period temporal table

When data is deleted from a system-period temporal table, the record (or records) is removed from the base table and then recorded in the history table. The SYS_START time in the history table is set to the SYS_START time for the record as it was in the base table, and the SYS_END time in the history table is set to the transaction start time of the DELETE statement. This start and end time assignment is done transparently by InfoSphere Warehouse and DB2.

If we consider that CUSTOMER_ID 14 closed its account with our company, then we remove it from the CUSTOMER_PROFILE table. The SQL statement in Example 5-9 accomplishes the removal.

Example 5-9 Deleting from a system-period temporal table

```
DELETE from CUSTOMER_PROFILE where CUSTOMER_ID=14;
```

In Figure 5-16, notice that the customer has been removed from the base table.

1	Male	Married	2012-07-09 03:04:11.32564	9999-12-30 00:00:00.0
10	Male	Married	2012-07-09 03:04:11.336528	9999-12-30 00:00:00.0
12	Female	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
16	Male	Married	2012-07-08 22:37:05.737418	9999-12-30 00:00:00.0
19	Female	Married	2012-07-09 02:02:09.710841	9999-12-30 00:00:00.0
20	Male	Married	2012-07-09 02:18:58.978855	9999-12-30 00:00:00.0

Figure 5-16 *CUSTOMER_PROFILE* contents after *DELETE* statement

We can further see, in Figure 5-17, that the deleted record has been captured in the history table with a *SYS_END* time set to the time that the *DELETE* command was executed by DB2. The presence of this information in the history table has many uses, including the ability to view the state of the information at a particular point in time, and the ability to recover from errors.

CUSTOMER_ID	GENDER	MARITAL_STATUS	SYS_START	SYS_END
1	Male	Single	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.239378
10	Male	Married	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.249488
14	Male	Married	2012-07-08 23:31:40.271781	2012-07-09 03:21:47.080319
10	Male	Divorced	2012-07-09 02:43:31.249488	2012-07-09 03:04:11.336528

Figure 5-17 *CUSTOMER_PROFILE_HISTORY* content after *DELETE* statement

Queries against system-period temporal tables

We are now ready to examine query behavior against system-period temporal tables. We can query data that is either current, or we can look back in time at data values for a given time or range of time.

Queries against the current valid data values

Queries that target the currently valid data in the temporal table are the most straightforward and in fact are no different than queries against non-temporal tables. There is no change in syntax and no difference in how query results are generated. Therefore, all existing reporting applications, dash boards, scripts, and use queries work without change against temporal tables without awareness of the new temporal features or logic.

We start with a query that examines the gender and age range of all customers in *CUSTOMER_PROFILE* table that are divorced (and not remarried) currently. Our query does not need to be concerned with any temporal logic and is shown in Example 5-10 on page 198.

Example 5-10 Simple select against current data in system-period temporal data

```
SELECT CUSTOMER_ID, GENDER, AGE_RANGE FROM CUSTOMER_PROFILE WHERE  
MARITAL_STATUS='Divorced';
```

Figure 5-18 shows the results of this query and reveals only two customers in our small database, both males in the same 20-35 age group.

CUSTOMER_ID	GENDER	AGE_RANGE
37	Male	20-35
57	Male	20-35

Figure 5-18 Simple system-period temporal query results against current data

Queries that look at data back in time (time travel)

For looking back at data that was valid at a point or range of time in the past, we have three different techniques that can be used in InfoSphere Warehouse 10. Note that queries that look back in time might find either previous data records that are no longer valid and have been changed, or they might also find data records that are still currently valid.

The three period specifications available include:

- ▶ **FOR SYSTEM_TIME AS OF ...** enables the user or application to query the data values valid at a specified point in time.
- ▶ **FOR SYSTEM_TIME FROM ... TO ...** enables the user or application to query the data within a specified date and time range that is *inclusive-exclusive*, that is, the date and time range *includes* or *starts* at the time in the FROM specification and proceeds up to but not including the TO specification.
- ▶ **FOR SYSTEM_TIME BETWEEN ... AND ...** provides a second approach to specify a period range that is *inclusive-inclusive*. Both date and time values provided on either side of AND are included in the query results.

Time periods: For a complete discussion about the effective use of inclusive-exclusive and inclusive-inclusive time periods, see *Best Practices: Temporal Data Management with DB2*, which is available at the following website:

<http://www.ibm.com/developerworks/data/bestpractices/temporal/index.html>

Example 5-11 on page 199 shows a simple query that looks at valid data at a point in time in the past. This query sums the number of customers in our database for each value of “MARITAL_STATUS” and displays the count. We are only interested in the numbers as of July 9, 2012.

Example 5-11 Simple SYSTEM_TIME AS OF query

```
SELECT DISTINCT MARITAL_STATUS, COUNT(MARITAL_STATUS) AS Count FROM  
CUSTOMER_PROFILE FOR SYSTEM_TIME AS OF '2012-07-09' GROUP BY MARITAL_STATUS;
```

Figure 5-19 shows the results.

MARITAL_STATUS	COUNT
Divorced	2
Married	10
Single	8

Figure 5-19 Simple query results for SYSTEM_TIME AS OF query

As an example that demonstrates a period range, we query for the various marital status values for a given customer. In other words, how many times did the customer's marital status change, including the initial status? The query in Example 5-12 accomplishes that simple task.

Example 5-12 SYSTEM_TIME period query with inclusive-exclusive range

```
SELECT MARITAL_STATUS FROM CUSTOMER_PROFILE FOR SYSTEM_TIME FROM '2012-07-08'  
TO '2012-07-10' WHERE CUSTOMER_ID=10;
```

The simple results, displayed in Figure 5-20, show that the status changed three times.

MARITAL_STATUS
Married
Divorced
Married

Figure 5-20 SYSTEM_TIME period query results with inclusive-exclusive range

If we adjust the query we can also include the valid date ranges that we might use in our application logic, for example, to offer a sales promotion to newlyweds. Or we can use the valid dates for audit purposes, for example our system might detect that the marital status changed twice in the same day which might indicate some type of fraud or application error.

Example 5-13 Date range query with sys_start and sys_end

```
SELECT MARITAL_STATUS, DATE(SYS_START) AS start_date, DATE(SYS_END) AS end_date
FROM CUSTOMER_PROFILE FOR SYSTEM_TIME FROM '2012-07-08' TO '2012-07-10' WHERE
CUSTOMER_ID=10;
```

The SQL in Example 5-13 augments the previous query with the valid start and end dates and returns the results shown in Figure 5-21. Notice that two changes were made on the same day.

MARITAL_STATUS	START_DATE	END_DATE
Married	2012-07-08	2012-07-09
Divorced	2012-07-09	2012-07-09
Married	2012-07-09	9999-12-30

Figure 5-21 Results for query with date range and valid start and end dates

Undoing a record change

Suppose we determine that the last change in marital status, as displayed in Figure 5-21, was an error or the result of a fraudulent transaction and we have to “roll back” the change to the previous value. In effect, we want to delete the “current” record in the base table and replace it with the most recent value in the history table. As derived from the sample in Figure 5-21, however, a rollback will result in the current marital status for this customer (10) to be Divorced, not Married.

So instead we begin by looking at the full **TIMESTAMP** values for the result in Figure 5-21. We can do this by dropping the “DATE()” functions in the SQL query in Example 5-13.

We show the results of the modified query in Figure 5-22.

MARITAL_STATUS	START_DATE	END_DATE
Married	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.249488
Divorced	2012-07-09 02:43:31.249488	2012-07-09 03:04:11.336528
Married	2012-07-09 03:04:11.336528	9999-12-30 00:00:00.0

*Figure 5-22 Query results with full **TIMESTAMP** values for start and end time*

Notice that the **END_DATE** time stamp on line 1 matches exactly the **START_DATE** time stamp on line 2. This is what we mean when we refer to the value of the inclusive-exclusive period having no gaps.

Also note that the `END_DATE` on row 2 is the same as the `START_DATE` on row 3. We know that row 3 refers to the currently “valid” record (by virtue of the indefinite `END_DATE`), and this is also the record we want to remove and roll back to the value of row 2.

Our two-step process consists of these tasks:

1. Remove the “current” row.
2. Insert the most recent row in the history table into the base table as the new current row.

This process essentially performs a full row revert. In our case, we know it is the marital status that is incorrect, so we can simply update the current record with the correct marital status based on the previous valid value. The SQL in Example 5-14 accomplishes this goal.

Example 5-14 Replace the current value with the most recent value in the history table

```
UPDATE CUSTOMER_PROFILE prof
SET prof.MARITAL_STATUS=(SELECT hist.MARITAL_STATUS FROM
CUSTOMER_PROFILE_HISTORY hist, CUSTOMER_PROFILE prof
WHERE prof.SYS_START = hist.SYS_END AND prof.CUSTOMER_ID=10)
WHERE CUSTOMER_ID=10;
```

The record in the base table for Customer 10 has the marital status updated to the most previous value of marital status in the history table for Customer 10.

Rerunning the prior query for the history of values for this customer reveals that the new valid marital status is set back to Divorced though the history table retains the history of the erroneous change. This is shown in Figure 5-23.

You can see the currently valid row (the last row) contains the corrected marital status and has the end date set to indefinite.

MARITAL_STATUS	START_DATE	END_DATE
Married	2012-07-08 22:37:05.737418	2012-07-09 02:43:31.249488
Divorced	2012-07-09 02:43:31.249488	2012-07-09 03:04:11.336528
Married	2012-07-09 03:04:11.336528	2012-07-09 14:00:26.545541
Divorced	2012-07-09 14:00:26.545541	9999-12-30 00:00:00.0

Figure 5-23 Updated results after reverting to prior marital status

Note: There is a script `~/sql1lib/samples/clp/temporal_revert.db2` that creates a stored procedure called `REVERT_TABLE_SYSTEM_TIME`. This stored procedure provides a general technique to revert any row or rows in a base table to any rows in the history based on a `TIMESTAMP` value. Details of the script usage are in the comment section of the script.

5.2.3 Temporal data management with business time

Business time refers to the time, either a simple date or down to the granularity of a time stamp, that has particular meaning in a business context. The date on which a legal contract goes into effect or the date that a sale coupon expires are both examples of business time.

Business time might or might not be the same as the corresponding system time for an event, and the two can be completely unrelated. For example, a change-of-address card received by the post office might be entered into the system on June 1 (system time); however, the actual change of address might not go into effect until July 1 (business time).

Many business applications, queries, and activities are most interested in business time handling. Generally speaking, business processes and applications require the ability to identify the time period during which some condition is valid or effective. They need the business start and business end dates (or dates and times).

To support the business logic that goes with managing the effective business start and end times, InfoSphere Warehouse 10.1 (and DB2 10.1) have been enhanced with new features to do just that. We examine these new features in this section.

To help illustrate the new features, we again use part of the database schema for the Customer Insight Warehouse Pack. In this case, we use the tables associated with tracking the involvement of a customer in a loyalty program.

Many retailers and other businesses use loyalty programs to retain and incent their customers by awarding them with points for each purchase. These points can be exchanged directly for goods or may go towards a discount or some other advantage. Airline frequent flyer programs are typical examples of how these plans work.

In our sample database, as in many actual programs, there are different grades or levels that offer increasing amounts of reward for customers who have the most purchases. Thus, “better” customers are rewarded with better loyalty program benefits.

In our sample database, we have three levels of loyalty program with the following benefits for each:

- ▶ **No Plan:** At this lowest level are customers who are not enrolled in any loyalty plan and therefore, have no plan benefits.
- ▶ **Loyalty Club:** The lowest plan level. Members earn a point for every dollar spent and can exchange points for vouchers that can be used to purchase goods or services offered by our company.
- ▶ **Gold Club:** The highest plan level. Members earn points for every dollar spent with an additional 50% bonus points. They redeem those points in the same way as the Loyalty Club level.

A great deal of interesting activity and analysis can take place in the context of the operational data warehouse when we use the business time data management and query features in InfoSphere Warehouse 10.1.

For the examples that follow, consider the simple table of information shown in Figure 5-24.

Customer	Loyalty_Scheme	Bonus_Pct
Albert Johnson	Customer Loyalty Club	100
Jos Smith	No Loyalty Scheme	0
Linda Moore	Customer Gold Club	150
Melissa Smith	Customer Loyalty Club	100

Figure 5-24 Customer Loyalty Program information

Even with this simple table of information, we see how we can perform significant and useful analysis in a data warehousing environment. For example, we can perform the following analysis:

- ▶ Track customer loyalty points and determine how many they have redeemed in a given time period.
- ▶ Compare purchasing behaviors of customers in the loyalty programs with those not enrolled.
- ▶ Compare the costs incurred by the loyalty program for both administration and point redemption with the increased revenue attributed to the program.

If we add business time and associated logic to Figure 5-24, we can perform many other useful activities related to the loyalty program such as:

- ▶ Compare a single customer's behavior during periods when the customer is enrolled or not enrolled in a loyalty program.

- ▶ Analyze the effectiveness of allowing a customer to participate in a premium loyalty program for a fixed period of time.
- ▶ Allow for loyalty program terms that can vary over time, and correctly calculate metrics related to the program such as accrued points and member revenue contribution.

In InfoSphere Warehouse 10.1 (and DB2 10.1), the table is augmented with a business start column and business end column that tracks the effective start and end dates during which the information in the table record is valid or in force.

This is similar to the system start and end columns in the system-time temporal tables described in previous sections. However, in the case of business time, we are more interested in the dates in a business context as opposed to the moment the data was entered or changed in the database. Also, with business time temporal tables, we do not use a history table as in the case of system time temporal data. All history is maintained in the single table unless the record is fully deleted.

The augmented table with business time temporal handling is shown in Figure 5-25 with newly added effective start and end dates. For our small sample, we show four customers, most of whom are enrolled in one of the programs.

The effective start date reflects the date on which their enrollment became active (or simply the date they became a customer if not enrolled in a program). The effective end date for all shown is December 30, 9999 or in effect, indefinite. This means their current enrollment status is not set to expire at any given time in the future.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Loyalty Club	100	2011-09-15	9999-12-30
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30
Linda MM	Customer Gold Club	150	2012-03-12	9999-12-30
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30

Figure 5-25 CUST_LOYALTY table enhanced with business time columns

We now consider the details of how to implement and manipulate the business time temporal tables in InfoSphere Warehouse 10.1.

Application-period terminology: Consistent with the ISO SQL 2011 terminology, the term *application-period* is used to indicate temporal tables based on business time as distinct from system-period temporal tables.

Creating an application-period temporal table

Similar to the system-period temporal tables, there are multiple techniques that can be employed to prepare a table in DB2 for use as an application-period temporal table. We consider the process in InfoSphere Warehouse Design Studio first, then we look at doing it with SQL that has been coded manually.

Using Design Studio tooling

To create an application-period temporal table in InfoSphere Warehouse using Design Studio, simply define the table as usual and then select **Business time period** under the “Temporal attributes” heading in the table Properties tab as shown in Figure 5-26.

Name	Prim	Data Type	Length	Scale	Not Null	Generated	Default	Period
Customer	<input type="checkbox"/>	VARCHAR	256		<input type="checkbox"/>	<input type="checkbox"/>		
Loyalty_Scheme	<input type="checkbox"/>	VARCHAR	40		<input type="checkbox"/>	<input type="checkbox"/>		
Bonus_Pct	<input type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input type="checkbox"/>		
Effective_Start_Date	<input type="checkbox"/>	DATE			<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME begin
Effective_End_Date	<input type="checkbox"/>	DATE			<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME end
Temporal attributes:								
<input type="checkbox"/> System time period <input checked="" type="checkbox"/> Business time period								

Figure 5-26 Design Studio properties for application-period time temporal table definition

Notice several items in this figure:

- ▶ The `Effective_Start_Date` and `Effective_End_Date` columns are automatically generated by Design Studio. These are given tool-defined generic names, and the designer or DBA can simply change them to names with more meaning for the business context. These columns may either be DATE types or TIMESTAMP types, depending on the application-specific requirements for these fields and how they will be used.
- ▶ The “Generated” flag is *not* checked for application-period temporal tables. Different from the system-period temporal feature discussed previously, DB2 has no way of auto-generating these values, because they are business times and dates as opposed to database transaction times. The DBA or application have to provide these values when data is inserted or updated. This is what we expect for business time.
- ▶ The “period” attribute for the application-period date columns defines the `Effective_Start_Date` and `Effective_End_Date` as BUSINESS_TIME “begin” and “end” columns respectively. This information is coded into the SQL DDL to create the table (or alter an existing table).

- The business period start and end columns are defined as “Not Null”, but no default values are provided. The application that populates these tables (or updates) must always provide an explicit value.

As with the system-period temporal tables, all of these attributes that specify the details of the application-period temporal table are automatically generated by Design Studio. They are used to generate the appropriate SQL DDL statements to create or alter the table to make it support business time data management and query logic.

Tip: You can also simplify the creation of application-period temporal tables in Design Studio by right-clicking the schema in the project explorer and selecting **Add Data Object** → **Application-period Temporal Table**. This creates a skeleton table definition with the two application-period start and end columns predefined.

This technique and results are shown in Figure 5-27 and Figure 5-28. Notice the default column names for the start and end times. These can be changed. Also, the data type for the columns can be changed from `TIMESTAMP` to `DATE` if time-of-day granularity is not required for the business application.

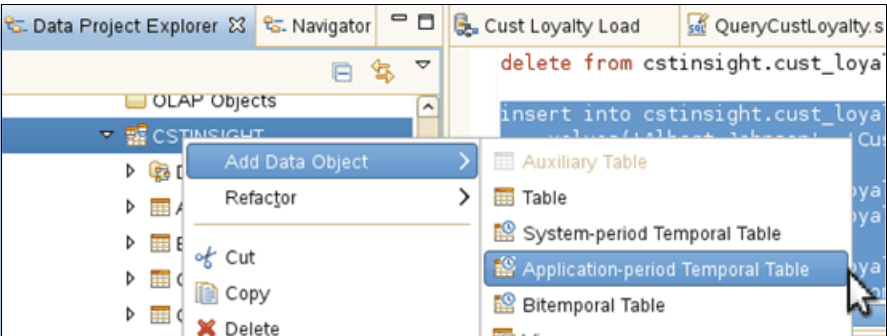


Figure 5-27 Create an application-period temporal table in Design Studio with shortcut

Figure 5-28 displays the results.

Name	Primary Key	Do	Data Type	Length	Scale	Not Null	Generated	Def	Period
APPLICATION_START_TIME1	<input type="checkbox"/>		TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME begin
APPLICATION_END_TIME1	<input type="checkbox"/>		TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME end

Figure 5-28 Results of creating an application-period temporal table in Design Studio

Notice that Design Studio adds new business start time and business end time columns and gives them the default names `APPLICATION_START_TIME1` and

APPLICATION_END_TIME1. You can change these column names by the modeler to provide more meaningful names in the business context or to follow corporate naming conventions.


Either of these techniques in Design Studio creates an application-period temporal table that appears in the overview diagram as shown in Figure 5-29. Notice the time icon in the table name (); this graphic is used consistently throughout the graphical tooling to identify temporal tables of any type (business or system-period).



Figure 5-29 Application-period temporal table created in Design Studio

Altering an existing table to support application-period temporal

It is possible to modify an existing table to support application-period temporal data management and queries using Design Studio. Simply select the table in the Data Project Explorer and view the column properties for the table. Select the “Business time period” check box (as when the table was created originally as h described in “Using Design Studio tooling” on page 205). This action creates the two columns for business start and end and also generates the proper DDL to enable the table for application-period temporal logic; see Figure 5-30.

The generated column names can be changed to the names you want, and the table is deployed as described in the next section.

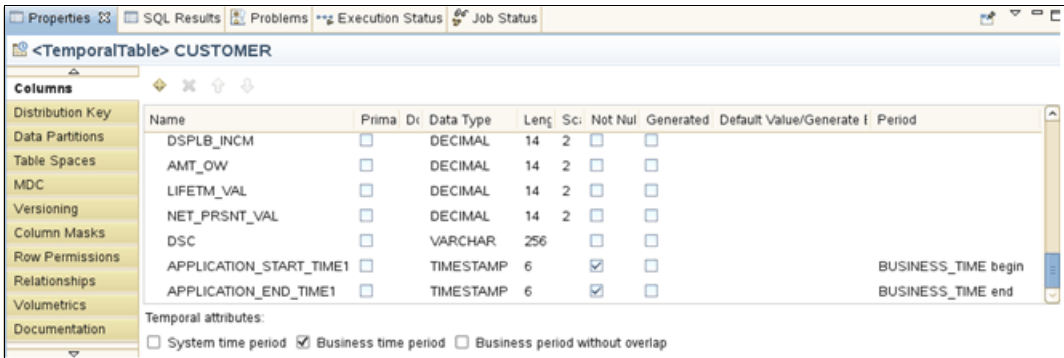


Figure 5-30 Altering a table to support application-period temporal data

It is also possible to convert an existing table into an application-period temporal table and use pre-existing columns that capture business start and end times. The Period column in the Design Studio table properties (shown in Figure 5-30 on page 207) is an “active” column. To select an existing column for either BUSINESS_TIME begin or end, follow these steps:

1. In the column property row for either of the default columns generated by Design Studio, select the value in the Period column for BUSINESS_TIME begin (or end) to see a drop-down list box.
2. Select the Blank (or empty) value in the drop-down list, effectively removing the designation for that column.
3. In the property row for the existing column that specifies the business start time (or end time, respectively), select the **Period** column, and select **BUSINESS_TIME begin** (or end). This action associates the existing column with the business start time for this application-period temporal table.
4. Select the Design Studio-generated APPLICATION_START_TIME or APPLICATION_END_TIME columns and delete it by clicking the red “X” symbol.

In this way, you can convert an existing table that already contains business start and end columns into an application-period temporal table. Applications that are already using this table can continue to use the table as is with existing application-specific logic. The logic might also be altered (and new applications developed) to benefit from the new temporal features built into DB2 and InfoSphere Warehouse 10.1.

Deploying the table to the physical database with Design Studio

After the application-period temporal table has been fully specified in the Data Project Explorer in Design Studio, it can be manipulated within the tooling similar to any other common table. Thus, it can be used in SQL Warehousing data flows, data mining flows, Cubing Services cube models, and so on.

The new table (or changed table) can be deployed to the physical database using any technique to deploy a table to DB2 for any other purpose. For example, the *compare-and-synch* capability in Design Studio is a simple way to propagate a new table from design to DB2.

Creating the application-period temporal table manually with SQL

It is also possible to create the application-period temporal table by using SQL as shown in Example 5-15.

Example 5-15 Manually coded SQL to create the application-period temporal table

```
CREATE TABLE "CSTINSIGHT"."CUST_LOYALTY" (  
    "Customer" VARCHAR(256),
```

```

        "Loyalty_Scheme" VARCHAR(40),
        "Bonus_Pct" INTEGER,
        "Effective_Start_Date" DATE NOT NULL,
        "Effective_End_Date" DATE NOT NULL,
        PERIOD_BUSINESS_TIME("Effective_Start_Date", "Effective_End_Date")
    )
    DATA CAPTURE NONE
    IN "USERSPACE1"
    COMPRESS YES ADAPTIVE;

```

Note key aspects of the SQL example:

- ▶ The two DATE type columns (“Effective_Start_Date” and “Effective_End_Date”) are simple DATE column specifications. There is no additional syntax as was seen for system-period tables.
- ▶ Remember, the column names can be anything that has meaning to the business context and can be DATE or TIMESTAMP types.
- ▶ The start and end times appear within the “..PERIOD_BUSINESS_TIME(..)” clause. This specification identifies them in order as to which is the business start and business end. This is necessary for DB2 to provide the proper logic when dealing with application-period temporal tables.

For tables with a primary key, such as a dimension or lookup table in an operational data warehouse scenario, it is important that uniqueness integrity not be compromised by the use of the application-period temporal feature. In this case it is important that no overlap be allowed in the specified time period. If overlap occurs, then we have a situation with two valid rows for the same primary key, which is not acceptable.

You can specify a constraint to disallow overlap with the SQL shown in Example 5-16.

Example 5-16 SQL to disallow time period overlap in the application-period table

```

ALTER TABLE "CSTINSIGHT"."CUST_LOYALTY" ADD CONSTRAINT "CUST_LOYALTY_PK"
PRIMARY KEY
("Customer", BUSINESS_TIME WITHOUT OVERLAPS);

```

When the WITHOUT OVERLAPS clause is used in either the ALTER TABLE or included in the CREATE TABLE syntax, it instructs DB2 to logically extend the primary key (for example, the CUSTOMER column) with the business time information. This enables DB2 to ensure that there are no duplicate records. In other words, there are no records with the same primary key value and overlapping time periods.

5.2.4 Application-period temporal tables

When the application-period temporal tables have been defined and deployed to DB2 using InfoSphere Warehouse 10.1, we can examine how to manipulate and query these tables. We continue to use our simple example table tracking our customers' use of our loyalty program. We begin by looking at the table insert and then move onto other, more interesting, actions.

Inserting data into an application-period temporal table

Different from the system-period temporal tables discussed previously in this chapter, row insertion into an application-period temporal table requires that the start and end time values be provided. Because these only have meaning in the business context and not the database context, DB2 cannot simply generate these values. They must be provided by the application or person inserting the data.

Recall the simple example table of customers and their enrollment status in various levels of the customer loyalty programs, shown here in Figure 5-31 for your convenience.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Loyalty Club	100	2011-09-15	9999-12-30
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30
Linda MM	Customer Gold Club	150	2012-03-12	9999-12-30
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30

Figure 5-31 Sample Customer Loyalty program table with temporal columns

These records were inserted into the table using the SQL INSERT statements displayed in Example 5-17.

Example 5-17 INSERT statements to populate an application-period temporal table

```
insert into cstinsight.cust_loyalty
  values('Albert Johnson', 'Customer Loyalty Club', 100, '2011-09-15',
'9999-12-30');

insert into cstinsight.cust_loyalty
  values('Joe Smith', 'No Loyalty Scheme', 0, '2012-06-22', '9999-12-30');

insert into cstinsight.cust_loyalty
  values('Linda MM', 'Customer Gold Club', 150, '2012-03-12', '9999-12-30');

insert into cstinsight.cust_loyalty
```

```
values('Melissa Smith', 'Customer Loyalty Club', 100, '2011-11-01',  
'9999-12-30');
```

Inserting and managing multiple time periods

To be really useful, the application-period temporal table can manage multiple rows for the same customer covering different time periods. This allows us, for example, to maintain a history for each customer of which plans they participated in and for what time periods. Having this information available in our data warehouse allows us to perform important analysis of the impact of these plans on both the cost and benefit sides of the equation to determine the effectiveness of the plans at driving increased revenue and retaining good customers.

Simple inserts into application-period temporal tables

The simplest form of INSERT is for the user or the application to provide the records and dates for each time period to be captured. For example, here we add a new customer named Jane Doe. Jane became a customer effective May 20, 2012 but did not immediately join a loyalty program. Starting July 1, 2012, she enrolled in the basic loyalty program. To insert the full history into our CUST_LOYALTY table using SQL INSERT, we use the statements in Example 5-18.

Example 5-18 INSERT for multiple periods for the same customer without overlap

```
insert into cstinsight.cust_loyalty  
  values('Jane Doe', 'No Loyalty Scheme', 0, '2012-05-20', '2012-07-01');  
insert into cstinsight.cust_loyalty  
  values('Jane Doe', 'Customer Loyalty Club', 100, '2012-07-01',  
'9999-12-30');
```

If we then look at the table contents as shown in Figure 5-32, we can clearly see the history of Jane Doe's involvement in the loyalty program.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Loyalty Club	100	2011-09-15	9999-12-30
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30
Linda MM	Customer Gold Club	150	2012-03-12	9999-12-30
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30
Jane Doe	No Loyalty Scheme	0	2012-05-20	2012-07-01
Jane Doe	Customer Loyalty Club	100	2012-07-01	9999-12-30

Figure 5-32 CUST_LOYALTY table after inserting multiple rows for Jane Doe

Note the end date of her first record is July 1, 2012", and the start date of her second record is also July 1, 2012. Keep in mind the discussion from the

beginning of this chapter about inclusive-exclusive time periods as implemented in DB2. Constant usage of inclusive-exclusive keeps us from getting gaps or overlaps in our data in the temporal table.

Implications: Consider the implications of application-period temporal tables and primary keys.

In our example, if the CUSTOMER column is the declared primary key, then the business time period provides an implicit extension to the primary key uniqueness constraint. Thus DB2 will allow, for example, multiple rows to exist for the CUSTOMER “Jane Doe” so long as there are no overlapping time periods.

Consider a second example where we have period overlap in a table that has been defined with the constraint against this kind of overlap. Assume we have the insert statements shown in Example 5-19.

Example 5-19 Example INSERT with overlap causing SQL error

```
insert into cstinsight.cust_loyalty
  values('Ray BB', 'No Loyalty Scheme', 0, '2012-04-15', '2012-06-30');
insert into cstinsight.cust_loyalty
  values('Ray BB', 'Customer Loyalty Club', 100, '2012-06-01', '9999-12-30');
```

What we see is that Ray joined us on April 15, 2012. On June 1, 2012 Ray joined our loyalty program. However, the end date for the first record indicating when he ceased to not have a loyalty program is after the start date that the new program was in force. This will violate the integrity constraints on this table and will give the error, shown in Example 5-20, when it tries to execute the second INSERT statement.

Example 5-20 Error generated by overlapping time period on INSERT

```
insert into cstinsight.cust_loyalty
  values('Ray BB', 'Customer Loyalty Club', 100, '2012-06-01', '9999-12-30')
```

One or more values in the INSERT statement, UPDATE statement, or foreign key update caused by a DELETE statement are not valid because the primary key, unique constraint or unique index identified by "1" constrains table "CSTINSIGHT.CUST_LOYALTY" from having duplicate values for the index key..
SQLCODE=-803, SQLSTATE=23505, DRIVER=3.63.108

Updating application-period tables for a portion of business time

A more interesting scenario involves updates to existing records that cover only a portion of the business time. These type of updates have the effect of “splitting” a single row of data into two or more.

For a first example of this, let us consider the customer Albert Johnson who has been a customer in the basic loyalty program since September 2011 and now wants to upgrade to the Gold program. He is approved for the upgrade and it is scheduled to become effective on September 1, 2012. To accomplish this update, we write a traditional DB2 UPDATE statement but include a new FOR PORTION of BUSINESS_TIME clause so that the time period will be handled properly. To maintain the history of Albert Johnson prior to the change, DB2 updates both the existing row with a new end date and inserts a new row with the new information, new start date, and indefinite end date. Example 5-21 shows the UPDATE statement.

Example 5-21 Update FOR PORTION OF BUSINESS_TIME with new record

```
update cstinisight.cust_loyalty
  FOR PORTION OF BUSINESS_TIME FROM '2012-09-01' TO '9999-12-30'
  SET Loyalty_Scheme='Customer Gold Club', Bonus_Pct=150
  WHERE Customer='Albert Johnson';
```

Note that the date range is inclusive-exclusive to avoid overlap or gaps. The listing in Figure 5-33 displays the records for Albert Johnson.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Loyalty Club	100	2011-09-15	2012-09-01
Albert Johnson	Customer Gold Club	150	2012-09-01	9999-12-30

Figure 5-33 Split records for Albert Johnson after UPDATE

Note that the “old” record now has an effective end date that matches the start date of the customer’s new loyalty plan. The plan information is updated as well.

Here is an additional example that splits one record into three. Recall that Linda MM has been a member of the Gold club for some time; see Figure 5-34.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	9999-12-30

Figure 5-34 Loyalty program status of Linda MM before any changes

Linda’s account manager has decided to offer her a special promotion whereby she will get double award points (Bonus_Pct = 200) for a 30-day trial period.

After the trial period (which will begin on August 1, 2012), management will review her buying patterns and assess the impact of the promotion to determine whether to extend it into a new elite plan category.

Example 5-22 shows the UPDATE statement to accomplish this.

Example 5-22 UPDATE customer record for a portion of business time

```
UPDATE "CSTINSIGHT"."CUST_LOYALTY"
  FOR PORTION OF BUSINESS_TIME FROM
    '2012-08-01' TO DATE('2012-08-01') + 30 DAYS
  SET "Bonus_Pct" = 200
  WHERE "Customer"='Linda MM';
```

Notice the use of the following function:

DATE('2012-08-01') + 30 DAYS

This is a small example illustrating that temporal functions can be used in place of DATE or TIMESTAMP literals to enhance business logic and to simplify programming.

Reviewing all of the records for Customer Linda MM reveals the impact of this UPDATE statement. In Figure 5-35, we can see that the impact of the UPDATE statement has been to split the single record into three.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30

Figure 5-35 Results of UPDATE covering a “Portion” of business time

In the original record, Linda MM was a member of the “Customer Gold Club” effective March 12, 2012, earning bonus points equal to 150% of every dollar spent. After the UPDATE statement to introduce the 30-day promotion, there are three records for the same customer account for Linda MM reflecting the changes to her account (the increase in bonus points) and the period for which the change is valid or effective, namely 30 days. Figure 5-36 on page 215 depicts this graphically.

Note: Recall that the Effective_End_Date is *exclusive*, so in our example the bonus point percentage value is 200% up through August 30; that is, *until* August 31, but not including August 31.

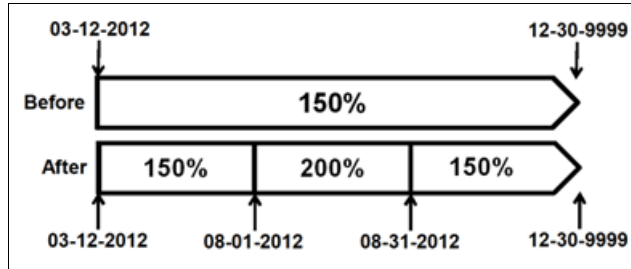


Figure 5-36 BONUS_PCT value over time after UPDATE for Portion of Business_Time

Deleting rows from application-period temporal tables

Rows can be deleted either completely from the temporal table (for example, all records for a given primary key) or a portion of the row or rows can be removed for a subset of the business period (FOR PORTION OF BUSINESS_TIME). We consider both cases, and assume the table has the records shown in Figure 5-37. Notice that Customer “Jane Doe” has two records.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Gold Club	150	2012-09-01	9999-12-30
Albert Johnson	Customer Loyalty Club	100	2011-09-15	2012-09-01
Jane Doe	No Loyalty Scheme	0	2012-05-20	2012-07-01
Jane Doe	Customer Loyalty Club	100	2012-07-01	9999-12-30
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30

Figure 5-37 CUST_LOYALTY table before DELETE

We consider a simple DELETE operation first, shown in Example 5-23.

Example 5-23 Simple DELETE from application-period temporal table

```
DELETE FROM cstinsight.cust_loyalty
WHERE "Customer"='Jane Doe';
```

This simple DELETE with no predicate on the business-time period deletes all records for the primary key “Jane Doe”. Figure 5-38 displays the results.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Albert Johnson	Customer Gold Club	150	2012-09-01	9999-12-30
Albert Johnson	Customer Loyalty Club	100	2011-09-15	2012-09-01
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30

Figure 5-38 Results of simple DELETE from application-period temporal table

Notice there are no longer any records corresponding to “Jane Doe”.

The next example illustrates the use of FOR PORTION OF BUSINESS_TIME in the DELETE statement, as shown in Example 5-24. Here, we want to shorten the promotion period offered to CUSTOMER “Linda MM” from 30 days to 15 days.

Example 5-24 DELETE from business-time temporal table for portion of period

```
DELETE FROM cstinsight.cust_loyalty
  FOR PORTION OF BUSINESS_TIME
  FROM '2012-08-16' TO '2012-08-31'
 WHERE "Customer"='Linda MM';
```

This causes the record update seen in Figure 5-39.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-16
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30

Figure 5-39 Results of DELETE for portion of business time

There are several points to note in these results:

- The period of the 200% promotion is changed from 30 days to 15 days, as seen in Figure 5-40 on page 217. Recall that the end date is always *exclusive* in temporal tables.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-16

Figure 5-40 Shortened time period after DELETE

- There is now a gap in the time period for CUSTOMER “Linda MM”. It can be seen in Figure 5-41 that the period of the 200% Bonus Point promotion runs from August 1, 2012 until August 16, 2012 (15 days).

The next period, with the standard 150% Bonus Point policy, starts as it did prior to the DELETE on August 31, 2012 running indefinitely (December 30, 9999). Thus, there is a 15-day gap from August 16 until August 31, 2012 where there is apparently no policy in effect.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-16
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30

15-day gap

Figure 5-41 Period gap caused by DELETE of portion of business-time

This type of gap can cause errors in the business processing and any applications that use this temporal table data. It is important that application logic be implemented to detect and avoid having gaps of this kind occur.⁴

The SQL statements in Example 5-25 will detect and report on gaps in the CUST_LOYALTY table. Database triggers can be implemented to detect and prevent gaps automatically whenever rows are inserted, updated, or deleted.

Example 5-25 Gap detection SQL for application-period temporal tables

```
SELECT "Customer",
       "Effective_Start_Date" AS gap_start,
       "Effective_End_Date" AS gap_end
FROM
  (SELECT "Customer", "Effective_Start_Date", "Effective_End_Date",
         MIN("Effective_End_Date") OVER (PARTITION BY "Customer" ORDER BY
         "Effective_Start_Date"
         ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING)
         AS previous_end
   FROM cstinsight.cust_loyalty)
WHERE "Effective_Start_Date" > previous_end;
```

⁴ For more information about detecting gaps between periods in temporal tables, refer to *Best Practices: Temporal Data Management with DB2*, which is available at the following website:
<http://www.ibm.com/developerworks/data/bestpractices/temporal/index.html>

Queries against application-period temporal tables

We can now examine query behavior against application-period temporal tables. With application-period tables, we can query the past, present, and future values for a given time or range of time.

Query with no business time context (non-temporal)

Any non-temporal or regular SELECT statement against the application-period temporal table is processed normally by DB2. Example 5-26 returns all rows for CUSTOMER “Linda MM” as shown.

Example 5-26 Simple non-temporal query against application-period temporal table

```
Select * from cstinsight.cust_loyalty
  WHERE "Customer" = 'Linda MM'
  ORDER BY "Effective_Start_Date";
```

The results are displayed in Figure 5-42; the gap from the previous example has been corrected.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30

Figure 5-42 Non-temporal query results for application-period temporal table

Queries with business time context

More interesting for application-period temporal tables are queries that contain business time context.

The following forms are used for this type of query:

- ▶ FOR BUSINESS_TIME AS OF ...
- ▶ FOR BUSINESS_TIME FROM ... TO ...
- ▶ FOR BUSINESS_TIME BETWEEN ... AND ...

We consider examples of each form to illustrate their meaning and usage.

First, Example 5-27 shows a business-time query for a particular date (can also be a time stamp). This query shows the record value in force on August 11, 2012.

Example 5-27 FOR BUSINESS_TIME AS OF ... query example

```
Select * from cstinsight.cust_loyalty
  FOR BUSINESS_TIME AS OF '2012-08-11'
  WHERE "Customer" = 'Linda MM'
```

```
ORDER BY "Effective_Start_Date";
```

The results are shown in Figure 5-43.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31

Figure 5-43 Results of *BUSINESS_TIME AS OF ... query*

Next, Example 5-28 shows a business-time query for a range of the form “FROM ... TO ...”. Keep in mind that the period range is given in the inclusive/exclusive form. The query is looking for the records in force in the period from July 15, 2012 to August 16, 2012, or roughly a one-month period starting July 15.

Example 5-28 *FOR BUSINESS_TIME FROM ... TO .. query example*

```
Select * from cstinsight.cust_loyalty
  FOR BUSINESS_TIME FROM '2012-07-15' TO '2012-08-16'
 WHERE "Customer" = 'Linda MM'
 ORDER BY "Effective_Start_Date";
```

The results are shown in Figure 5-44.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31

Figure 5-44 Results of *BUSINESS_TIME FROM... TO ... query*

Finally, Example 5-29 shows a business-time query for a range of the form “BETWEEN ... AND ...”. The query asks for the loyalty program information for the year 2013.

Example 5-29 *FOR BUSINESS_TIME BETWEEN ... AND .. query example*

```
Select * from cstinsight.cust_loyalty
  FOR BUSINESS_TIME BETWEEN '2013-01-01' AND '2014-01-01'
 WHERE "Customer" = 'Linda MM'
 ORDER BY "Effective_Start_Date";
```

The results are shown in Figure 5-45.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30

Figure 5-45 Results of FOR BUSINESS_TIME BETWEEN ... AND .. query

5.2.5 Bitemporal tables

It is possible to define temporal tables for use as both system-period and application-period. These are known as *bitemporal* tables. These are used in cases where you have to capture both the *system time* at which a data record is entered or valid in the database system, and the *business time* for which the record is valid or in force. As a simple example, you might want to track the time period that customer loyalty program terms are valid, and also want to keep track of when changes to a customer's loyalty program policy are made.

Creating the bitemporal table

As with system-period and application-period temporal tables, the easiest way to create bitemporal tables is with InfoSphere Warehouse 10.1 Design Studio. Simply select the schema that you want in the Data Project Explorer, right-click and choose the menu option **Add Data Object** → **Bitemporal Table** as shown in Figure 5-46.

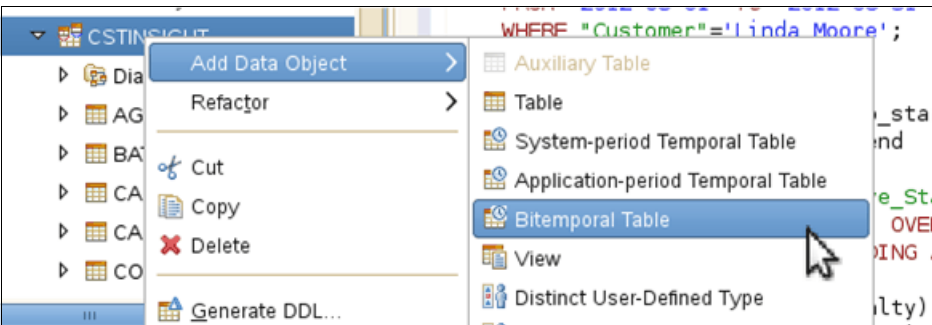


Figure 5-46 Creating a bitemporal table in Design Studio

This creates a skeleton table structure with predefined column definitions for both system-period and application-period start and end times, as shown in

Figure 5-47. We added a data column and primary key column for completeness and chose to restrict overlap, as shown.

Name	Primary Key	Data Type	Length	Scale	Not Null	Generated	Default Value/Generate	Period
PK_COLUMN	<input checked="" type="checkbox"/>	INTEGER			<input checked="" type="checkbox"/>	<input type="checkbox"/>		
DATA_COL	<input type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input checked="" type="checkbox"/>		
SYSTEM_START_TIME1	<input type="checkbox"/>	TIMESTAMP	12		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AS ROW BEGIN	SYSTEM_TIME begin
SYSTEM_END_TIME1	<input type="checkbox"/>	TIMESTAMP	12		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AS ROW END	SYSTEM_TIME end
TRANS_ID1	<input type="checkbox"/>	TIMESTAMP	12		<input type="checkbox"/>	<input checked="" type="checkbox"/>	AS TRANSACTION START	
APPLICATION_START_TIME1	<input type="checkbox"/>	TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME begin
APPLICATION_END_TIME1	<input type="checkbox"/>	TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>		BUSINESS_TIME end

Temporal attributes:

☒ System time period ☒ Business time period ☒ Business period without overlap

Figure 5-47 Columns generated for bitemporal tables in Design Studio

The Design Studio also automatically creates the history table for the bitemporal table with all of the appropriate columns included in the definition as shown in Figure 5-48.

Name	Primary Key	Data Type	Length	Scale	Not Null	Generated
PK_COLUMN	<input type="checkbox"/>	INTEGER			<input checked="" type="checkbox"/>	<input type="checkbox"/>
DATA_COL	<input type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input type="checkbox"/>
SYSTEM_START_TIME1	<input type="checkbox"/>	TIMESTAMP	12		<input checked="" type="checkbox"/>	<input type="checkbox"/>
SYSTEM_END_TIME1	<input type="checkbox"/>	TIMESTAMP	12		<input checked="" type="checkbox"/>	<input type="checkbox"/>
TRANS_ID1	<input type="checkbox"/>	TIMESTAMP	12		<input type="checkbox"/>	<input type="checkbox"/>
APPLICATION_START_TIME1	<input type="checkbox"/>	TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>
APPLICATION_END_TIME1	<input type="checkbox"/>	TIMESTAMP	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 5-48 History table generated by Design Studio for bitemporal

As with pure system-period or application-period tables, all of the generated column names can be renamed in Design Studio per the developer's requirements or preferences.

Example 5-30 shows the SQL required to create these tables and designate them to DB2 for bitemporal use.

Example 5-30 SQL to create a bitemporal table manually

```
CREATE TABLE "CSTINSIGHT"."Table2" (
  "PK_COLUMN" INTEGER NOT NULL,
  "DATA_COL" INTEGER,
  "SYSTEM_START_TIME1" TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW
BEGIN ,
  "SYSTEM_END_TIME1" TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END ,
  "TRANS_ID1" TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID ,
```

```

        "APPLICATION_START_TIME1" TIMESTAMP NOT NULL,
        "APPLICATION_END_TIME1" TIMESTAMP NOT NULL,
        PERIOD SYSTEM_TIME("SYSTEM_START_TIME1", "SYSTEM_END_TIME1"),
        PERIOD BUSINESS_TIME("APPLICATION_START_TIME1", "APPLICATION_END_TIME1")
    )
    DATA CAPTURE NONE
    COMPRESS NO;

CREATE TABLE "CSTINSIGHT"."Table2_HIST1" (
    "PK_COLUMN" INTEGER NOT NULL,
    "DATA_COL" INTEGER,
    "SYSTEM_START_TIME1" TIMESTAMP(12) NOT NULL,
    "SYSTEM_END_TIME1" TIMESTAMP(12) NOT NULL,
    "TRANS_ID1" TIMESTAMP(12),
    "APPLICATION_START_TIME1" TIMESTAMP NOT NULL,
    "APPLICATION_END_TIME1" TIMESTAMP NOT NULL
)
DATA CAPTURE NONE
COMPRESS NO;

ALTER TABLE "CSTINSIGHT"."Table2" ADD VERSIONING USE HISTORY TABLE
"CSTINSIGHT"."Table2_HIST1";

ALTER TABLE "CSTINSIGHT"."Table2" ADD CONSTRAINT "Table2_PK" PRIMARY KEY
("PK_COLUMN", BUSINESS_TIME WITHOUT OVERLAPS);

```

Using bitemporal tables

In this section we provide examples of using bitemporal tables in a DB2 and InfoSphere Warehouse 10.1 solution. Here we have created a table for bitemporal analysis in our customer loyalty program scenario. Figure 5-49 shows a summary of the column information.

Name	Primary Key	Data Type
Customer	<input checked="" type="checkbox"/>	VARCHAR
Loyalty_Scheme	<input type="checkbox"/>	VARCHAR
Bonus_Pct	<input type="checkbox"/>	INTEGER
Effective_Start_Date	<input type="checkbox"/>	DATE
Effective_End_Date	<input type="checkbox"/>	DATE
SYSTEM_START_TIME	<input type="checkbox"/>	TIMESTAMP
SYSTEM_END_TIME	<input type="checkbox"/>	TIMESTAMP
TRANS_ID1	<input type="checkbox"/>	TIMESTAMP

Figure 5-49 Bitemporal table columns for CUST_LOYALTY

Figure 5-50 shows the history table for this bitemporal table. As expected, the layout of the HISTORY table is identical to the base bitemporal table.

Name	Prima	Don	Data Type
Customer	<input type="checkbox"/>		VARCHAR
Loyalty_Scheme	<input type="checkbox"/>		VARCHAR
Bonus_Pct	<input type="checkbox"/>		INTEGER
Effective_Start_Date	<input type="checkbox"/>		DATE
Effective_End_Date	<input type="checkbox"/>		DATE
SYSTEM_START_TIME	<input type="checkbox"/>		TIMESTAMP
SYSTEM_END_TIME	<input type="checkbox"/>		TIMESTAMP
TRANS_ID1	<input type="checkbox"/>		TIMESTAMP

Figure 5-50 Bitemporal HISTORY table for CUST_LOYALTY

If we run a few simple INSERT and UPDATE statements on our bitemporal CUST_LOYALTY table as shown in Example 5-31, we can view the impact on the base and HISTORY tables. We see that we INSERT four new customer records, then UPDATE one of the records (Linda MM) to give her the promotional bonus rate for 30 days.

Example 5-31 SQL to populate the bitemporal CUST_LOYALTY table

```
insert into cstinsight.cust_loyalty ("Customer", "Loyalty_Scheme", "Bonus_Pct",
    "Effective_Start_Date", "Effective_End_Date")
    values('Albert Johnson', 'Customer Loyalty Club', 100, '2011-09-15',
'9999-12-30');

insert into cstinsight.cust_loyalty ("Customer", "Loyalty_Scheme", "Bonus_Pct",
    "Effective_Start_Date", "Effective_End_Date")
    values('Joe Smith', 'No Loyalty Scheme', 0, '2012-06-22', '9999-12-30');

insert into cstinsight.cust_loyalty ("Customer", "Loyalty_Scheme", "Bonus_Pct",
    "Effective_Start_Date", "Effective_End_Date")
    values('Linda MM', 'Customer Gold Club', 150, '2012-03-12', '9999-12-30');

insert into cstinsight.cust_loyalty ("Customer", "Loyalty_Scheme", "Bonus_Pct",
    "Effective_Start_Date", "Effective_End_Date")
    values('Melissa Smith', 'Customer Loyalty Club', 100, '2011-11-01',
'9999-12-30');

UPDATE "CSTINSIGHT"."CUST_LOYALTY"
    FOR PORTION OF BUSINESS_TIME FROM
        '2012-08-01' TO DATE('2012-08-01') + 30 DAYS
    SET "Bonus_Pct" = 200
```

WHERE "Customer"='Linda MM';

After these statements, our CUST_LOYALTY and CUST_LOYALTY_HIST tables appear as shown in Figure 5-51 and Figure 5-52, respectively.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date	SYSTEM_START_TIME	SYSTEM_END_TIME
Albert Johnson	Customer Loyalty Club	100	2011-09-15	9999-12-30	2012-07-19 16:54:50.952	9999-12-30 00:00:00.000
Joe Smith	No Loyalty Scheme	0	2012-06-22	9999-12-30	2012-07-19 16:55:22.250	9999-12-30 00:00:00.000
Linda MM	Customer Gold Club	150	2012-03-12	2012-08-01	2012-07-19 16:55:42.564	9999-12-30 00:00:00.000
Linda MM	Customer Gold Club	200	2012-08-01	2012-08-31	2012-07-19 16:55:42.564	9999-12-30 00:00:00.000
Linda MM	Customer Gold Club	150	2012-08-31	9999-12-30	2012-07-19 16:55:42.564	9999-12-30 00:00:00.000
Melissa Smith	Customer Loyalty Club	100	2011-11-01	9999-12-30	2012-07-19 16:55:22.302	9999-12-30 00:00:00.000

Figure 5-51 CUST_LOYALTY bitemporal table after the INSERT and UPDATE SQL

Figure 5-52 displays the bitemporal table after INSERT/UPDATE.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date	SYSTEM_START_TIME	SYSTEM_END_TIME
Linda MM	Customer Gold Club	150	2012-03-12	9999-12-30	2012-07-19 16:55:22.26634	2012-07-19 16:55:42.564

Figure 5-52 CUST_LOYALTY_HIST bitemporal history table after INSERT/UPDATE

Notice that the history table has only one record so far, from the UPDATE statement.

A single transaction: The EFFECTIVE_START_DATE for the three “Linda MM” records in Figure 5-51 are identical TIMESTAMP values. This indicates that all three records were changed as part of a single transaction, which was the UPDATE operation in Example 5-31 on page 223.

Having the customer loyalty information stored in a bitemporal table allows different kinds of questions and queries to be satisfied by the same table.

For example, questions of this form can be asked: “What is the bonus point policy that applies to customer Linda MM during the first week of the month of August, 2012 (during the end of summer sale week)?”

That query is expressed as shown in Example 5-32.

Example 5-32 Querying the bitemporal table using business-time

```
SELECT "Customer", "Loyalty_Scheme", "Bonus_Pct", "Effective_Start_Date",
"Effective_End_Date"
  FROM cstinsight.cust_loyalty
  FOR BUSINESS_TIME FROM '2012-08-01' TO '2012-08-08'
```

WHERE "Customer" = 'Linda MM'

The results are shown in Figure 5-53. We can see that Linda MM is under the terms of the 200% bonus point promotion during the first week of August, 2012.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date
Linda Moore	Customer Gold Club	200	2012-08-01	2012-08-31

Figure 5-53 Results of the business-time query against the bitemporal table

A second kind of question involves the transaction time of activity in the database relating to the customer loyalty program. For example, a customer service representative might use the operational data warehouse to answer the question: "How many changes have been made to the customer loyalty program for customer Linda MM in the past year?"

That query is expressed as shown in Example 5-33.

Example 5-33 Querying the bitemporal table using system-time

```
SELECT * FROM cstinsight.cust_loyalty
  FOR SYSTEM_TIME FROM CURRENT TIMESTAMP - 1 YEAR TO CURRENT TIMESTAMP
 WHERE "Customer" = 'Linda MM'
 ORDER BY SYSTEM_START_TIME;
```

Figure 5-54 shows the results. We can see that Linda MM's record was updated four separate times, including the transaction that created it, over the past year.

Customer	Loyalty_Scheme	Bonus_Pct	Effective_Start_Date	Effective_End_Date	SYSTEM_START_TIME	SYSTEM_END_TIME
Linda Moore	Customer Gold Club	150	2012-03-12	9999-12-30	2012-07-19 16:55:22.266363	2012-07-19 16:55:42.564
Linda Moore	Customer Gold Club	150	2012-03-12	2012-08-01	2012-07-19 16:55:42.564833	9999-12-30 00:00:00.0
Linda Moore	Customer Gold Club	200	2012-08-01	2012-08-31	2012-07-19 16:55:42.564833	9999-12-30 00:00:00.0
Linda Moore	Customer Gold Club	150	2012-08-31	9999-12-30	2012-07-19 16:55:42.564833	9999-12-30 00:00:00.0

Figure 5-54 Results of the system-time query against the bitemporal table

There are many such examples of how bitemporal tables can be utilized. The power of the bitemporal table is the ability to ask both system-oriented and business-oriented questions of the same table in a data warehouse solution using InfoSphere Warehouse 10.1.

5.2.6 Views and temporal tables

InfoSphere Warehouse 10.1 and DB2 10.1 allow views to be created using both system-period and application-period temporal tables. Generally, they can be

used like any other views in DB2, but they have two different usage scenarios with distinct restrictions on how each can be used.

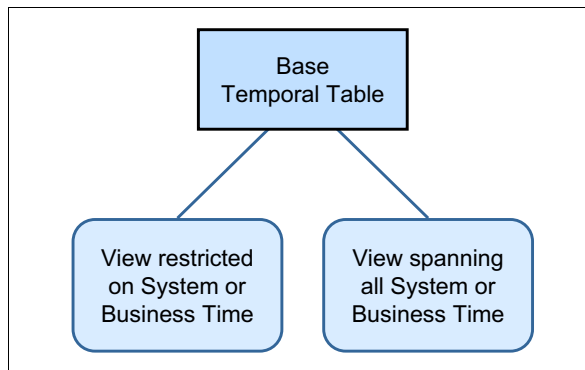


Figure 5-55 Two types of view on temporal tables

Shown in Figure 5-55, the two types of view usages are characterized as described here; in both cases, the view can be defined on the base temporal table of either system-period or application-period types:

- Restrict on temporal period

Define the view on either a certain point of time or a time period using the `FOR SYSTEM_TIME` or `FOR BUSINESS_TIME` clauses to restrict on time. Any other `WHERE` clause or non-temporal column restriction can be included in the view definition.

Any query against the view of this style cannot include any additional `FOR BUSINESS_TIME` or `FOR SYSTEM_TIME` temporal restrictions. The reason for this restriction is that the time constraint on time in the view can conflict with the additional time constraint in the query, which can lead to ambiguity.

- No temporal restriction

Define the view in terms of the base temporal table with no time period restriction on `BUSINESS_TIME` or `SYSTEM_TIME`. Data for all time periods included in the base temporal table are now available to queries against the view.

Queries against the view can include `FOR SYSTEM_TIME` or `FOR BUSINESS_TIME` clauses to restrict on time, and those restrictions are applied to all temporal tables included in the definition of the view.

5.2.7 Register settings for temporal tables

In this final section describing how the new temporal table functionality in InfoSphere Warehouse 10.1 and DB2 10.1 works, we take a look at new register

settings that facilitate running existing applications and simplify new application programming.

Using these new register settings can allow existing queries to gain maximum benefit from temporal features without having to be modified with the appropriate FOR SYSTEM_TIME or FOR BUSINESS_TIME clauses. The “CURRENT” time can be set and applied to all queries against system-period or application-period tables that do not otherwise include a FOR SYSTEM_TIME or FOR BUSINESS_TIME clause.

The statement in Example 5-34 sets the SYSTEM_TIME register to one week prior to the current time stamp.

Example 5-34 Set the current system time to offset of current time

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT_TIMESTAMP - 1 WEEK;
```

After this statement, all queries against system-period temporal tables that do not have explicit FOR SYSTEM_TIME clauses have this clause applied implicitly:

```
FOR SYSTEM_TIME AS OF CURRENT_TIMESTAMP - 1 WEEK
```

The statement in Example 5-35 sets the BUSINESS_TIME register to the literal date value for March 1, 2012.

Example 5-35 Set the current business time to a date literal

```
SET CURRENT TEMPORAL BUSINESS_TIME = '2012-03-01';
```

After this statement, all queries against application-period temporal tables that do not have explicit FOR BUSINESS_TIME clauses have this clause applied implicitly:

```
FOR BUSINESS_TIME AS OF '2012-03-01'
```

Avoid having two time constraints applied simultaneously: Similar to the restriction on views, it is critical to avoid having two time constraints applied to the same query simultaneously.

You cannot:

- ▶ Set the special SYSTEM_TIME register and use the FOR SYSTEM_TIME AS OF clause in a SQL query.
- ▶ Set the special BUSINESS_TIME register and use the FOR BUSINESS_TIME AS OF clause in a SQL query.

You must use one approach or the other, but never both at the same time.

5.3 Implications of temporal tables on operational warehousing

The new temporal data management features in InfoSphere Warehouse 10.1 (and DB2 10.1) provide a new and enhanced capability to address many of the requirements and challenges of a modern operational data warehousing solution. In this section we examine some best practices for temporal data management and analysis that drive the greatest benefit from this new technology.⁵

Areas where temporal data management and analysis have implications on operational data warehousing include:

- ▶ Complex workload and concurrency: impacts of read/write environments with high levels of user concurrency on data integrity of temporal tables
- ▶ Performance and storage efficiency: storage implications of system-period temporal tables and corresponding history tables
- ▶ Warehouse archiving: implications and practices around data archiving and system-period temporal history tables.
- ▶ Backup and recovery strategies: implications for range partitioning a temporal table solution
- ▶ Additional recovery considerations: roll forward implications and practices for system-period temporal and related history tables

5.3.1 Temporal tables, complex workloads and concurrency

Operational data warehousing solutions are characterized by high levels of user concurrency and mixed read and write database workloads. The combination of read and write and user concurrency can sometimes lead to issues of data integrity in system-period temporal history tables. The DB2 engine in InfoSphere Warehouse 10.1 ensures that any data integrity conflicts are detected. The DBA has a choice on how to handle the resolution, either through system-time adjustments or rollback.

First we examine how the high concurrency environment can create implications for data integrity in history tables. Consider a simple example of a system-period temporal table and its corresponding history table⁶. Imagine two concurrent

⁵ Much of this section's content has been drawn from the IBM white paper by Matthias Nicola, "Best practices: Temporal data management with DB2", IBM Corp., April 2012, which is available at the following website: <http://www.ibm.com/developerworks/data/bestpractices/temporal/index.html>

transactions, A and B, that both insert and update the same temporal table; see Table 5-1.

Table 5-1 Concurrent write operations on system-period temporal table

Time	Transaction A	Transaction B
T1	INSERT INTO mytable(c1, c2) VALUES(1, 15);	
T2		INSERT INTO mytable(c1, c2) VALUES(2, 30);
T3		COMMIT;
T4	UPDATE mytable SET c2 = 33 WHERE c1 = 2;	
T5	COMMIT;	

For system-period temporal tables, the time stamp of the first write operation within any transaction determines the system time that is assigned to *all* rows modified in the same transaction. From Table 5-1, we can see that transaction A performs an INSERT and UPDATE on the same table, so all rows impacted by those statements in both the base and history tables will have the system time stamp T1.

When Transaction B commits the INSERT statement, that row has a time stamp of T2. Starting with an empty system-period temporal table MYTABLE at the start of this example, we see the contents of MYTABLE and MYTABLE_HISTORY at time T3, as shown in Figure 5-56.

MYTABLE					MYTABLE_HISTORY			
C1	C2	System_start	System_end	Committed?	C1	C2	System_start	System_end
1	15	T1	9999-12-30	No				
2	30	T2	9999-12-30	Yes				

Figure 5-56 Base and history table contents as of time stamp T3

As of time T3, notice the following:

- ▶ The history table is empty, because we processed two INSERT statements with new rows for the base system-period temporal table MYTABLE.
- ▶ There are two records INSERTED into the base table MYTABLE; one each for Transactions A and B, respectively.

⁶ Source: Matthias Nicola, “Best practices: Temporal data management with DB2”, IBM Corp., April 2012.

- ▶ The record inserted at time stamp T1 (Transaction A) is uncommitted because the COMMIT does not occur until T5.
- ▶ The record inserted at time stamp T2 (Transaction B) is committed as of time stamp T3.

At time T4, Transaction A performs an UPDATE on the row inserted by Transaction B, which generates a corresponding history row. Consider the content of the history table, which is shown in Figure 5-57 with the modified base table.

- ▶ The historical values of the data columns C1 an C2 are copied from the updated row in the base table.
- ▶ The SYSTEM_START value is the time stamp of the original row in the base table, which is T2.
- ▶ The SYSTEM_END value is the time stamp of the UPDATE transaction, which is T1 in our case.

MYTABLE					MYTABLE_HISTORY				
C1	C2	System_start	System_end	Committed?	C1	C2	System_start	System_end	Committed?
1	15	T1	9999-12-30	No	2	30	T2	T1	No
2	33	T1	9999-12-30	No					

Figure 5-57 Base and history table contents as of time stamp T4

As a result of the concurrent operations and COMMIT sequence, the SYSTEM_START time stamp (T2) in the history table is *after* the SYSTEM_END timestamp (T1). This violates the data integrity constraint on system-period temporal tables that system start time stamps must be less than system end time stamps in the same history row.

This violation can be resolved in one of two ways:

- ▶ Rollback (SQL20528N)

The default behavior is to automatically roll back Transaction A and return error code 20528.
- ▶ System time adjustment (SQL5191W)

The DBA can choose for DB2 to automatically increase the conflicting timestamp (T1) to T2 + *some delta* that reflects the next possible time stamp after T2. This automatically eliminates the conflict and allows the transaction to continue. To use this option, the database configuration parameter SYSTEM_PERIOD_ADJ must be set to YES. Whenever the conflict and corresponding adjustment occur, warning SQL5191W is issued.

5.3.2 System-period temporal tables and history table storage

Although the benefits of using system-period temporal tables in InfoSphere Warehouse 10.1 have been thoroughly discussed at this point in the chapter, you also must consider the trade-offs. The introduction and management of history tables for every system-period temporal tables carries with it the performance and storage overhead associated with managing that history table. In this section, we consider these trade-offs and discuss various best practices to use when selecting and designing for system-period temporal tables.

Selecting tables that require history

When selecting tables that require history, the first step is to consider the implications of introducing system-period temporal tables and history tables and make a careful selection of which tables truly require history. Avoid introducing the history logic if there is no legitimate business justification that outweighs the overhead.

Consider the following⁷ resource implications for history tables:

- ▶ History tables consume storage. Although mitigated or reduced by use of DB2 compression, history tables will increase storage consumption by retaining a complete copy of every changed row in the base table.
- ▶ INSERT operations into the history table are subject to logging and impact logging I/O bandwidth.
- ▶ INSERTs into the history table use buffer pool resources and increase the need for page cleaning
- ▶ History tables usually have at least one index that carries its own storage and maintenance overhead.
- ▶ History tables impact elapsed time and resource requirements for database backups.

The scale of impact of the resource utilization areas depends on several factors, most important among them:

- ▶ Frequency of UPDATE and DELETE operations on your base table

In traditional data warehouse environments, the primary table operation other than SELECT is INSERT; most rows are inserted and left alone until removal to archive or deletion. In these environments, the ratio of history records to base table records will be quite low.

However, with the increasing emergence of operational data warehousing for which InfoSphere Warehouse 10.1 is so well suited, we expect a much higher

⁷ Source: Matthias Nicola, “*Best practices: Temporal data management with DB2*”, IBM Corp., April 2012.

frequency of UPDATE and DELETE activities, and a high ratio of history to base table rows. Thus, in the more active operational warehouse environments, we expect to see history tables proportionally much larger than traditional data warehouse environments.

- ▶ Average number of rows affected by UPDATE and DELETE operations

This factor goes along with the previous point, but it also speaks more to data model and application complexity. In the operational warehouse environment, the more sophisticated business and client self-service applications and functions tend to impact more tables in the schema and thus can compound the impact on history table resource consumption for temporal tables.

- ▶ Number of indexes defined on the history table and number of columns in those indexes

This is again related to the previous point. The more activity on the history table and the more complex the operations, then the more dependency on indexes and the more resources they require.

Important: Business need should be the primary driver determining the need for history tracking.

Given these implications, business need and requirements determine where history tracking and temporal data management is applied. Although it can theoretically be applied everywhere, use it only where business necessity dictates. Having made that determination, you can use the expected number of history inserts (based on workload characteristics) to estimate the size of history tables. Additional workload and business application analysis can help estimate the number and complexity of indexes on the history tables required to support the business requirements.

Vertical table splitting to reduce history table resources⁸

Building on the this discussion, we understand that the use of history tables and system-period temporal features introduces resource overhead to manage the history tables. Because we consider that DB2 will write the *entire before-change* row contents from the base table into the history table, we can foresee that even though an update on the base table might only change a single column such as a 2-character field, the history table requires space to store the entire row. With large row sizes, which is common in operational data warehouse implementations, we can see that there are fixed overhead costs for each row of the history table even if only a small proportion of the row is actually changed.

⁸ Source: Matthias Nicola, “Best practices: Temporal data management on DB2”, IBM Corp., April 2012.

This situation is brought into stark relief when we consider rows that have the following items:

- ▶ Large objects (BLOBs, CLOBs) or large XML columns
Though rarely updated, they are copied to the history table for each UPDATE/DELETE on the base table row.
- ▶ A large number of columns (in the hundreds) with a significant majority that are rarely or never updated
Again, even though they never change, they are copied over and over to the history table for each row UPDATE/DELETE.

Important: Consider a vertical table split to move mostly-static columns that do not require history tracking into a separate, non-temporally-managed table.

You can split the base table into two tables:

- ▶ One table to manage the columns that do change and require history tracking
- ▶ A second table to manage columns that do not change or otherwise do not require history tracking

This method provides the benefit of reducing the resource requirements of the history table, but it has some new overhead trade-offs of its own that must be considered:

- ▶ Each INSERT into the base temporal table now requires two INSERTs (one for the base, one for the split table) with impacts on primary keys and so on. This can reduce insert performance and increase application logic complexity.
- ▶ Less history is tracked (by design), which reduces logging and correspondingly improves UPDATE and DELETE performance (trade-off with reduced INSERT performance).
- ▶ Queries will frequently have to join both the base and split-off tables to get the values they need, thereby increasing application complexity and decreasing performance for those queries.

You must evaluate these trade-offs against your business requirements when considering these trade-offs and determining if vertical table splitting is a net gain or not.

5.3.3 History table implications for the warehouse archive strategy

The data archival strategy and solution for a traditional data warehouse or operational data warehouse must take into account the impact of the new

temporal data management capabilities in InfoSphere Warehouse 10.1 and DB2 10.1. Namely, the history tables generated for system-period temporal tables have to be incorporated into the data archive solution.

As with any other table in the warehouse, history tables accumulate rows over time and consume more and more resources. Also like any other table in the warehouse, history tables have usage patterns and availability requirements that have to be planned for. As such, history tables in particular and temporal tables in general have to be considered and included in the overall data retention, archival, and removal strategies. And as with the rest of the data warehouse solution, consider using tooling such as the IBM Optim Data Growth software or something similar.

However, after the data in the history table is archived, the next step is to remove, or “prune” the records that have been archived to free up database system resources. There are several ways to implement this pruning that can meet the needs of particular solution requirements.⁹

- ▶ One simple approach is to delete history rows that are older than a certain age. Consider a history table CUST_HISTORY. You can delete all history records older than three years with the SQL in Example 5-36.

Example 5-36 Delete history rows older than three years

```
DELETE FROM CUST_HISTORY WHERE SYS_END < CURRENT_DATE - 3 YEARS;
```

- ▶ Another approach can be to remove history rows that correspond to rows that have been deleted from the base temporal table. This can be accomplished as shown in Example 5-37.

Example 5-37 Delete history rows that have been deleted in the base table

```
DELETE FROM CUST_HISTORY  
WHERE CUSTOMER_ID NOT IN (SELECT CUSTOMER_ID FROM CUSTOMER);
```

- ▶ It might be decided that the history table will keep at most N versions of a row, and versions older than the last N versions can be deleted from the history table.
- ▶ DB2 range partitioning can be employed in the design of the history table to facilitate rapid and high performance pruning.

A combination of these techniques can be employed, depending on the particular requirements and infrastructure policies in place for any given solution.

⁹ Source: Matthias Nicola, “Best practices: Temporal data management with DB2”, IBM, Corp., April, 2012.

Integrate pruning into procedures: Perform history table pruning as part of the overall warehouse archival and data removal procedures. It is important to avoid unintended or unauthorized history data removal. Only grant DELETE, UPDATE, and INSERT privileges on the history table for the user ID that is intended to perform the pruning processes.

5.3.4 Backup, recovery, and pruning history with range partitioning

Temporal table and history tables are tables like any other in DB2 for all intents and purposes, and as such they can be range partitioned. This means that temporal and history tables can use range partitioning to enhance the implementation of backup and recovery, and archive and pruning activities in the warehouse solution.

Partitioning a history table: The partitioning scheme of a history table can be different from that of its base temporal table. For example, the temporal base table can be partitioned by month and the history table by year. However, take care to ensure that any partitions defined for the history table can absorb rows that are moved from the base to the history table.



Managing complex query workloads in an operational warehouse environment

In this chapter we discuss the challenges faced by complex query workloads in an operational business intelligence (BI) environment, and explain how you can address these challenges using Optim Performance Manager in conjunction with DB2 workload manager (DB2 WLM).

An operational data warehouse that must facilitate fundamentally different use cases and maintain performance objectives for each can be described as having a complex query workload. Short tactical queries such as fraud assistance or client care assistance applications that target small subsets of rows and must complete in two or three seconds are processed alongside decision support and long-running analytics queries that must complete in minutes or hours. Each task is important to the business and each must finish within the objectives set.

Using DB2 WLM in conjunction with Optim Performance Manager can help you define structures that resolve situations where unrestricted access to resources by complex query workloads with mixed priorities can cause the system to become unresponsive.

The recommended best practices for implementing a comprehensive DB2 WLM configuration are discussed in the paper *DB2 best practices: Implementing DB2 workload management in a data warehouse*, which can be found on the IBM developerWorks® site:

<http://www.ibm.com/developerworks/data/bestpractices/workloadmanagement/>

DB2 WLM is implemented as part of the DB2 database software, and this default implementation is referred to as stage 0. The Workload Manager Configuration dashboard in Optim Performance Manager can be used to automatically apply a stage 1 implementation of DB2 WLM, which you can then build on through stages 2 and 3.

6.1 Get started with Optim Performance Manager in your environment

Optim Performance Manager is a highly configurable application for monitoring the performance and health of a database. Performance metrics are collected from the monitored database and stored in an Optim Performance Manager repository database. Using the Optim Performance Manager web interface you can switch views between real-time and historical analysis of collected data to determine current and historical performance for an individual query or for an entire query workload.

Using Optim Performance Manager in a warehouse environment helps you identify which queries are consuming resources on individual database partitions and across the entire database.

Individual query workloads can be identified through a common attribute such as the client application ID, and IP address, or user ID. From within Optim Performance Manager, you create and configure a DB2 WLM workload object to align with a specific query workload. This enables you to filter data on dashboards and reports by workload object so that you can monitor and compare the performance of individual query workloads.

Service level objectives for the number of concurrent user sessions, the number of concurrent queries, and for elapsed query time can be easily aligned with performance metrics by workload.

To begin using Optim Performance Manager, you perform the following tasks:

- ▶ Create a database connection
- ▶ Configure database monitoring settings

6.1.1 Creating a database connection

Use the **Databases** dashboard in Optim Performance Manager to **Add** a database connection. Figure 6-1 shows a standard database connection where the DB2 instance user ID and password are used to create the database connection.

Edit Database Connection

[Learn more about database connections.](#)

Database Connection

Database connection name: * 203CSTINSDB

Comment:

Data server type: * DB2 for Linux, UNIX, and Windows

Database name: * CSTINSDB

DB2 instance: db2inst1

DB2 CLP alias:

Host name: * 101.101.101.101

Port number: * 50001

JDBC security: * Clear text password

Kerberos server principal:

User ID: * db2inst1

Password: *

Additional JDBC properties: Example: traceLevel=32;progressiveStreaming=1

JDBC URL: jdbc:db2://101.101.101.101:50001/CSTINSDB:retrieveMessagesFromServerOnGetMessage=true;securityMechanism=3;

Test Connection OK Cancel

Figure 6-1 Optim Performance Manager Database Connection page

6.1.2 Configuring database monitoring settings

In the Databases dashboard, select the database connection created and click **Configure Monitoring** to start the Edit Configuration Monitoring wizard. This wizard will guide you through the configuration steps.

Metrics are collected for different monitoring profiles such as SQL Statements and Connections, and Locking. Predefined templates for different database environments, including Business Intelligence, are available and these represent a collection of appropriate monitoring profiles and settings.

For an operational data warehouse, it is best to use one of the following predefined system templates:

- ▶ BI production system with low overhead

This system template is sufficient for normal operating conditions because it minimizes the effect on the database being monitored.

- ▶ BI production system with all details

This system template collects all details recommended for a BI production system. It is suggested that you use this template when you are troubleshooting a specific issue because it collects a significant amount of data, which can have an effect on the monitored database and on the size of the Optim Performance Manager repository database.

Figure 6-2 on page 242 shows Step 1 of the wizard where the “BI production system with low overhead” system template has been chosen. The Activate

monitoring check box is automatically checked to enable Optim Performance Manager to collect and process monitoring data for the database.

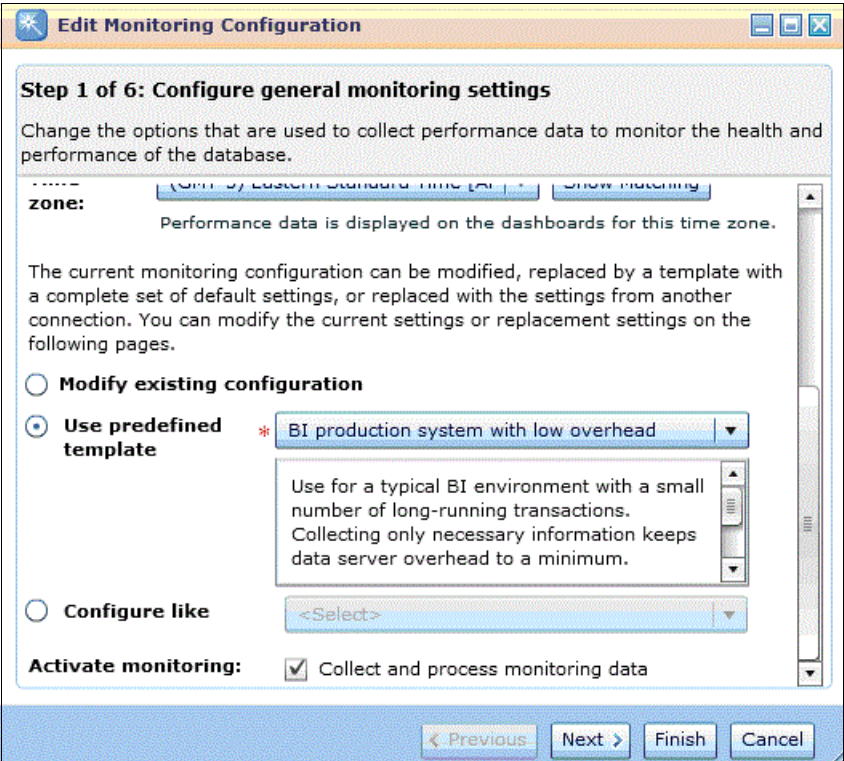


Figure 6-2 Using the predefined system templates to monitor a database

Step 2 of the Edit Monitoring Configuration wizard allows you to set the collection sampling interval and choose the individual monitoring profiles for which to collect data.

Important: It is useful to manually add the Locking and Workload Manager monitoring profiles during this step because they are not included in this template.

Figure 6-3 shows the default settings determined when you choose the “BI production system with low overhead” template.

Edit Monitoring Configuration

Step 2 of 6: Configure monitoring profiles

Define the type of monitoring data that is collected by enabling the corresponding monitoring profiles.

Selected configuration: System template BI production system with low overhead

Monitoring settings

Retention times and default sampling interval (--)

DB2 event monitor configuration

Data collection method

☐ Collect snapshots

☒ Collect in-memory metrics (DB2 9.7 or later)

Monitoring profiles

Inflight performance, reporting, or Workload Manager

These profiles collect performance statistics for the data server, which are shown in the inflight dashboards, in Workload Manager, or in the reports.

☒ Basic (1 min.)

☐ Locking (--)

☒ SQL statements and connections (1 min.)

☒ I/O and disk space (1 min.; 1 min.; --)

☐ Workload Manager (15 min.; 30 min.)

Extended Insight

< Previous Next > Finish Cancel

Figure 6-3 Monitoring profile settings for selected system template

Figure 6-4 on page 244 shows the Retention times and sampling intervals dialog box called by clicking **Edit** in Step 2 of the Edit Monitoring Configuration wizard. Use of the default settings shown in this figure enables data to be collected from

the monitored database every minute, and aggregated for up to two years before being automatically removed.

Retention times and sampling intervals

Specify how long to keep performance data and how often to collect performance data.

Inflight performance data collection settings

Inflight performance data includes information and statistics for the data server, which are shown in the inflight dashboards, in Workload Manager, or in the reports.

Sampling rate in minutes: 1 minutes

Inflight and Extended Insight performance data retention settings

Database information is stored in four aggregation levels. The aggregation levels control the amount of stored data. You can specify how long to retain the data for each aggregation level.

Aggregation level	Storage period
1: 1 minute	1 day
2: 15 minutes	1 month
3: 1 hour	3 months
4: 1 day	2 years


OK Cancel

Figure 6-4 Retention times and sampling intervals for performance metrics

The “Sampling rate in minutes” configuration parameter determines how often data is collected through DB2 in-memory metrics and snapshot table functions.

The “Aggregation level” determines when data is written to the Optim Performance Manager repository database to support the historical analysis of data. When data ages, it is aggregated into a table that reflects the time period of the aggregation. The value for aggregation level one always matches the value of the sampling rate.

The “Storage period” determines the retention period for aggregated data. When data ages past aggregation level four, it is removed.

The sampling rate for Workload Manager and Extended Insight is managed separately by clicking the Edit icon  beside each monitoring profile on Step 2. The sampling rate for Health monitoring is managed separately through the Health monitor dashboard.

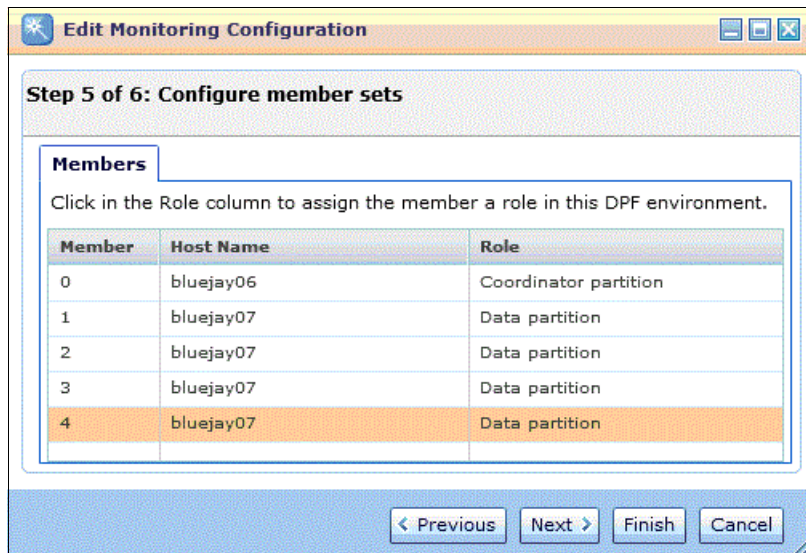
6.2 Optim Performance Manager in a partitioned database environment

Managing query workloads in a partitioned data warehouse database means that you have to interpret multiple sets of metrics across database partitions and often across multiple data nodes.

In a partitioned database environment, by default, data in Optim Performance Manager is displayed as an average across all database partitions for which data is being collected. When this data includes data for the coordinator partition, the metrics might appear skewed.

Step 5 of the Edit Monitoring Configuration wizard allows you to assign a role to each database partition in the partitioned database. These roles can be used in the Optim Performance Manager dashboards to filter data by a specific set of database partitions, also referred to as members. This eliminates the coordinator partition from the metrics you are viewing.

Figure 6-5 shows the role assigned to each database partition in a test environment that contained a single coordinator partition and four database partitions. You can also include ETL partitions, catalog partitions, and other roles.



Edit Monitoring Configuration

Step 5 of 6: Configure member sets

Members

Click in the Role column to assign the member a role in this DPF environment.

Member	Host Name	Role
0	bluejay06	Coordinator partition
1	bluejay07	Data partition
2	bluejay07	Data partition
3	bluejay07	Data partition
4	bluejay07	Data partition

< Previous Next > Finish Cancel

Figure 6-5 Assigning a role to each database partition in your database

Within Optim Performance Manager, you can look at metrics for a set of database partitions or for an individual database partition. This feature is made available on dashboards through the role drop-down box as shown in Figure 6-6.

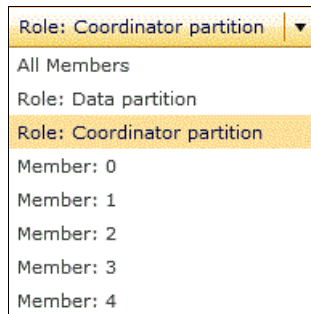


Figure 6-6 Filtering dashboard metrics by role

6.2.1 Collecting large volumes of performance metrics

If you modify the monitoring configuration to collect additional performance metrics, the default table space can become full before the data gets written back to the Optim Performance Manager repository database.

To avoid the default table space becoming full and data collection being disabled, create a separate 32 K table space for event monitors, as follows:

1. Create a table space across each data module by using the following sample statement, where OPMPG is the database partition group:

```
CREATE DATABASE PARTITION GROUP OPMPG ON ALL DBPARTITIONNUMS
CREATE BUFFERPOOL OPM32KBP DATABASE PARTITION GROUP OPMPG PAGESIZE 32K
NUMBLOCKPAGES 100000 BLOCKSIZE 16;
CREATE TABLESPACE OPMTS IN OPMPG PAGESIZE 32K BUFFERPOOL OPM32KBP OVERHEAD
4.0 TRANSFERRATE 0.04
```

Choose the media attribute values for OVERHEAD and TRANSFERRATE to reflect the storage media in your environment. The values used in this example are for illustration purposes.

2. Configure Optim Performance Manager to use the new table space:

On the Database dashboard, click **Configure Monitoring** to access the Edit Monitoring Configuration wizard for the monitored database. Change the DB2 event monitor configuration by using Step 2 of the wizard. To change the DB2 event monitor configuration, click the edit icon.

Click **Use custom table space** and from the list, select the table space that you created in step 1.

6.3 Using Optim Performance Manager to implement a Stage1 configuration of DB2 WLM

A stage 1 implementation of DB2 WLM, as described in the DB2 WLM best practices paper, can be configured through Optim Performance Manager by following these steps:

1. In the Workload Manager Configuration dashboard, click **OK** on the bottom right-side corner.
2. Click **OK** on the dialog box to use the default Service Superclass name, “Main.”
3. To apply the DB2 WLM configuration, click **Preview and Run SQL** to preview the proposed changes and then click **Run SQL** to apply the configuration changes to the database.

Note: This option is not presented to you if Optim Performance Manager detects an existing DB2 WLM configuration.

Performing this task creates and enables the Service Superclass “Main” together with a set of service subclasses and suggested limits and thresholds. These are created but not enabled.

Figure 6-7 on page 248 shows the Service Subclasses web page for the “Main” service superclass. Concurrency limits can be enforced at Service

Subclass level and thresholds for Time, Row, Concurrency and other limits can also be configured and enforced.

The screenshot displays the 'Service Subclasses' configuration window in the Optim Performance Manager. The left pane, titled 'Limits for the Service Subclasses of a Service Superclass', shows a table of service subclasses under the 'MAIN' superclass. The right pane, titled 'Thresholds', shows configuration options for 'Time Limits' and 'CPU time'.

Limits for the Service Subclasses of a Service Superclass

Select the service superclass: **MAIN**

To apply concurrency limits, enable the enforcement of concurrency limits for each service subclass that you want to limit.

Service Sub	Minimum Cost	Maximum Cost	Concurrency	Enforce	Activity types
TRIVIAL_DM	0	5000	2	<input type="checkbox"/>	DML
MINOR_DML	5000	30000	80	<input type="checkbox"/>	DML
SIMPLE_DML	30000	300000	32	<input type="checkbox"/>	DML
MEDIUM_DM	300000	5000000	16	<input type="checkbox"/>	DML
COMPLEX_D	5000000	Unbounded	8	<input type="checkbox"/>	DML
ETL	Not applicable	Not applicable	8	<input type="checkbox"/>	LOAD
SYSDEFAULT	Not applicable	Not applicable	2	<input type="checkbox"/>	

Thresholds

Time Limits

Threshold type: **Activity total time**

Detects and controls rogue activities that might run for too long.

Time limit (seconds): 14400

Monitor the activities that exceed the limit: ☒

Details collected: WITH DETAILS

Database partition: COORDINATOR

Stop the activities that exceed the limit: ☐

Enable threshold: ☒

Threshold type: CPU time

Detects and controls the activities that use excessive processor resource

CPU time limit (seconds): 1

Check interval (seconds): 60

Figure 6-7 Using Optim Performance Manager to configure DB2 WLM

6.3.1 Sample scenario using DB2 WLM and Optim Performance Manager to manage performance

This scenario demonstrates how Optim Performance Manager and DB2 WLM can be used to monitor and manage performance in a database. In the scenario, the business had reported that performance for the operational reports application was slow and this was affecting normal business operations.

The two query workloads referenced are:

- **Reports**
The reports workload represents operational queries. An SLO determines that reports workload queries should complete within 30 seconds.
- **Sales**
The sales workload represents analytics queries, from an IBM Cognos application, that have SLOs for minutes rather than seconds.

To begin, we apply a filter to the SQL statements dashboard, as shown in Figure 6-8, so we can view queries issued for the REPORTS workload. Using the Historical Data view, we review data for the previous 20 minutes.

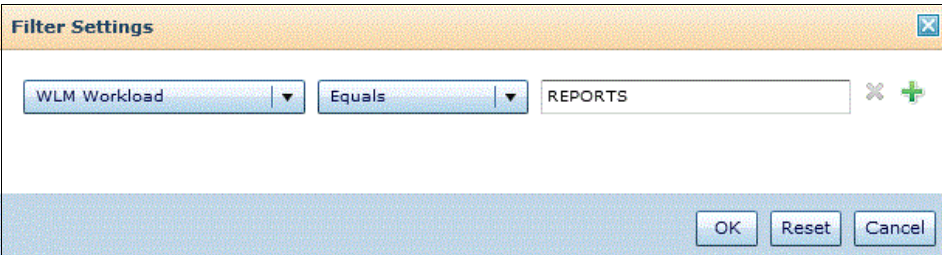


Figure 6-8 Create filters to focus your view of SQL statements

The SQL Statements dashboard, shown in Figure 6-9, reveals that performance of REPORTS queries has deteriorated to 57 seconds over the last 20 minutes, which breaches the SLO of 30 seconds.

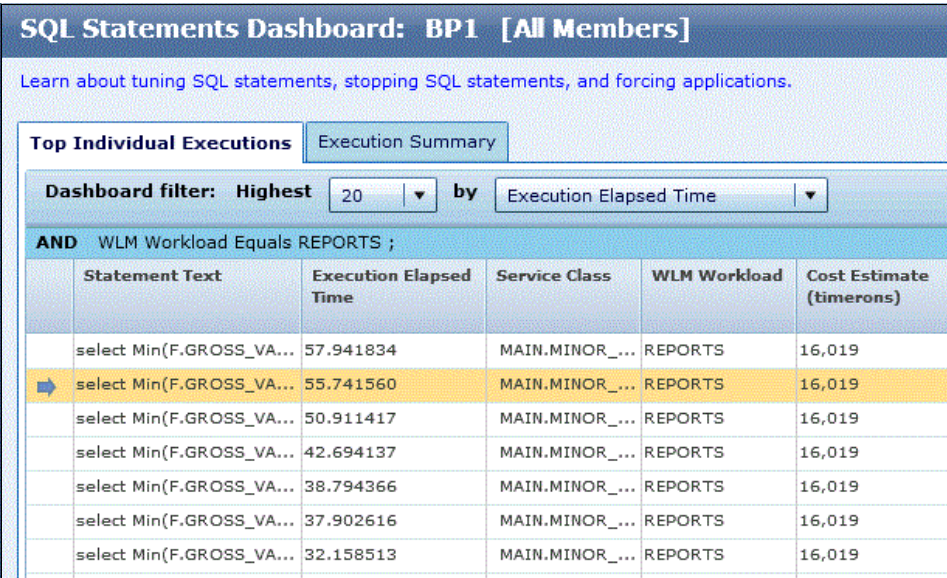


Figure 6-9 Elapsed time for operational queries deteriorates from 24 to 57 seconds

By adding a second filter to the SQL Statements dashboard for the SALES workload, queries for both workloads for a 20-minute period are displayed as shown in Figure 6-10 on page 250. When we order the SQL Statements by start

time, it is noted that multiple SALES workload queries are executing during the same period that REPORTS query performance deteriorates.

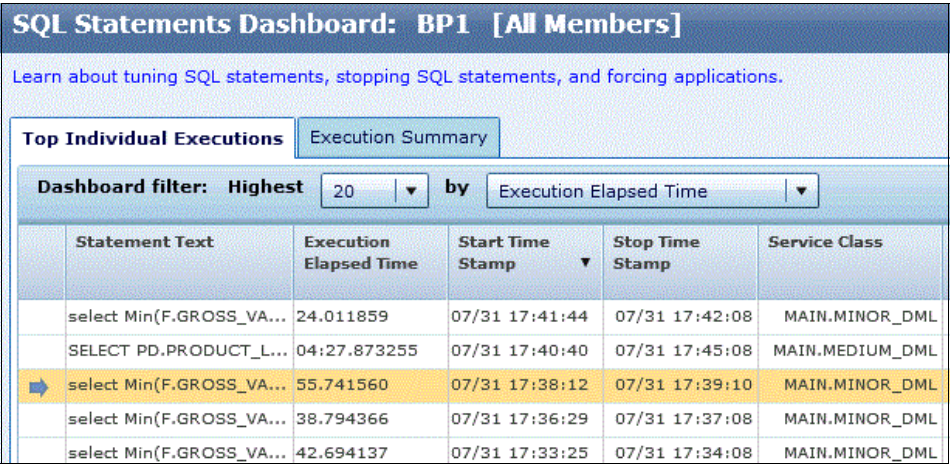


Figure 6-10 REPORTS workload queries are slower when multiple SALES queries are in operation

Using the SQL Statements dashboard, we can determine that the SLO of 30 seconds is being breached and the root cause is when analytics queries from the SALES workload are issued concurrently.

To monitor this situation, the Activity Total Time threshold type is enabled at the service subclass level to monitor and collect data where a REPORTS query exceeds 30 seconds. Figure 6-11 on page 251 shows how the Time limit of 30

seconds was enabled. In the example shown, we elect to collect data when the threshold is violated rather than stop the activity (query) from completing.

The screenshot shows a configuration window with three main tabs: 'Properties', 'Thresholds', and 'Workloads'. The 'Thresholds' tab is selected. Inside this tab, there are three sub-tabs: 'Time Limits', 'Row Limits', and 'Concurrency Limits'. The 'Time Limits' sub-tab is active. It displays the following settings:

- Threshold type:** Activity total time
- Description: Detects and controls rogue activities that might run for too long.
- Time limit (seconds):** 30 (with up/down arrows)
- Monitor the activities that exceed the limit:** ☒
- Details collected:** WITH DETAILS (dropdown menu)
- Database partition:** COORDINATOR (dropdown menu)
- Stop the activities that exceed the limit:** ☐
- Enable threshold:** ☒

Figure 6-11 Enabling monitoring of 30 seconds threshold for REPORTS workload

When we return to the SQL statements dashboard, we revert the view to **Real-time data** and add the **Threshold violations** column to the dashboard. We can now see each SQL statement for the REPORTS workload that has exceeded the 30-second SLO set and what other queries were executing when the 30-second time limit threshold was violated.

To ensure that the SLO for REPORTS workload was safeguarded, we enabled a **Concurrency Limit** such that only one SALES workload query could be executing at any one time. Further SALES queries submitted will be placed in a wait queue. Figure 6-12 on page 252 highlights the key configuration elements:

- ▶ The SALES workload is selected.
- ▶ The Work Classes tab is selected.
- ▶ The Concurrency Limits tab is selected.
- ▶ Maximum concurrent activities is set to 1.
- ▶ The unbounded queue length check box is set.

Overvi... Database **Workloads** Service Superclass... Service Subclasses Performance Objectiv...

Create workloads to identify, categorize, and manage different sources of database requests. For each workload, specify connection attributes that re
A best practice is to create one workload for each source of work that you want to manage.

Workload Evaluation [Hide details](#)

Requests are assigned to the first workload with connection attribute values that match the request.

[+ Add](#) [Delete](#) [Up](#) [Down](#)

Name	Connection Attributes
REPORTS	systemUser=REPORTS
SALES	systemUser=SALES
SYSDEFAULTUSE	
SYSDEFAULTADM	

Properties **Thresholds** [Monitoring Properties](#) [Workload Privileges](#) [Work Classes](#)

Work class name: SALES
Work class type: ALL

[Time Limits](#) [Row Limits](#) **Concurrency Limits** [Temp Space Limits](#) [Estimated Cost Limits](#)

Threshold type: Concurrent Database Coordinator Activities

Limits the number of concurrent coordinator activities.

Maximum concurrent activities:

Unbounded queue length: ☒

Queued activities:

Monitor the activities that exceed the limit: ☒

Details collected: [WITH DETAILS](#)

Database partition: [ALL](#)

Stop the activities that exceed the limit: ☐

Enable threshold: ☒

Figure 6-12 Enabling a Concurrency Limit threshold for a workload

Returning to the SQL Statements dashboard, we can now see where SALES queries are issued concurrently. Only one query is executed while the other query or queries wait. Figure 6-13 shows the Workload Manager section of the Application tab on the dashboard for a query.

Workload Manager	
Queue wait time:	01:18.745000
Queue waits:	1
Latest WLM workload:	SALES

Figure 6-13 Sales workload queries are queued where multiple are submitted

The Workload Manager Service Subclass report can also be used to review details by Service superclass and Service subclass. Figure 6-14 shows the report parameter window.

Report type:	* Workload Manager Service Subclass report
Description:	The Workload Manager Service Subclass report provides the definition of a particular service subclass and the definitions of associated workloads and thresholds. It also provides detailed information about the service subclass, which include histograms for queue time, execution time, and lifetime.
Report duration: (GMT-04:00):	<input checked="" type="radio"/> Show the most recent activity of the last 1 hour <input type="radio"/> Show period of 1 hour With start date of 07/23/2012 and time 09:56 <input type="radio"/> Start date 07/23/2012 and time 09:56 End date 07/23/2012 and time 09:56
Service superclass:	* MAIN
Service subclass:	* MEDIUM_DML

Figure 6-14 Workload Manager Service Subclass report

6.4 Understanding query workloads in your environment

A system with a complex query workload can become unresponsive if access to resources is not managed in a prudent way. You can avoid such unresponsive situations by using careful planning and workload management techniques.

A *query workload* can be defined as a set of queries that access the same set of tables and retrieve similar result sets. A query workload can be identified through shared attributes such as being through the same client application interface or through shared connection data which might include application ID, user ID, source IP address, client ID, or client version.

Use these simple guidelines to begin the process of being able to manage the query workloads in your environment:

- ▶ Define service level objectives that align with each query workload. (example: elapsed time with workload filter)
- ▶ Allow queries to begin processing when resources are available for the query to complete.

- ▶ All queries with a priority must be given resources to complete within service level objectives (SLO).
- ▶ Educate the business as to how queries are identified, grouped, and assigned resources.

6.4.1 Identify/define query workloads with Optim Performance Manager

Start by identifying each application that is to access the database and understand the nature of that application, how queries are generated, and how many concurrent queries and users you expect.

Queries in a data warehouse environment are less predictable than in traditional OLTP environments. For example, an interactive business intelligence (BI) application might generate many unique queries that might be transformed from multidimensional expression (MDX) to SQL and represent a challenge for both the DB2 optimizer and for the optimization of your cache hit ratio.

Each query workload is typically aligned to a group or department of users, a BI application, or a set of reports.

Perform these steps to identify and define your workload:

1. Use the Connections dashboard to review all database connections.

Identify and reconcile individual database connections with expected workloads.

Optim Performance Manager can capture many connection attributes and not all are shown by default. Click **Choose columns** to add additional columns to the Connection dashboard to further help you identify each connection.

Figure 6-15 shows some of the additional connection related data that you can add to the data grid on the Connections dashboard.

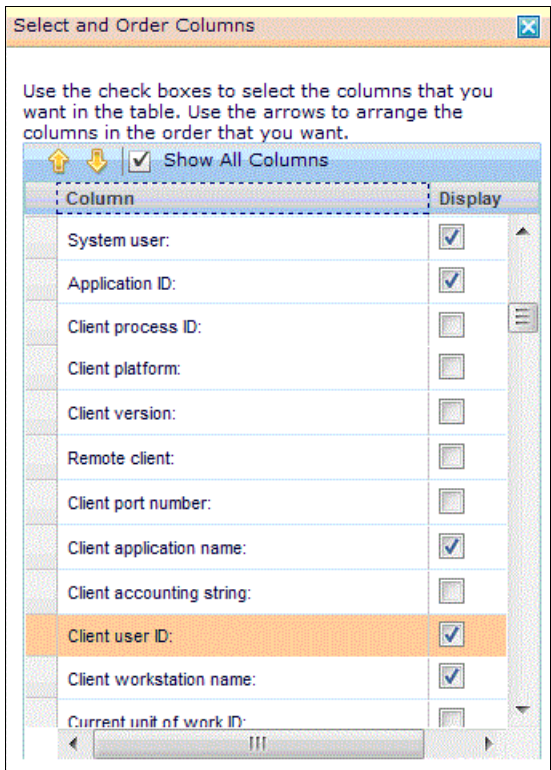


Figure 6-15 Column list of items that appear on the Connections dashboard

2. Use the SQL Statements dashboard to identify queries by connection.

When you identify an individual connection, use the filters in both the Connections dashboard and the SQL statements dashboard to get a more complete picture of usage patterns and queries executed.

In the SQL statements dashboard, identify the top five SQL statements issued by a single connection over a period of time; use the time slider in conjunction with the Historical analysis mode to achieve this.

Replace the connection filter with the most common SQL statement text used and establish which other connections also issue the same SQL statement. Using this approach allows you to establish patterns of use and shared workloads across different connection types.

3. Confirm your query workload definitions.

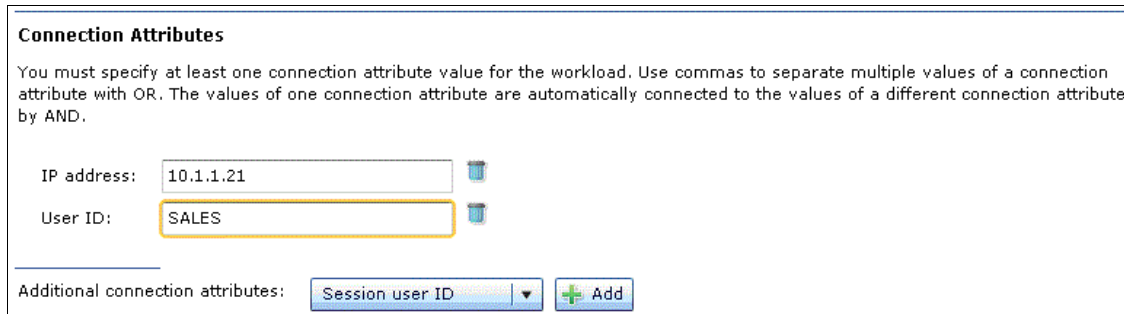
Complete a definition of each workload by confirming connections attributes used, query types executed, and tables accessed. Reconcile and align these findings with the service level objectives set by the business.

After you have identified and reconciled each connection with each workload, you can begin to create DB2 WLM objects through which these individual workloads can be monitored and managed.

6.4.2 Create/monitor new workloads with Optim Performance Manager

You can build on the stage 1 implementation of WLM by further configuration of WLM through Optim Performance Manager. The following steps show how to add a sample query workload called Sales:

1. On the Workload Manager Configuration dashboard, click the **Workload** tab, then click **Add** and enter the name of the workload that you want to add. In this example, the workload name is SALES_WL. Click **OK**.
2. In the Connection Attributes section of the Properties pane, define a connection attribute by clicking **Add**. Figure 6-16 shows the user ID and IP address that are associated with the application workload used. You might choose other identifiers in your environment.



Connection Attributes

You must specify at least one connection attribute value for the workload. Use commas to separate multiple values of a connection attribute with OR. The values of one connection attribute are automatically connected to the values of a different connection attribute by AND.

IP address:

User ID:

Additional connection attributes: Session user ID + Add

Figure 6-16 Connection attributes used to identify a workload

3. On the Service Superclasses tab, add a SALES_SC service superclass object by clicking **Add** and then clicking **Create a service superclass** from a

template. This option creates the service superclass based on best practices and also implicitly creates the underlying service subclasses.

Figure 6-17 shows the dialog box where you enter the Service Superclass name and select the radio button to create the service superclass from a template.

Overview Database Workloads Service Superclasses

Create service superclasses for the activities that are routed f

Service Superclasses

+ Add Delete

Add a Service Superclass

Specify the name of the service superclass.

Name: * SALES_SC

☒ Create a service superclass from a template

☐ Create an empty service superclass

OK Cancel

Figure 6-17 Creating a new Service Superclass

4. Map the SALES_SC service superclass to the SALES_WL workload object that you created in Step 1. To map the superclass to the workload object, use the Workload Definition section, shown in Figure 6-18, on the Properties tab of the Workloads tab.

Workload Definition

Specifies the service classes, evaluation position, connection attributes, and concurrency basis.

Name: SALES_WL

Comments:

Related service superclass: * SALES_SC ▼

Figure 6-18 Mapping the new service superclass to the new workload

5. To view and issue the DDL statements that apply the SALES workload and service superclass to the WLM configuration on the monitored database, click **Preview and Run SQL** on the top right of the web page.

You have now created a workload through which you can identify all activities for connections that use this IP address or user ID to connect to the monitored database. This enables easier reporting, ordering, and filtering of dashboard data grids. In addition, you can further configure DB2 WLM to establish controls against individual aspects of a workload.

Figure 6-19 shows the workload-related columns that can be added to the Connections dashboard so that connections can be ordered and filtered by workload.

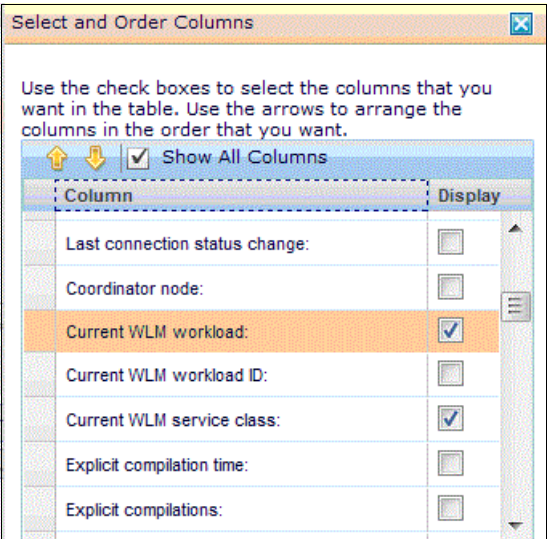


Figure 6-19 Adding columns to the connections dashboard

6.4.3 Define service level objectives for all query workloads

In defining service level objectives for each type of query, you can set expectations for the business and also determine thresholds within your workload management configuration that are aligned with the service level objectives set by the business. You can use Optim Performance Manager to observe and report on workload activity.

Although the query workload represents the capacity of the business to gain insight from the data warehouse, resources must also be guaranteed to meet the objectives for data ingest and data maintenance operations. Chapter 7, “Understand and address data latency requirements” on page 267 looks at the

data ingest workload and Chapter 8, “Building a corporate backup and recovery strategy” on page 289 looks at the backup strategy workload in more detail. It is advisable that you create a separate user ID for each maintenance workload so that you can more easily monitor, secure, and audit each maintenance workload separately.

6.5 Improving the overall performance of an activity

A large proportion of database administrator time is spent reacting to query and workload performance issues. There is a point of diminishing returns where the capacity of the system and the combined unmanaged workloads do not reconcile. A production system is not the correct place to resolve under-performing database design structures or inappropriate concurrent workload numbers.

To improve the overall performance of an activity, you must understand what other processes are running in the same time frame and what options are available in terms of introducing change. Adding an index, MQT, or changing the structure of a table might improve the performance of a query; however, it might also reduce the performance of an ETL or backup operation. Context is key. The performance of an activity must be judged primarily against the service level objective for the activity and against best practice principles for database and query design; is the database design or query inefficient?

Consider the following when trying to improve the performance of an activity:

- ▶ Can the activity or other activities that run concurrently be moved to another schedule or priority?
- ▶ Is the query inefficient?
- ▶ Is the underlying database design structure inefficient?
- ▶ Can more resources be assigned to the query?
- ▶ Is the query using hot, warm, or cold data?

Optim Performance Manager integration with Optim Data Studio can be used to record and tune an individual query or group of queries.

6.6 Approaches to workload tuning

When tuning a query workload, you must understand the conditions under which the workload must run and the expectations for performance. Consider the following questions:

1. What is the elapsed execution time for each query in the workload, and for the entire workload?
2. What are the elapsed execution times under normal conditions and under peak usage conditions?
3. What is the service level objective for the workload?
4. What other workloads are to run concurrently with this workload, and what are the service level objectives for those workloads?

When you have the answers to these questions, you know what is expected and you also know how to validate any improvements you make. By using Optim Performance Manager to configure each separate query workload, you can easily derive the answers for questions 1, 2, and 4. In addition, you can create a performance baseline for the system and database key indicators to evaluate any effect that might occur if you make changes.

6.6.1 Tuning a workload for optimal performance

Use Optim Performance Manager with IBM Data Studio Version 3.1.1. Configure Optim Performance Manager so that SQL statement text can be transferred from the SQL Statements and Extended Insight dashboards directly into IBM Data Studio. You can then analyze and tune the SQL statement or query workload by using the features in IBM Data Studio.

Tuning SQL statements

An Actions drop-down box on the SQL Statements dashboard, shown in Figure 6-20 on page 261, calls and passes the individual statement text or entire

workload to IBM Data Studio. IBM Data Studio must be open on your desktop and connected to the database.

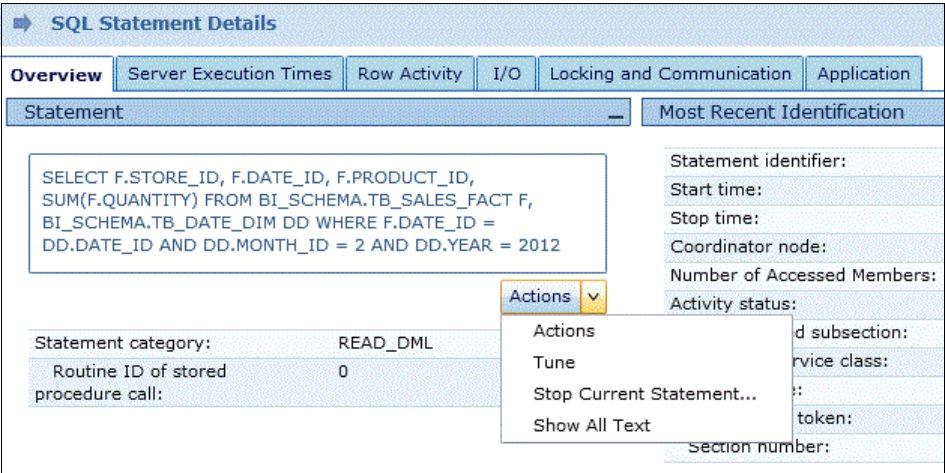


Figure 6-20 Using the Tune action sends the query text to IBM Data Studio for analysis

To run the statement through the DB2 optimizer and generate an optimizer plan, cost, and recommendation report, in IBM Data Studio, click **Select What To Run**.

Based on the result and recommendations for the query, you can tune and issue the query multiple times and measure any cost improvements. You can then use the SQL Statements dashboard to view a history of SQL statement executions over a period of time to determine improvements.

When you have completed all changes, evaluate the performance baseline and elapsed execution times for tuned queries and for the entire workload for those workloads affected.

6.7 Monitoring query workloads: Real-time operational to show the state and health of the system

System health, database health, query workload health, and the health of an individual query are all important but subjective judgments. Reacting to events can help minimize the effect any change in state or degradation in performance might have.

Good housekeeping and attention to regular administration tasks can help maintain a healthy system, as listed here:

- ▶ Performing regular backups.
- ▶ Pruning the recovery history file.
- ▶ Managing the data lifecycle to maintain a high ratio of active data in your database.
- ▶ Using MDC tables where appropriate to avoid costly reorganization tasks.
- ▶ Referencing system logs on a regular basis.
- ▶ Adhering to best practice recommendations for database design.
- ▶ Regularly evaluating distribution keys to avoid data skew.

Optim Performance Manager provides two key features that can help you detect potential problems in advance:

- ▶ Health monitoring
- ▶ Performance baselines

6.7.1 Health monitoring

The health monitoring feature allows you to quickly view a list of utilities in operation, connected applications, and table space details. This can help you quickly identify if there are anomalies in your environment. You can also set Alert thresholds to allow you to let Optim Performance Manager monitor your database and notify you when a threshold has been breached.

Alerts are triggered when a predefined threshold or condition exists on the configured database or within the Optim Performance Manager application. Alerts are generated against Performance, Health, and Operational thresholds

and events. You configure alerts through the Alert dashboard, as illustrated in Figure 6-21.

Edit Alert Notification

Alert type: Database Partition Availability

Alert description: The availability of the database partitions.

Severity: Critical

☒ Enabled

Email addresses: Add Edit Delete

joe@ie.ibm.com

☒ Send SNMP notifications

Reminder interval: Repeat every 15 minutes

☒ Send a notification when an alert is closed

Blackout time: ☒

Start time: 01 : 00 AM

End time: 08 : 00 AM

Notes:

OK Cancel

Figure 6-21 An alert to notify of database partition failure

Alerts thresholds can be enabled for a wide and extensive predefined list of events; for example, the number of connections to the database, table space container utilization, log space used, application timeouts, or deadlocks. In addition, alert thresholds can be assigned to an individual database partition or role.

When enabling performance alerts, keep the following recommendations in mind:

- Limit the number of alert thresholds to those that have a demonstrable effect on the performance or health of the monitored database.

- Configure the “Performance alerts at a Partition role level” so that you receive alerts for thresholds that directly relate to specific database partitions. For example, the number of sort overflows on the administration node is often different from the number of sort overflows on a database partition. In this situation, set a separate threshold for the data partition role only.

6.7.2 Performance baselines

Optim Performance Manager provides baseline functionality that allows you to compare metrics for two historical time periods for an historical time period versus real-time data.

Figure 6-22 shows a performance baseline set for a specific two hour period of time. This time could represent normal operating conditions or peak operating conditions when you know the system was within service level objectives. The baseline is set by simply selecting the time period on the time slider, clicking the **base** drop-down box, and then clicking **Set**.

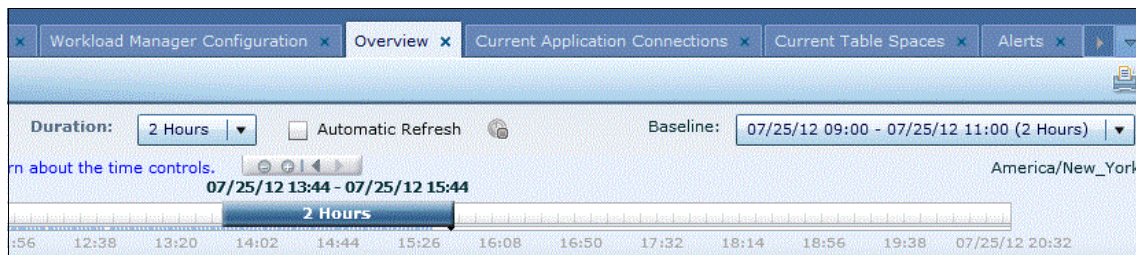


Figure 6-22 Performance baseline set for a specific two hour period

To do a baseline comparison, then switch Optim Performance Manager to Real-time Data monitoring. This is achieved by setting the View drop-down box to Real-time, as shown in Figure 6-23.

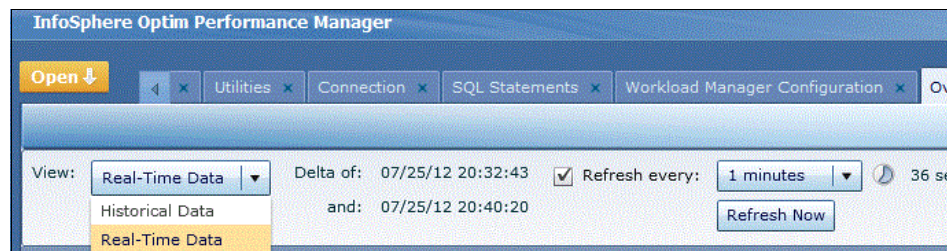


Figure 6-23 Switching view from historical data to real-time data

The performance baseline comparison functionality now becomes available. Optim Performance Manager marks any deviations by changing the metric bar

from blue to orange. By hovering the mouse over the metric, you can see the performance baseline and real-time metrics and standard deviation.

Figure 6-24 shows a baseline comparison where an increase occurred in the number of connections, average response time, and row throughput that indicates that the system is busier now than when the baseline was set. The metrics box shown relates to the Rows Selected indicator. The current value for Rows selected is 7,256. However, the baseline value for Rows selected was 130, revealing a marked difference in behavior.

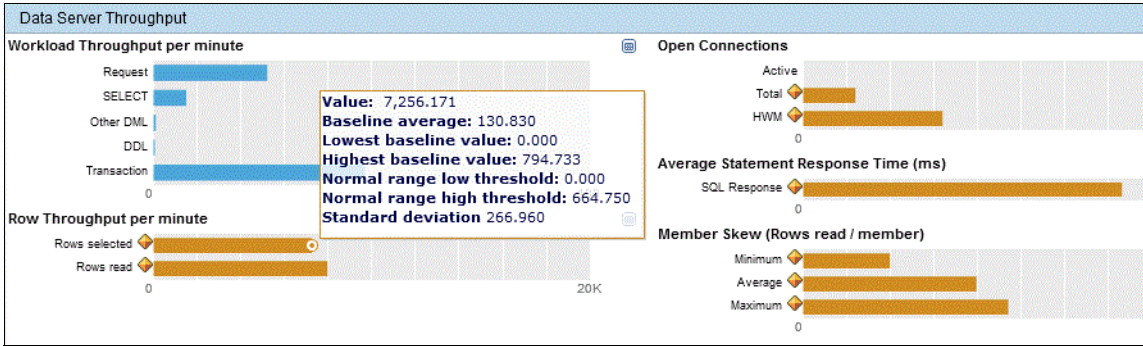


Figure 6-24 Baseline comparison showing marked changes in system usage



Understand and address data latency requirements

In this chapter we describe how to approach the challenge of identifying and meeting your data latency requirements in an operational business intelligence (BI) environment. We also take a look at how the SQL Warehousing (SQW) tool can be used to develop an application for ingesting data into the data warehouse.

Getting the right information to the right people at the right time and in the right format can be more complex in an operational BI environment than in a traditional data warehousing environment because the timelines for each workload are more aggressive.

Understanding the data latency requirements for each data source, which is the elapsed time required to collect and store data for presentation to the query workload, is key in determining the most appropriate tools and approach to use to ingest data in as efficient a manner as possible.

In an environment that needs to ingest data and facilitate online backup and maintenance operations while meeting service level objectives for data availability and query performance on a 24x7 basis, finding the right balance of resource allocation is a challenging task.

7.1 Understand your data latency requirements

The initial planning and architecting steps of a data ingest project must quickly identify the key characteristics in the ingest process for this project.

When that is done, you can then proceed to work on details (or seek guidance) applicable to your situation. This section covers:

- ▶ Quantifying your service level objectives: quantifying the key elements of the business requirements for the ingest process.
- ▶ Analyzing your ETL scenario: identifying what general parameters of the data ingest process you are dealing with.
- ▶ Calculating the ingest rate: help determine how challenging the performance requirements are.

Note: The term ETL is used to describe an application used to ingest data into a data warehouse. The SQL Warehousing (SQW) tools discussed in this chapter are more closely aligned with an ELT process where data is extracted from source, loaded into a staging area of the database, and transformed and moved to the data warehouse tables.

7.1.1 Quantify your service level objectives

This section describes how to quantify key aspects of the business requirements for your ETL application design. These requirements are expressed as service level objectives (SLOs).

To calculate SLOs, establish the ingest schedule and also the volume of data presented and the processing window available during each ingest schedule. Use these figures to calculate the data ingest rate that your ETL application must contain. Collectively, these figures comprise your service level objectives for the ETL application. All design and development decisions must be anchored around meeting these objectives in concert with all other workloads.

A data warehouse often has multiple SLOs for different target tables or data sources. Defining SLOs requires quantifying the following elements:

- ▶ Data latency

Data latency refers to the gap, in time, between when data enters source systems and when it must be available for query in the data warehouse. Typical values are: “start of next business day,” “30 minutes,” or “5 minutes”.

- Processing window

The ingest processing window is the period where the data is available for processing by the ingest process. For example, if data is presented to the data warehouse at 4 p.m. and the processing must be complete by 5 p.m., then the processing window is one hour.

- Data volume

Data volume refers to the quantity of data presented for ingest during each processing window. Consider both average and peak volumes and plan for peak volumes.

7.1.2 Calculate the data ingest rate

Use the values from the SLOs to calculate the rate of ingest required for each of the data sources. Express the target data ingest rate per data node as megabytes per second (MBps).

Calculate the ingest rate as follows:

- Estimate the volume of data in megabytes (MB) to be ingested per ingest cycle.
- State the time in seconds allowed for the ingest process to complete.
- Divide the volume by the time. In a partitioned database, then divide by the number data nodes receiving data.

The data volume and ingest rate each refer to the raw data to be ingested. Data transformations and the implementation of indexes, materialized query table (MQTs), and multi-dimensional clustering (MDC) tables can significantly increase the actual data and the number of transactions needed in the database. After they are identified, execution time for these additional transactions must be accommodated.

Through this process you will understand at what rate you need to be able to ingest data to meet your service level objectives. Your infrastructure must have the capacity to support the ingest rate and also support your service level objectives for the query and maintenance workloads. This must be the focus of your initial infrastructure and data throughput tests.

7.1.3 Analyze your ETL scenarios

This section provides a checklist to help you identify the key characteristics of your ETL application in a systematic fashion. The key distinctions and options are presented for each item.

Choose the one that best matches your situation. A checklist item can have multiple answers because there might be different answers for different data sources and tables in your project.

The checklist has the following sections:

- ▶ Determining your data ingest pattern
- ▶ Transformations involving the target database
- ▶ Data volume and latency
- ▶ Populating summary (or aggregate) tables

Determine your ETL pattern

In an operational data warehouse environment, the luxury of having an offline window at the end of each day to process data in large batches is not always available. It is expected that data is presented for processing at frequent intervals during the day and that data must be ingested online without affecting the availability of data to the business. The different patterns can be described as follows:

- ▶ **Continuous feed**
Data arrives continually in the form of individual records from a data source or data feed using messaging middleware (or by OS pipe, or through SQL operations). The ETL processes run continuously and ingest each insert and update as it arrives. Thus, new data is constantly becoming available to business users rather than at fixed intervals.
- ▶ **Concurrent batch (“Intra-day batch”)**
Several times a day, data is extracted from source system and prepared for ingesting into the target database. The ETL processes data in batches (files) as they arrive or on a schedule. The target table is updated at scheduled intervals, ranging from twice a day to every 15 minutes.
- ▶ **Dedicated batch window (“daily batch”)**
After the close of the business day (for example, 5 p.m.), data is extracted from a source system and prepared for ingesting into the target database. The ETL application populates the target project table during a dedicated, scheduled batch window (for example, 5 p.m. to midnight).

A given database, star-schema (or even a given dimension or fact table) might be populated using more than one pattern.

Although the pattern labels emphasize that each pattern differs in terms of latency, that is not the only or primary difference. Each pattern requires a somewhat different approach in articulating service level objectives and deciding which ingest methods might be suitable.

Transformations involving the target database

Data cleansing, surrogate key identification, or the application of business rules are some of the transformation tasks that might have to occur before data is ingested into the production database. Table 7-1 identifies the patterns that might exist in your environment.

Table 7-1 Data transformation patterns

Label	Description
None - Ingest/Insert ready	When the data is first presented to the warehouse, the records are ready to populate directly into the production table.
Store only	When the data is first presented to the warehouse, the records still need one or more transformation steps. The database is used to store the data during these steps; data is extracted out of a staging table in the database, transformed, then put back into the next staging table or into the production tables.
Process transformations	Same as the “Store only” option, but transformation logic is executed within the database, usually using stored procedures. (This is the so-called “ELT” or “ETLT” approach.)

The options presented in Table 7-1 represent valid ways to allocate data processing between the database server and an external server. Each approach has situations where it is most appropriate, and the following examples illustrate when each option is suitable:

- ▶ **Ingest ready**
When ETL processing (for example, DataStage) exists to prepare the data, and when there is a need to minimize the load on data server resources.
- ▶ **Store only**
Same as for “Ingest ready” but data must be queried by users during intermediate stages of transformation, or the data server has to manage (that is, store) data during processing.
- ▶ **Process transformation**
When there is a preference for using data server resources and capabilities, perhaps because suitable ETL processing does not exist.

These options have different effects on the database:

- ▶ Staging tables

For each transformation step performed within the database, you will need to design and manage a staging table for each transformation operation. However, using this approach means that the data touches disk multiple times and this can increase the overall ingest time.

- ▶ Transformation logic

You have to design and manage processing logic in the database (usually by using stored procedures).

- ▶ Recovery

Your ability to recover data is based on your backup strategy, but is also based on the number of transaction logs required to complete the recovery and any exposure to data ingested using non-logged transactions, that is, LOAD. Your ETL schedule and backup schedule must be aligned to mitigate against longer recovery times.

Chapter 8, “Building a corporate backup and recovery strategy” on page 289, discusses the use of DB2 Merge Backup in your backup strategy where full database backups are replaced by more frequent incremental backups, thereby helping to reduce the number of transaction logs required in a recovery scenario.

DB2 utilities for loading data

There are two general methods available within DB2 10.1 for loading data into your data warehouse:

- ▶ Load utility

The load utility is the fastest way to get data into the database layer. This is achieved by loading data as non-logged transactions. The load utility is ideal when loading data into staging tables where transformations can be applied to the data before it is inserted into the production database as a logged transaction, or attached to the production table as a data partition. The key items here are data availability and recoverability.

Because the load utility has full access to the table, additional features are possible. For example, GENERATED ALWAYS and SYSTEM_TIME columns can be specified in the input file. Use the load utility when you have to load data at faster speeds than the ingest utility can achieve and where other applications do not need to access the table.

- ▶ Ingest utility

The ingest utility, introduced in DB2 10, is the fastest method of getting data into the database as a logged transaction. The ingest utility is compatible with

previous versions of DB2 software because it has a client interface and is ideal when data is to be loaded directly into the production database.

Use the ingest utility when you need other applications to access the table while data is being ingested, or where you need the ability to pause and continue processing or recover from error.

Data volume and latency

The volume and velocity of data to be ingested into a data warehouse can present challenges when determining the approach to use. Identifying which one of these represents your environment is important before you begin to implement an architecture to support your data ingest needs.

Table 7-2 identifies the deployment patterns for approach to ETL development.

Table 7-2 Deployment patterns for approach to ETL development

Label	Description
Regular	Regular ingest design practices should be sufficient to meet the data volume and latency requirements.
High performance	Special high-performance design approaches are needed to meet the data volume and latency requirements.

High-performance design approaches include the need to incorporate the ability to increase parallelism and volume in all aspects of ETL component design and to take advantage of all the features available in DB2 to reduce maintenance operations that also compete for resources outside of the query workload to a minimum.

Here are a few situations where the “high performance” case is chosen:

- ▶ Late in the development process during performance testing, the team discovers that ingest service levels are not being met. Therefore, additional design to achieve higher performance is required. At this point the hardware has been purchased and it is unacceptable to request the purchase of additional hardware.
- ▶ The project team is “performance oriented”, wanting to obtain the best performance possible from the available server resources.

Populating summary (or aggregate) tables

The ingest process has to incorporate the population of all the tables that are affected by the new data, not simply the initial detail or atomic data table.

Table 7-3 on page 274 identifies the summary tables.

Table 7-3 Refreshing aggregated data tables

Label	Description	Refresh time
None	Database contains only tables with detail data	0 minutes
MQTs	Database contains MQTs that must be refreshed	n minutes
Custom aggregates	Database contains summary or aggregate tables that are custom-built (that is, not using MQT feature) and must be updated	n minutes

Tip: Include the need to refresh MQTs or custom aggregate into your ingest rate calculations and latency objectives.

7.2 Design and develop your ETL application

The discussion so far has highlighted the different data ingest patterns, objectives, and approaches. At this point you know the principle characteristics of your ETL application and have an appreciation of what the final ETL application is to look like.

When designing an ETL application, it is important to follow some basic rules:

- ▶ Design discrete ETL components to process data as soon as possible and as far as possible through the ETL process rather than waiting for all data to be available.
- ▶ Fail only rows that cannot be processed and avoid processing valid data more than once; this reduces I/O.
- ▶ Parameterize your ETL components such that multiple instances of each component can be run in parallel at peak times to increase your data ingest rate.
- ▶ Know your recovery position from a data ingest and database perspective at all points in the process.

SQL Warehousing, a feature of IBM Design Studio that is part of InfoSphere Warehouse, is a graphical tool that allows you to build and compile an executable ETL application. InfoSphere Warehouse Administration Console is used to register, deploy, and schedule the compiled SQW application.

SQW follows the control flow and data flow architecture common to other ETL tools. An Eclipse-based interface provides a drag, drop, and configure approach

to development and supports “Slowly Changing Dimensions”, “Bulk Load”, and “Surrogate key lookups” functionality as shown in Figure 7-1.

7.3 Use SQW and Bulk Load to get batch data into staging tables

In this section we show how SQW can be used to develop an ETL component to take data from a source database and bulk load the data into a target staging table.

Figure 7-1 illustrates the Integrated development environment (IDE) for SQW. The left side of the window is used to define the structure and flow of the application. The right side of the window shows a palette of configurable objects to drag and drop onto the canvas.

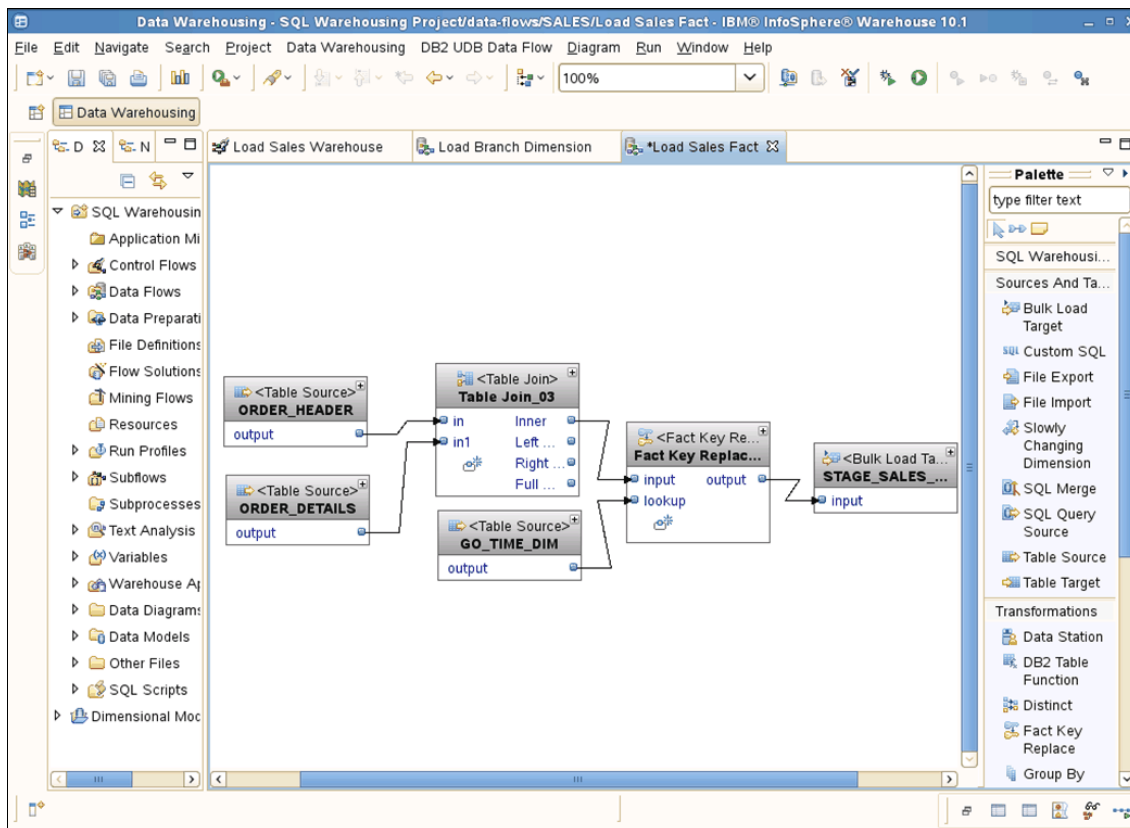


Figure 7-1 SQW data flow showing flow of data from source to target

7.3.1 SQW data flow

Figure 7-1 on page 275 shows a simple data flow that extracts sales transactions from a source database and loads that data into a target data warehouse fact table.

- ▶ A Table Source object is used and configured to select data from a source table ORDER_HEADER.
- ▶ A second Table Source object is used and configured to select data from a source table ORDER_DETAILS.
- ▶ A Table Join object is used and configured to join the ORDER_HEADER and ORDER_DETAILS data flows and provide a single data flow for each transaction within an order that is aligned with the target fact table.
- ▶ A third Table Source object is used and configured to select datetime data from the date dimension.
- ▶ A Fact Key Replace object is used and configured to replace the date in the data flow with a date key to join the target fact table to the date dimension.
- ▶ A Bulk Load object is used and configured to load the transformed data into the staging table.

The data can then be inserted into the production tables from the staging tables. In this way, the production table does not have to be taken offline, full query access is supported, and recoverability is maintained because the INSERT statement is a logged transaction.

This simple data flow represents a discrete component of what is a larger application; extracting, loading, transforming and inserting data. By developing further discrete components to process transformation and loading of dimension and additional fact data, you are positioned to perform these tasks:

- ▶ Change, debug, and monitor at a component level.
Having several smaller ETL components rather than a single larger component allows for changes to be isolated and debugged at a more granular level.
- ▶ Manage and control parallelism at a component level.
Where significant volumes of data are presented for only one of several data sources, for example, you can increase the parallelism at which data is processed for just a single data source.

You can validate, debug, or execute each component to ensure that each component processes as intended. Figure 7-2 shows the options available for a data flow component.

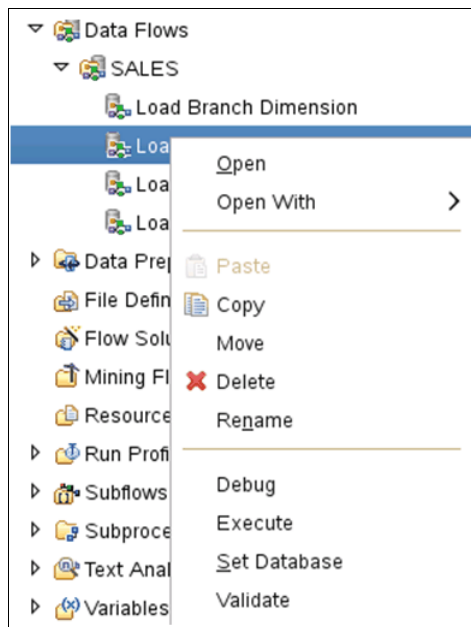


Figure 7-2 Validate, debug, and execute a data flow component

7.3.2 SQW control flow

The execution of a data flow or multiple data flows is managed through a control flow object. A typical ETL application processes dimensional data before fact data so that fact key replacement finds the most recent dimension data. Such a control flow in SQW is shown in Figure 7-3 on page 278.

A Parallel container object is used and configured to hold the Branch, Product, and Time dimension data flows. These data flows have no interdependencies and can be processed in parallel. A Data flow object is used and configured to hold the Sales Fact data flow. The fact Data flow object is dependent on the

success of the parallel dimension container because fact data might exist for a new product or time period contained in the dimension updates.

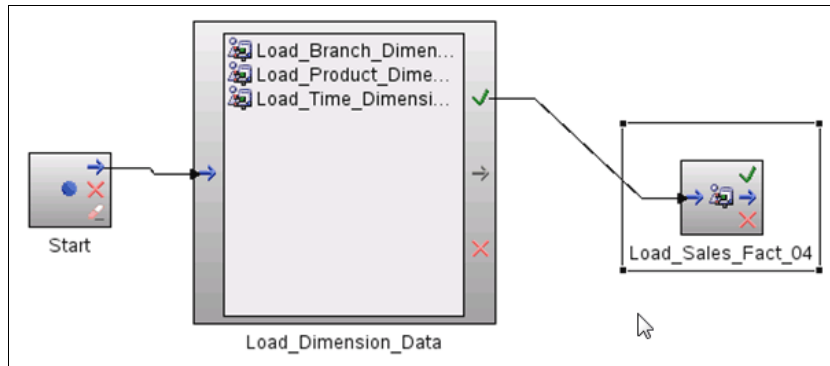


Figure 7-3 SQW control flow calling from dimension and fact data flows

For example, if a new branch is created and a transaction occurs for that branch in the same time frame, then you need the branch dimension entry to exist in the data warehouse before you attempt to assign a transaction to the branch. An alternative is to assign the fact data to a default branch and update the fact data when the late-arriving data is presented in full.

Using SQW variables

Variables, an object on the Data project explorer palette displayed on the left side of Figure 7-4 on page 279, allow you to control the behavior of the ETL component at certain points, including at run-time.

A number of reserved variables, including RUN_ID and INSTANCE_NAME, allow you to collect metadata by writing out the contents of these variables at debug time or during execution in a production environment to a database table.

The example in Figure 7-4 shows that a variable, YEAR_WANTED, is created and used to influence data ingest behavior at execution time.

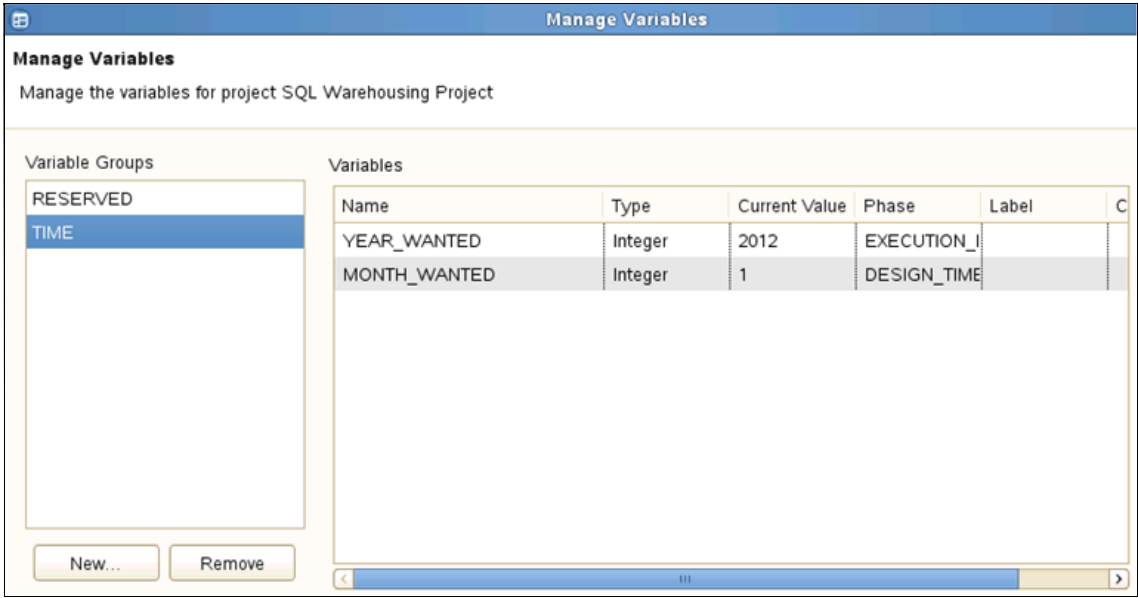


Figure 7-4 Creating ETL application variables to control execution behavior

Using the debug capability

When debugging your data flow, breakpoints can be set at the object level. Sample rows and variable state and values can be viewed when the breakpoint is reached, as shown in Figure 7-5.

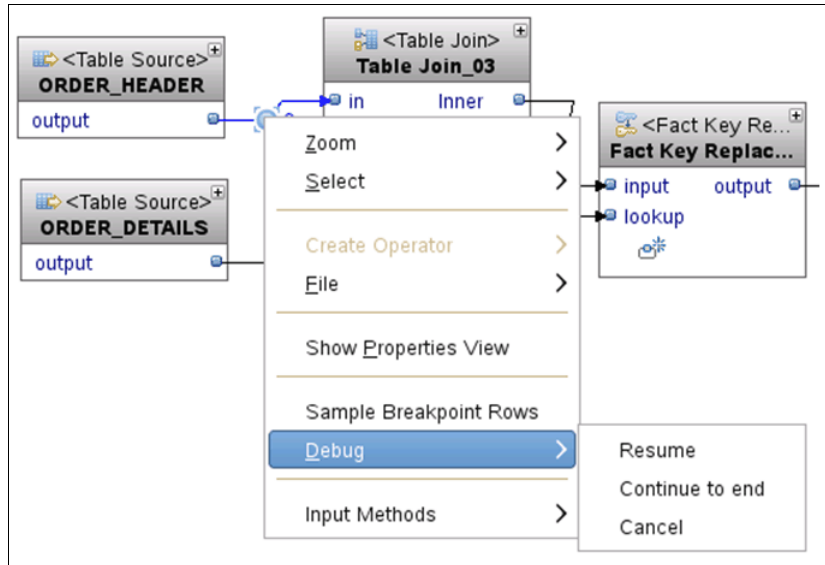


Figure 7-5 On breakpoint, options exist to Resume, Continue to end of data flow, or Cancel

7.3.3 SQW and temporal tables

SQW supports and provides tooling for slowly changing dimensions (SCDs) in the dimensional modelling sense. When implementing temporal tables as described in Chapter 5, “Temporal data management and analytics in an operational warehouse” on page 173, remove any SCD processing in your ETL engine and use the temporal functionality instead.

For example, to introduce SCD processing to a table, a database designer might add the columns `<EFFECTIVE_START_DATE>`, `<EFFECTIVE_END_DATE>`, and `<IS_CURRENT>`. An update to a row in the table using traditional SCD techniques does the following:

1. Updates the `<EFFECTIVE_END_DATE>` and `<IS_CURRENT>` column values of the existing row.
2. Generates a new row in the table with the new values.

In contrast to Figure 5-4 on page 184, Figure 7-6 on page 281 shows how a traditional SCD might look. Two rows exist for the same `CUSTOMER_ID`. The

IS_CURRENT column denotes which row is current and the data columns denote the effective start and end dates for each row.

CUSTOMER_ID	GENDER	MARITAL_STATUS	EFFECTIVE_START	EFFECTIVE_END	IS_CURRENT
1	Male	Married	2012-07-09	2012-07-10	N
1	Male	Married	2012-07-11	9999-12-30	Y
10	Male	Divorced	2012-07-09	9999-12-30	Y
12	Female	Married	2012-07-08	9999-12-30	Y
14	Male	Married	2012-07-08	9999-12-30	Y

Figure 7-6 An example of traditional or 'manual' SCD processing

When temporal data management is used, these three columns can be removed and the ETL components that manage these columns can be retired. Figure 5-4 on page 184 shows that the table columns SYS_START, SYS_END, and TRANS_START have been automatically added to a table definition in Design Studio where the System Time Period temporal attribute check box was set. Design Studio also creates the temporal history table automatically. The ETL components need only to be concerned with INSERT, UPDATE, and DELETE statements against the current row. Temporal management maintains the history changes made to each row.

7.4 Create and deploy your SQW application

After you develop, validate, debug, and compile your control and data flow components, you can create a data warehouse application. Data warehousing applications are deployed in the InfoSphere Warehouse Administration Console. The runtime environment works with WebSphere Application Server underneath.

Create SQW application

IBM Design Studio uses a wizard to help you create a new data warehousing application. The wizard prompts you to select the control flows, data flows, connections, and variables to include in the compiled application. The output of the wizard is a zipped package.

Deploy a compiled SQW application

Click **Deploy** on the SQL Warehousing tab in the InfoSphere Warehouse Administration Console to Deploy the zipped package created for your application in IBM Data Studio.

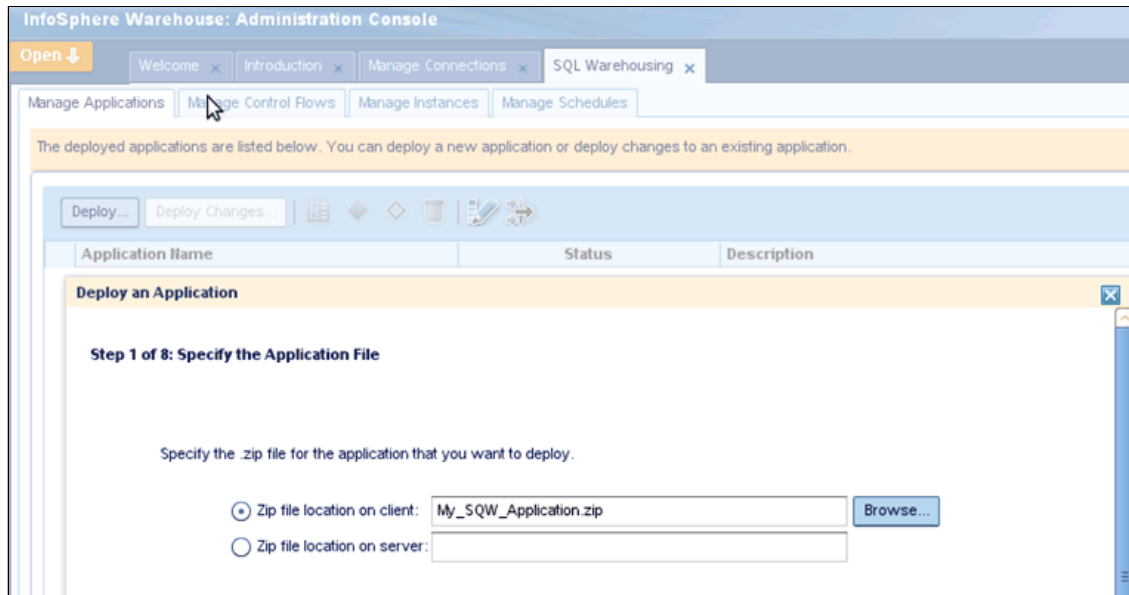


Figure 7-7 Use the Deploy application wizard to register your new SQW ETL application

The wizard also allows you to set and include variables, database connections, and system resources such as FTP servers, for the environment on which you are installing the application. This method allows you to register the same application in multiple environments without having to recompile the application.

Manage and schedule an SQW application

The InfoSphere Warehouse Administration Console is also used to run and administer the warehouse applications.

Figure 7-8 shows the example control flow application as available to be run or scheduled.

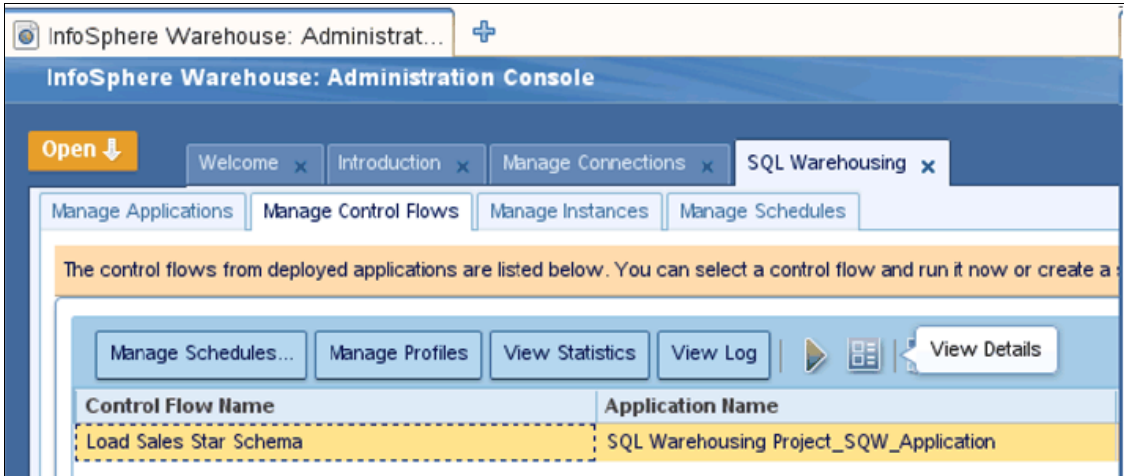


Figure 7-8 InfoSphere Warehouse Administration Console

When we run the application interactively, we are prompted to confirm or change any variables for the application; in our case it is the YEAR_WANTED as shown in Figure 7-9.

Run Control Flow: Load Sales Star Schema

Step 2 of 2: Specify the Values for the Variables

Learn More.

Specify values for the variables for this instance. If you specify a profile, the values that are associated with that profile are used. If you do not specify a profile, you can use the Value field to modify the default values.

Variable Name	Variable Type	Change Phase	Value
\$(TIME/YEAR_WANTED)	Integer	Execution	2012

1 - 1 of 1 10 < 1 >

Previous Done Cancel

Figure 7-9 Variables can be set at runtime or through the instance schedule

You can also control the level of logging required. In our example, there was no email server and so the task failed. The log details are shown in Figure 7-10.

Message Level	Time stamp	Message
SEVERE	07/13/2012 08:13:50 AM	SQW030844E: Execution of instance Load Sales Star Schema failed.
INFO	07/13/2012 08:13:50 AM	SQW03368I: Dumping variable values : \$(INT-RES/RUN-ID) = 418400 \$(RESERVED/ACTIVITY_Email_09 \$(RESERVED/ACTIVITY_TYPE) = Email \$(RESERVED/APP_NAME) = SQL Warehouse Project_SQW_Application \$(RESERVED/APP_PATH) = /home/db2inst1/HOL_Projects/apps/ \$(RESERVED/CF_ACTIVITY_NAME) = Load Sales Star Schema/Email_09 \$(RESERVED/CURRENT_TIMESTAMP) = 2012-07-13 08:13:50 \$(RESERVED/DATE) = 2012-07-13 \$(RESERVED/DAYNAME_SHORT) = Fri \$(RESERVED/DAYNAME) = Friday \$(RESERVED/DAY) = 13 \$(RESERVED/INSTANCE_NAME) = Load Sales Star Schema \$(RESERVED/MSC_DIR) = /home/ \$(RESERVED/PROJECT_NAME) = SQL Warehousing Project_SQW_Application/etl/misc/ \$(RESERVED/MONTHNAME_SHORT) = Jul \$(RESERVED/MONTHNAME) = July \$(RESERVED/MONTH) = 07 \$(RESERVED/PROC_NAME) = Load Sales Star Schema \$(RESERVED/RUN_ID) = 418400 \$(RESERVED/SCD_DATE) = \$(RESERVED/SCD_TIMESTAMP) = \$(RESERVED/START_TIMESTAMP) = 2012-07-13 08:12:53 \$(RESERVED/TIME) = 08:12:53 \$(RESERVED/TMP_DIR) = /tmp/SQL Ware Project_SQW_Application/tmp/Loa_SQW002 \$(RESERVED/USER_DIR) = /opt/IBMWebSphere/AppServer/profiles/AppSrv01 \$(RESERVED/USER_HOME) = /root \$(RESERVED/USER_NAME) = \$(RESERVED/YEAR) = 2012 \$(TIME/MONTH_WANTED) = 1 \$(TIME/YEAR_WANTED) = 2004
SEVERE	07/13/2012 08:13:50 AM	SQW03406E: Execution failed for activity Email_09
INFO	07/13/2012 08:13:50 AM	
SEVERE	07/13/2012 08:13:50 AM	SQW03203E: Email_09 task failed

Figure 7-10 InfoSphere Warehouse Administration Console - application log details

7.5 Introduce parallelism to increase ingest volumes

Split the incoming data or source files into multiple streams or directories and use variables to determine the source directory to use and the target set of staging tables to use at runtime. In this way, you can issue multiple instances of an application in parallel to increase your ingest rate.

Adhere to the following recommendations to facilitate parallelism in your data ingest process.

1. Develop separate ETL data flow components to:
 - Load data into staging.
 - Cleanse and transform data in staging tables if applicable.
 - Insert data into production tables or data partitions for subsequent attach.

2. Issue and adjust the number of instances of each component in operation to address the area experiencing a performance or data bottleneck.
 - Create separate components for separate data sources.

For example, where source files with the same format arrive from individual branches, process them separately using a different set of staging tables. Where service level objectives require, you can process the data in parallel.
 - Separate processes to select ranges of data in same staging table.

Define multiple components that uniquely identify different ranges of data and insert select in parallel.

For example, create two fact table data flow components that select from staging and insert into target production table in commit ranges of 1000 rows.

7.5.1 How Ingest and MQT tables correlate with data availability and data latency

MQTs can greatly improve query performance by pre-computing the results of a query or queries for some or all the columns in the underlying table or tables. Additional storage and resources are required to compute the query and store the results.

The DB2 optimizer rewrites all or a portion of a query to take advantage of an MQT when it is determined that the MQT offers a better execution plan. The query workload does not need to be rewritten to take advantage of the MQT. This means that you can continue to identify new and refine existing MQTs to react to changes in how data is accessed throughout the lifecycle of your data warehouse.

An MQT can be populated automatically as the underlying data changes or controlled manually; the manual method is recommended in a warehouse environment to ensure control over all aspects of data processing and minimize the number of MQT refreshes during data ingest cycles. It is best that all MQTs are refreshed as part of the ETL application to help ensure data availability in line with data refresh and service level objectives. Again, this avoids unnecessary MQT refresh operations.

The features and functionality available through MQTs is well documented elsewhere, for example:

<http://www.ibm.com/developerworks/data/bestpractices/databasesdesign/>

Example 7-1 represents one type of MQT, and shows how to create a partitioned MQT for “customer sales by product line by day.”

Example 7-1 Creating partitioned MQT

```
CREATE TABLE CSTINSIGHT.MQT_CUST_PD_SALES_DATE AS
(SELECT C.CST_NUM, C.FULL_NM, F.DATE_ID, P.PD_GRP, SUM(QTY_SOLD*UNIT_PRICE) AS
SALES_TOTAL_DAY
FROM CSTINSIGHT.CUSTOMER C, CSTINSIGHT.CUSTOMER_TXN F, CSTINSIGHT.PRODUCT P
WHERE F.PD_ID= F.PD_ID AND C.CST_NUM = F.CUST_NUM
GROUP BY F.CST_NUM, P.PD_ID)
DATA INITIALLY DEFERRED REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION MAINTAINED BY SYSTEM
COMPRESS YES
DISTRIBUTE BY HASH(PD_ID)
PARTITION BY RANGE(DATE_ID)
(PART PART_PAST STARTING(MINVALUE)
ENDING('2010-06-30') EXCLUSIVE IN TS_PROD_MQT_SALES,
PART PART_2010_Q3 STARTING ('2010-07-01')
ENDING('2010-09-30') EXCLUSIVE IN TS_PROD_MQT_SALES,
PART PART_2010_Q4 STARTING ('2010-09-01')
ENDING('2011-12-31') EXCLUSIVE IN TS_PROD_MQT_SALES,
PART PART_2011_Q1 STARTING ('2011-01-01')
ENDING('2011-03-30') EXCLUSIVE IN TS_PROD_MQT_SALES)
ORGANIZE BY (DATE_ID);
```

After the MQT table is created, perform these tasks:

1. Populate the CSTINSIGHT.MQT_CUST_PD_SALES_DATE by issuing the following SQL statement; this operation is logged in the database transaction logs.

Example 7-2 Refresh MQT table

```
REFRESH TABLE CSTINSIGHT.MQT_CUST_PD_SALES_DATE;
```

2. After populating the MQT, create an index on product and date columns to improve the performance of operations on the MQT by issuing the DDL statements shown in Example 7-3.

Example 7-3 Create index on MQT table

```
CREATE INDEX CSTINSIGHT.IDX_MQT_SALES_PD_DATE_PROD ON
CSTINSIGHT.MQT_CUST_PD_SALES_DATE (PD_ID, DATE_ID);
```

3. Collect statistics on the MQT by issuing the SQL statement shown in Example 7-4 on page 288.

Example 7-4 Issue RUNSTATS process

```
RUNSTATS ON TABLE CSTINSIGHT.IDX_MQT_SALES_DATE_PROD WITH DISTRIBUTION AND  
SAMPLED DETAILED INDEXES ALL;
```

The table created in Example 7-1 on page 287 must be refreshed each time you want to accommodate changes to the underlying tables. In addition, MQT tables can be layered. For example, to create an MQT similar to that this example but for a granularity of Month rather than day, the SELECT statement would incorporate the MQT shown here rather than the base table. In this way the number of rows to be read to produce the new MQT are significantly reduced.



Building a corporate backup and recovery strategy

In this chapter we discuss how to use IBM InfoSphere Warehouse Advanced Edition to build a backup and recovery strategy that maximizes data availability, minimizes exposure to data loss, and further leverages your backup infrastructure to deliver real value to your business.

Database backups and the associated infrastructure are often considered expensive but only because they are underutilized assets. The traditional approach to database backups is to speed the full database backup operation by introducing more hardware. However, backing up hot, warm, cold, and dormant data at the same frequency is fundamentally inefficient.

Employing software tools using the correct backup strategy allows you to achieve the same reliability with greater speed and less hardware.

Using the features of DB2 Merge Backup, DB2 Recovery Expert and High Performance Unload can help you maximize the return on your investment in backup infrastructure by being able to:

- ▶ Back up only the data that has changed.
- ▶ Restore only the data that has been lost.
- ▶ Leverage your backup infrastructure to provide data extract, data migration, and data repartitioning capability.

- Manage, monitor, and analyze your backup, recovery, and transaction positions.

Tip: Maximize the return on your backup infrastructure investment by using the tools available for that infrastructure.

8.1 Advanced features for backup and recovery

IBM InfoSphere Warehouse Advanced Edition contains a number of tools that allow you to transform how you build and implement your backup and recovery strategy and ultimately improve your ability to recover data quickly.

In an operational business intelligence (BI) environment, data is continuously loaded into the database and offline or off-peak maintenance windows are not always available. Using the tools described in this chapter enables you to:

- ▶ Eliminate full database backups.
Reduce backup operations and traffic by building full database backups from incremental and delta images without connecting to the production database.
- ▶ Restore data at granular levels to avoid unnecessary outages.
Restore data at table space, table, and row level; target and undo individual transactions within the log files.
- ▶ Perform data unload and preparation for restore on another server.
Unload data from backup images to flat files outside of the production system and then discretely restore that data to the production database, thereby minimizing or eliminating any outage.
- ▶ Generate targeted data extracts directly from your backup images.
Eliminate expensive resource usage and time-consuming workloads on the production database by creating data extracts from merged full database backup images.
- ▶ Populate development and test systems automatically.
Migrate data from full database backup images or table space containers to test or downstream environments, avoiding costly database operations and manual tasks.

8.1.1 Advanced recovery solutions explained

There are a number of tools available in IBM InfoSphere Warehouse Advanced Edition to complement the native DB2 backup, restore, and recover functions. By incorporating these tools into your backup and recovery strategy, you can help minimize potential outage caused by data loss.

DB2 Merge Backup

IBM DB2 Merge Backup for Linux, UNIX and Windows (DB2 Merge Backup) creates a complete and current full database or table space backup by combining an original full backup with subsequent incremental and delta

backups. Creating a full database backup can now take place independent of the production data warehouse. Support for IBM Tivoli® Storage Manager is included. This tool eliminates the need to take repeated full database backups, reduces backup times, and can help provide access to a current full database backup at all times.

DB2 Recovery Expert

The DB2 Recovery Expert tool interrogates your database transaction logs and backup history to help identify the restore options available to you, and even generates the undo or redo SQL necessary for recovery. In addition, DB2 Recovery Expert can analyze logs to identify which tables have been changed, and this information can be used to determine which table spaces are hot and need to be backed up. Backup images created using the DB2 Merge Backup tool are supported.

Optim High Performance Unload

IBM Optim High Performance Unload for DB2 for Linux, UNIX, and Windows (Optim High Performance Unload) allows data to be unloaded directly from backup images or from table space containers without being directed through the DB2 software layer and optimizer. This allows for increased unload speeds and avoids unnecessary effect on DB2 resources and the query workload. When used in conjunction with the LOAD or INGEST commands, application data can be unloaded and directed to the database more efficiently.

8.2 Plan a backup and recovery strategy

A recovery strategy drives a database backup plan. The most effective recovery strategy is the one that requires the least amount of data and steps to complete; only the data that is lost is recovered.

Identifying and planning for the recovery scenarios that are most likely to occur are the key factors in determining the speed with which you can recover from data loss or corruption. To begin planning your backup and recovery strategy, be aware of the following information:

- ▶ Recovery scenarios and objectives to be addressed by the backup strategy
- ▶ Strategy and schedule for loading data
- ▶ Backup and recovery infrastructure
- ▶ Data needs of downstream systems

8.2.1 Recovery scenarios and objectives

Define the objectives for recovering data for each recovery scenario you have identified. Recovery time objective (RTO) refers to the service level objective for the completion of any recovery process; that is, how long it is to take to complete the restoration of lost data. Recovery point objective (RPO) refers to the service level objective for how much data loss might be acceptable to the business in relation to achieving your RTO. Together, these objectives represent how much time you have to restore data and what amount of data loss can occur when meeting the time objective.

By documenting your recovery scenarios along with the accepted recovery objectives, you are clarifying exactly what is to be covered by your backup and recovery strategy. Table 8-1 shows how a recovery strategy can be documented. Depending on your strategy, you must choose between the features available: undo the transaction, reload the data from backup images, or perform a traditional restore from backup image. DB2 Recovery Expert can help you identify the recovery paths available.

Table 8-1 Documenting a recovery strategy by scenario

Recovery scenario	Recovery method	Impact	RTO/RPO
Data loss in HOT partitioned table	<ol style="list-style-type: none">1. Use IBM Recovery Expert for individual DML error and to help determine root cause.2. Reference temporal history where deployed to assist in “rollback.”3. Use Optim High Performance Unload to recover data from table space backup image. Recover using LOAD or Ingest, as appropriate.4. Perform full table or table space restore from table space backup image. Inform business of outage.	<ul style="list-style-type: none">► <i>Low</i>: Invalid data for a period of time. Online recovery.► <i>High</i>: Table space is offline for period of recovery.	n minutes (Use test environment to determine restore speed.)

Recovery scenario	Recovery method	Impact	RTO/RPO
Data loss in COLD partitioned table.	<ol style="list-style-type: none"> 1. Use Recovery Expert for individual DML error and to help determine root cause and recovery paths. 2. Use Optim High Performance Unload to unload data from backup image and stream to table. Use Detach and Attach to replace erroneous data partitions. 3. Perform full table or table space recovery as appropriate and inform business of outage. 	<ul style="list-style-type: none"> ► <i>Low</i>: Invalid data for a period of time. Online recovery. ► <i>High</i>: Table space is offline for period of recovery. 	<p>Data loss not acceptable.</p> <p>n minutes</p> <p>(Use test environment to determine restore speed.)</p>
Disaster Recovery	<ol style="list-style-type: none"> 1. Implement log shipping. 2. Use full database backup image as prepared by DB2 Merge Backup to restore to secondary system. Make available to business and replay ETL. 	<ul style="list-style-type: none"> ► <i>High</i>: System unavailable until restore or ETL replay completes. ► <i>Low</i>: Some data loss allowed. ETL replay to be used to regain data lost from last merged backup. 	<p>n minutes</p> <p>(Frequent tests of DR procedures should take place.)</p>

8.2.2 ETL application architecture and process schedule

The more data that exists in the data warehouse, the more actionable insight you can gain by using the tools discussed in this book. Issuing a BACKUP command that coincides with the completion of an extract-transform-load (ETL) cycle is highly productive; the recovery time will be quicker because the restore requires fewer transaction logs to be replayed.

Make assumptions at the upper end of any range given for the volume of data to be loaded for a given day; your backup and recovery strategy has to be able to cater for peak usage and for predicted data growth into the future.

Be aware of any features of the ETL application. For example, if the ETL application can identify and reprocess source data that has already been processed, then a recovery scenario might require a restore to end of backup followed by the reprocessing of source data by the ETL application. This reduces the need to restore and replay transaction logs, which can reduce outage time.

Similarly, some ETL applications in certain industry sectors have the capability of simulating or regenerating data where data loss occurs or using the ETL process to recover lost data from source files saved during the initial data load process.

In a disaster recovery environment, DB2 Merge Backup images can be used to restore from a merged database backup; the recovery can be completed by using ETL processes to complete the recovery through reprocessing ETL data. An alternate option is to use Optim High Performance Unload to unload data from table space containers where the database layer is not available.

8.2.3 Backup infrastructure

Understand the capabilities of your backup infrastructure. Include the following capabilities:

- Have enough local disk space.

Where local disk backup is your chosen strategy, have enough storage capacity on each data node for two full database backup images plus regular backups of hot table spaces per database partition. Use backup compression to minimize the actual storage required. An amount of storage capacity is also required to stage DB2 Merge Backup and DB2 Recovery Expert data where used; the amount of storage required is dependent on the features used.

- Integrate Tivoli Storage Manager (TSM) with DB2 and recovery tools.

Integration means that the database software can communicate with the storage manager to retrieve backup images and archived logs as required to perform a recovery.

- Have enough tape drives to support your chosen strategy.

A natural limit in how many database partition backup operations can write to tape in parallel is determined by the number of physical tape drives. If you have only one tape drive then you can only backup and restore one database partition at a time from tape. Carefully assess your need for both disk and tape capacity based on your recovery objectives.

- Use LAN-free.

Storage-to-storage communication of backup traffic takes place outside of the internal application network and is recommended for full and incremental backup and restore traffic to and from tape.

- Set up merge backup infrastructure.

Merge backup can integrate with TSM when creating new backup images from existing full and incremental backup images. Alternatively, backup images can be merged and recreated on disk. Merge backup supports full, incremental, and delta database, in addition to table space backups.

8.2.4 Data needs of downstream systems

The availability of data to the query workload increases the value of the data asset. The availability of the Optim High Performance Unload and DB2 Merge backup software provides many possibilities when it comes to creating data extracts for downstream systems including disaster recovery, latent replication, data marts, large report listings, and the population of development, test, and quality assurance (QA) environments.

Use Optim High Performance Unload to take cuts of data for downstream systems without affecting the production system. Incorporate these needs into the development of your backup schedule where possible to take full advantage of your backup infrastructure.

From this planning and data gathering process, have the information available to develop and implement a backup strategy that will meet your recovery objectives and fully utilize the investment in your backup infrastructure.

8.3 Implement a backup and recovery strategy

In an environment where hot, warm, cold, and dormant data exists, it is impractical from a cost or resources context to continue to perform full database backups and maintain multiple backups of data that is not going to change. In addition, restoring a table or table space from a full database backup image will take significantly longer to complete because the restore process will have to traverse the entire backup image or images to ensure that all data pages for the tables specified have been recovered.

Choose a backup strategy that is simple to implement, as explained here:

- ▶ Perform online incremental and delta database backups to local disk or to TSM using LAN-free technology.

This approach helps ensure that you only back up data that has changed on the production database, and that changes to data not picked up as part of your table space backup strategy is captured.

- ▶ Perform frequent online table space backups of individual hot table spaces to local disk.

The most likely recovery scenario is typically related to active table spaces. By having the most recent backup image available on disk, you can reduce recovery time. In addition, the merge database backup can take advantage of these table space backups.

- Perform less frequent online table space backups for groups of warm table spaces to local disk.

You do not want to have an excessive amount of backup operations processing at the same time because this increases the resources consumed by the backup workload. As the temperature of data cools, back it up less frequently.

- Use DB2 Merge Backup to recreate full database and table space backup images after each incremental or delta backup completes. This process is to be a continual cycle.

Take advantage of the backup infrastructure where possible to complete I/O-intensive operations away from the production database.

- Align backup and merge operations with your TSM disk pool to tape migration policy to avoid tape contention.

The most efficient restore process is one that uses the fewest amount of steps and references the least number of data objects.

Figure 8-1 illustrates the process of using local disk database and table space backup images to create a new merged full database backup image.

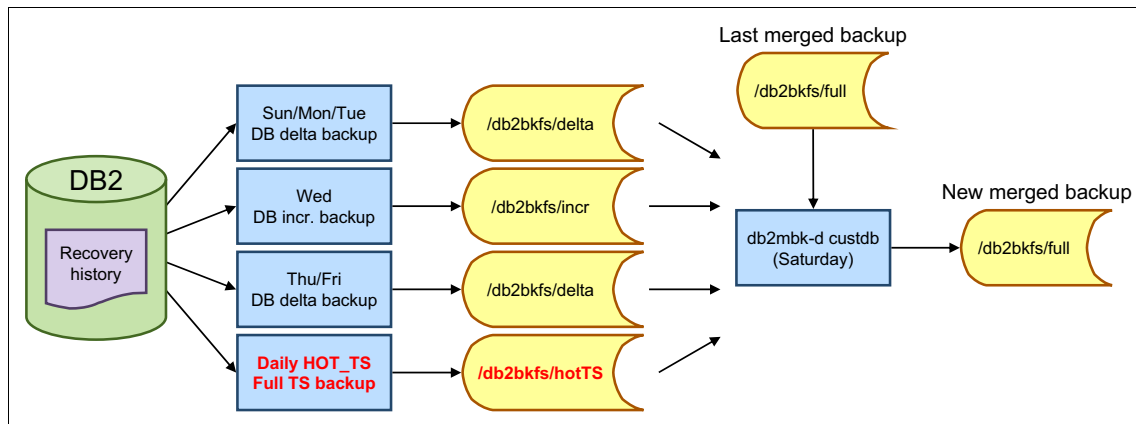


Figure 8-1 Example local disk backup strategy incorporating DB2 Merge Backup

8.4 Implementing Optim High Performance Unload as part of your recovery strategy

IBM InfoSphere Optim High Performance Unload for DB2 for Linux, UNIX, and Windows is a high-speed tool for unloading, extracting, and repartitioning data in DB2 for Linux, UNIX, and Windows databases. Optim High Performance Unload has additional uses in data migration and repartitioning, which are discussed in Chapter 9, “Managing data lifecycle with InfoSphere Warehouse” on page 335.

The database is not affected by the Optim High Performance Unload process because the Optim High Performance Unload can bypass the DB2 software layer and access backup images and table space containers directly. Table spaces do not need to be placed in an offline state, although you might want to flush data from the buffer pool to disk; remember Optim High Performance Unload does not read or replay transaction logs. Optim High Performance Unload can therefore offer advantages over other recovery methods in both data availability and processing speed.

The Optim High Performance Unload utility uses a command-line interface with a control file to perform the data unload operation. The control file is created using the following control blocks:

- ▶ GLOBAL block contains instructions for the entire control file.
- ▶ UNLOAD block contains instructions about how data is to be unloaded.
- ▶ MIGRATE block contains instructions about how data is to be migrated from one database to another, as explained in Chapter 9, “Managing data lifecycle with InfoSphere Warehouse” on page 335.

To avoid any database connections to the production database when using Optim High Performance Unload, retain a backup image of the database catalog on disk for use with Optim High Performance Unload. The best practice is to perform a full backup of the catalog database partition on a daily basis and keep a copy of the backup image on local disk.

8.4.1 Optim High Performance Unload with backup images

Optim High Performance Unload directly reads backup images created by DB2 or by DB2 Merge Backup. Use the following method when transactional integrity is required and where ETL processes are continuously in operation:

- ▶ Use online backup images only when you are certain that no transactions took place against the objects you are unloading during the most recent online backup.

- ▶ Use table space backups instead of full database backups when possible to reduce the size of the backup image you are reading and therefore enable faster data unload times.
- ▶ Include the capture of table create statements in your backup strategy. When a table is dropped, it has to be recreated before recovery using a LOAD command.

If no backup image time stamp is specified in the control file, then Optim High Performance Unload determines the most recent full backup taken and also identifies incremental and delta backup images created since the full backup.

Optim High Performance Unload temporarily stages data from the full database backup and then applies any changes to the data found in the incremental or delta backup images before creating the final output file.

Note: Additional staging space is required when data is unloaded from incremental backup images.

8.4.2 Optim High Performance Unload with backup images usage

Optim High Performance Unload can complement your existing recovery strategy in the following situations:

- ▶ The recovery time objective (RTO) is aggressive.

Optim High Performance Unload does not require you to connect through the DB2 software layer. Instead it reads data directly from the backup image, thus increasing the speed of unloading data. Use Optim High Performance Unload to unload data from a backup image to multiple output files in parallel for one or more tables. Use the DB2 Load utility to complete the recovery of data.

For example, the method of recovering each dropped table in separate sequential restore operations is not compatible with some recovery time objectives (RTO). Optim High Performance Unload can perform data unload from multiple tables in parallel and does not require you to perform a ROLLFORWARD to end of logs, thereby helping to ensure faster data recovery times.

- ▶ The database has to be available to queries during the recovery process.

Because Optim High Performance Unload unloads data directly from backup images instead of through the database engine, you can recover data for a table without the need to put a table space or database offline.

By default, Optim High Performance Unload unloads data for all tables in the table space in parallel, reducing the effect on query workload by minimizing network traffic. This approach helps ensure that backup images are read a

minimum number of times. Optim High Performance Unload uses system resources but does not require you to connect to the production database.

Data can be restored by using the LOAD command with an INSERT with SELECT statement. Alternatively, the ALTER TABLE DETACH and ATTACH command can be used to switch the recovered data partition with the corrupted data partition. These operations do not take the system offline as is required when using the RESTORE command.

- The recovery process has to take place away from the production system.

When data must be restored and validated outside of the production system, you can unload data for dropped or corrupted tables into a non-production system. This action allows you to cleanse and prepare data for load into the production system. Use this approach to minimize the effect on the production system.

When used in this way, Optim High Performance Unload must be installed on the non-production system and a copy of the database catalog backup image must be accessible. If only a single column has been compromised, use Optim High Performance Unload to unload only the primary key and the contents of the affected column.

8.4.3 Optim High Performance Unload in a production environment

Optim High Performance Unload is designed to unload data at fast speeds and all available system resources are consumed in achieving this. It is important that you configure Optim High Performance Unload to use the resources that you want Optim High Performance Unload to use. There are two principle methods of controlling the resources used by Optim High Performance Unload:

- Control parallelism with the MAXSELECTS parameter.

The MAXSELECTS parameter determines the number of SELECT statements and therefore the number of tables to be processed in parallel when unloading data from backup images. This minimizes the number of times the backup image is read but increases the resources required by the Optim High Performance Unload process.

If a value for MAXSELECTS is not specified, Optim High Performance Unload attempts to process all SELECT statements in the unload block, or all tables in a specified table space.

We recommend that you include a MAXSELECTS entry in the Optim High Performance Unload configuration file, `db2hpu.cfg`, and associate a value of 1 with the entry. This action prevents an Optim High Performance Unload process from consuming more resources than intended. Use the MAXSELECTS control file configuration parameter to restrict the number of

tables to be unloaded from in parallel where you want to reduce the resources consumed.

- ▶ Controlling processor resources with the `nbcpu` configuration file parameter

The `nbcpu` configuration parameter specifies the maximum number of threads allowed when unloading data. Default behavior is to start a work unit for each processor core. This parameter is set in the `db2hpu.cfg` configuration file and cannot be overridden in the control file.

In a scenario involving a database outage, you might want to allow Optim High Performance Unload to use all available processor resources to unload data as quickly as possible. In an environment where queries are running during recovery, configure Optim High Performance Unload to restrict the resources consumed, thus reducing the amount of processor used.

8.4.4 Optim High Performance Unload with named pipes or output files

When using Optim High Performance Unload, you can choose to unload data to output files for subsequent processing, or you can unload data directly to named pipes for immediate processing by another utility or application such as DB2 LOAD or DB2 CDI.

Optim High Performance Unload uses a staging area to stage data pages read from the backup image. Data is then written to output files in the output area or to named pipes:

- ▶ Output files: Unloading data to output files allows this process to be separated from the process of loading data, allowing for validation or postponement of the load operation.
- ▶ Named pipes: Using named pipes allows data to be unloaded from a backup image and loaded to the target database table concurrently, and uses less disk space to perform the recovery process.

To facilitate parallel unload and load operations through named pipes, create a control file to unload data for each database partition to individual named pipe file markers on the appropriate node. Then use the DB2 Load utility to load data in parallel across all the database partitions, thereby avoiding unnecessary network traffic between data nodes during the recovery process. The syntax or use of the DB2 Load utility is not altered by Optim High Performance Unload. All features and parameters of the DB2 Load utility are supported and operate normally.

When using the DB2 Load utility to complete a recovery scenario, we recommend that you perform a backup of the recovered data in line with your service level objectives for backup and recovery.

8.4.5 Optim High Performance Unload control files

Optim High Performance Unload processes one single database partition per node in sequence on each data node in parallel. All the tables in the table space unload on each data node in the following format:

```
file.NNN_<tablespace>_<schema>_<table>
```

where NNN is database partition and <schema> and <table> are the schema and table names.

When used to unload data from a backup image, the structure of the Optim High Performance Unload control file will use the GLOBAL and UNLOAD control blocks.

Global block

The global control block contains configuration data that is common to all unload blocks in the Optim High Performance Unload control file. There is only one global block per control file and the global block must be the first block in the control file.

The **db2hpu** command references a database catalog to gather details about the tables to be unloaded. Specify the USING BACKUP CATALOG clause when the objects being recovered are not available on the existing database catalog contained in the database referenced. Scenarios for this action include where objects were dropped, the catalog is corrupted, or you are executing the **db2hpu** command on a non-production system where DB2 software is not installed.

Example 8-1 illustrates the content of a control block where the database catalog backup image for the database CSTINSIGHT is to be used. The backup image for the database catalog to be used is located on disk and the backup image to use has all or partial time stamp 2012063106. A semi-colon (;) is used to terminate the statement and the block.

Example 8-1 Sample Optim High Performance Unload control file showing Global Block

```
-- Global Block specifies disk location for the database catalog
GLOBAL USING BACKUP CATALOG CSTINSIGHT from "/work0/HPU/catalog_BACKUP" TAKEN
AT 2012063106;
;
```

Unload blocks

The unload block specifies the table space or tables and the SELECT statement for which data is to be unloaded. Multiple unload blocks can be used in a single control file to perform a sequence of unload operations. Scenarios for this usage

include when unloading from multiple table spaces or from multiple table space backup images.

Optim High Performance Unload attempts to read the backup image one time per unload block. Multiple SELECT statements in a single unload block are processed in parallel subject to the MAXSELECTS parameter value.

When creating a control file in a partitioned database environment:

- ▶ Specify the USING BACKUP DATABASE clause as the last one in the UNLOAD TABLESPACE command when unloading the whole table space. Otherwise, the Optim High Performance Unload command will fail with incorrect syntax.
- ▶ Use a period (.) after the file name in the control file; this notation is required for the DB2 Load utility to load the data successfully.
- ▶ Use the OUTFILE clause in preference to the deprecated OUTPUT clause.
- ▶ Unload data for all tables needed separately; Optim High Performance Unload does not support JOINS unless a direct connection to the source database is made.

Example 8-2 illustrates how an unload block is created to unload data from a table space backup image from database partitions 1 and 9 in parallel.

Example 8-2 Sample Optim High Performance Unload control file showing Unload Block

```
-- Unload Block specifies backup image for partitions 1 and 9
-- A table space backup is used.
UNLOAD TABLESPACE
PART(1,9)
USING BACKUP DATABASE CSTINSIGHT TABLESPACE ("FACT_SALES_2007") USE TSM ON
SOURCE HOST TAKEN AT 2012063106;
-- Select statement specifies customer table
SELECT * FROM GOSALESDW.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/GOSALESDW_customer.file.%{source_node}") FORMAT DEL
;
-- Select statement specifies customer_demographics table
SELECT * FROM GOSALESDW.CUSTOMER_DEMOGRAPHICS;
OUTFILE("%{source_host}:/work%{source_node}/HPU/GOSALESDW_demographics.file.%{source_node}") FORMAT DEL
;

```

Example 8-2 incorporates these recommendations:

► **TAKEN AT**

Use this parameter to specify a partial backup image time stamp to reference multiple backup images, including incremental and delta backup images.

For example, a time stamp of 20120630 instructs the **db2hpu** command to look for all backup images on June 30, 2012. Where multiple full backup images contain the partial time stamp, Optim High Performance Unload terminates and outputs a message to that multiple backup images with the same partial time stamp were found.

To allow Optim High Performance Unload to determine the most recent backup images to use, omit the **TAKEN AT** clause.

► **ON SOURCE HOST**

Use this parameter to instruct Optim High Performance Unload to unload data in parallel to individual output files on each database partition instead of creating a single large file on the administration node.

This action increases parallelism in a shared-nothing architecture because each output file is created locally on each data node. Each data node involved recovers the data for its local DB2 database partitions to its local output path with the correct file per node specified by the **OUTFILE** key.

► **OUTFILE** with {source_host} and {source_node} keywords

Use the **OUTFILE**("%{source_host}:/OUTPUTPATH/FILE.%{source_node}") keywords to generate the output file on the correct file system associated with each database partition.

Each database partition has a separate output file created with the database partition number appended to the name of the file.

8.4.6 Create the control file and issue a db2hpu command

Follow these recommendations when creating a control file for an unload operation:

- Specify the data to be recovered in the control file by using the **WHERE** clause to specify the range of values from columns; using the **DATAPARTITION** ID to reference particular table range; and using the **PART** clause to reference particular database partitions.
- Reference a database catalog backup image that is consistent with the data to be restored. For example, if you are unloading data for a dropped table you need to reference a database catalog that has the definition of that dropped table. Using an up-to-date catalog reduces the need for Optim High Performance Unload to access the backup image through tape.

- ▶ Verify that the output paths specified in the control file for the staging and output file areas exist and are accessible to the DB2 instance user ID.
- ▶ Throttle the resources used by Optim High Performance Unload using the `nbcpu` configuration parameter to reduce the number of threads created to mitigate the effect of `db2hpu` on other workloads in a production environment.
- ▶ Control the amount of memory to be used by Optim High Performance Unload by using the `bufsize` parameter.

Issue the `db2hpu` command

The `db2hpu` command is issued as shown in Example 8-3, where `-i` is the DB2 instance name and `-f` specifies the control file to be used.

Example 8-3 Issuing the `db2hpu` command from the command line

```
# db2hpu -i instname -f controlfile
```

The control file settings override the defaults set in the product configuration file, `db2hpu.cfg`, which is located in the installation directory, typically `/opt`.

Ensure that any backup, ingest, or other operations scheduled that might affect the unload process have been altered for the duration of the unload exercise. Avoid contention for tape drives and TSM resources by ensuring that no other operations are active at the time of running Optim High Performance Unload.

8.4.7 Install and configure Optim High Performance Unload

Install and configure Optim High Performance Unload on each node where you intend to use the product. Avoid copying the installation package to each individual node by locating it on the shared `/db2home` file system.

When installed in a DB2 10.1 environment, Optim High Performance Unload can unload data from a backup image for DB2 9.5 or above.

The configuration of Optim High Performance Unload consists of three tasks:

- ▶ Creating and sharing a single configuration file across all nodes
- ▶ Creating a directory structure for Optim High Performance Unload staging and output files
- ▶ Setting configuration file parameters

Share a single configuration file across all nodes

Minimize administration of configuration files by creating a shared configuration file on the administration node and modifying the local configuration file on each

node to reference it. To create a single configuration file that is shared across all nodes on which Optim High Performance Unload is installed:

1. Make a copy of the default configuration file, customize it as required, and save it on a file system that is accessible across all nodes; using /db2home is recommended.
2. Add the dir_cfg parameter to the default Optim High Performance Unload configuration file on each data node where Optim High Performance Unload is installed to reference the shared Optim High Performance Unload configuration file.
3. Make further customizations to the referenced configuration file only.

Create a directory structure for Optim High Performance Unload staging and output files

A table in a table space that is 100 GB in size requires up to 100 GB of disk space for staging data. Additional disk space is required for staging data if incremental backup images are referenced.

Staging area

You can specify only one directory for the staging area. Create the same directory structure on each data node to accommodate staging of data. Where possible, locate the output and staging directories for each node on a separate file system on separate logical unit numbers (LUNs). Size the staging area to accommodate the largest table space in your database, and also take into account the incremental and delta backup policy.

Output area

To enable parallelism when using Optim High Performance Unload, create an output directory for each database partition. Size the output area to accommodate the largest table space in your database. Avoid using the same file systems as used by your database; you do not want DB2 to run out of space on the production database.

Set configuration file parameters

The product configuration file holds default parameter values that are referenced each time the Optim High Performance Unload utility is used. Recommendations apply to these parameters:

► nbcpu

Determine the number of work threads to be initiated by assessing the recovery time objective required and the resources available. Change the configuration parameter as required before running the **db2hpu** command.

- **stagedir**

Use separate directories for the staging area and output area where the load formatted output files are written. The stagedir parameter in the db2hpu.cfg configuration file is set to /tmp by default. Change this entry to reference a directory that exists on each data node and has enough disk space to accommodate your Optim High Performance Unload usage strategy.

- **bufsize**

Default memory usage of 8 MB per database partition is recommended for most implementations, which is calculated as 4 MB for each Optim High Performance Unload buffer pool. Increasing this configuration parameter value increases the amount of memory reserved by Optim High Performance Unload with only a marginal increase in throughput and is not recommended.

- **maxselects**

Determine the number of tables to process in parallel based on testing and the amount of resources allocated to recovery.

Optim High Performance Unload disk requirements

The staging area must be sized to accommodate the temporary files created when data is unloaded. The minimum size needed for the staging area depends on the DB2 software version and table space type, as noted:

- **The DB2 software version**

In DB2 10.1, the entire table space is staged up to the High Water Mark (HWM) for database-managed space (DMS) table spaces. Use the ALTER TABLESPACE <TSNAME> REDUCE statement to avoid unnecessary large HWM on DMS table spaces.

- **The table space type**

System-managed space (SMS) table spaces do not require the entire table to be staged; staging is complete when all table initialization data has been found.

It is best to allocate space equivalent to the size of the table space.

When multiple tables are unloaded in parallel, the Optim High Performance Unload utility attempts to minimize the number of times the backup image is accessed. Additional processing is required for objects including LOB, XML, LONG, LONG VARCHAR, and LONG VARGRAPHIC.

8.5 Example recovery scenarios using Optim High Performance Unload

The scenarios and example control files in this section illustrate how Optim High Performance Unload can form a part of your recovery strategy using the suggestions given in this document. The examples illustrate the recovery of partitioned tables through backup images stored on TSM. These examples cover:

- ▶ Recover existing partitioned tables using backup images on local disk.
- ▶ Recover dropped partitioned table using backup images on TSM and named pipes.
- ▶ Recover data using non-production system and backup images on TSM.

The backup strategy for each table space was to back up one database partition per node in parallel until the table space was backed up. In the following examples, the same sequence was applied when unloading data.

8.5.1 Recover existing partitioned tables using backup images on disk

This scenario uses Optim High Performance Unload to recover a partitioned table from online table space backup images stored on disk. This scenario applies where the table exists in the database but the contents have been compromised or corruption has occurred.

The unload block must specify each path where backup images files exist. Because the table to be recovered exists in the database, the database catalog is referenced in the global control block rather than a catalog associated with a backup image. This reduces the number of backup images to be read and therefore increases the speed of the Optim High Performance Unload operation. Example 8-4 shows the control file used in this scenario.

Example 8-4 Sample control file to unload data from disk

```
-- Global Block will use local database catalog
GLOBAL CONNECT TO TPCDS;
-- Unload Block specifies backup images for partitions 1 to 16
UNLOAD TABLESPACE
PART(1:16)
USING BACKUP DATABASE TPCDS TABLESPACE ("HASHTS") FROM
"/work1/backup", "/work2/backup", "/work3/backup",
"/work4/backup", "/work5/backup", "/work6/backup",
"/work7/backup", "/work8/backup" ON SOURCE HOST TAKEN AT 2011033108;
```

```
-- Select Blocks specifies customer table and output files
SELECT * FROM db2inst1.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/db2inst1.customer.file.%{
source_node}")FORMAT DEL
;
```

The LOAD command is issued to complete the recovery (Example 8-5).

Example 8-5 Sample control file to unload data from disk

```
db2 "LOAD from db2inst1.customer.file OF DEL INSERT
INTO db2inst1.CUSTOMER PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
PART_FILE_LOCATION /work1/HPU OUTPUT_DBPARTNUMS(1,9)"
```

```
db2 "LOAD from db2inst1.customer.file OF DEL INSERT
INTO db2inst1.CUSTOMER PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
PART_FILE_LOCATION /work2/HPU OUTPUT_DBPARTNUMS(2,10)"
```

```
-- Issue LOAD commands for all database partitions to be recovered
```

8.5.2 Recover dropped partitioned table using backup images on TSM and named pipes

This scenario used Optim High Performance Unload to recover partitioned tables from online table space backup images stored on TSM and, by using named pipes, simultaneously loaded the data into the target database table.

The control file shown in Example 8-6 was created to use in unloading the table from backup images on TSM for database partitions 1 and 9, which were the first database partitions on each data node in the test environment. Because the table had been dropped on the production database, the catalog on the catalog backup image was referenced.

The table recovery steps are listed here:

1. Create the control file.

Each unload block processed two database partitions. Example 8-6 shows the first two unload blocks for database partitions 1, 9 and 2, 10. The pattern continued until all database partitions were processed.

Example 8-6 Sample control file to unload data from disk

```
-- Global block uses backup image of catalog on disk
GLOBAL USING BACKUP CATALOG CSTINSIGHT FROM "/work0/HPU/catalog_BACKUP"
TAKEN AT 2012033108;
```

```

;
-- 1,9 specifies partitions 1 and 9
UNLOAD TABLESPACE PART(1,9)
USING BACKUP DATABASE CSTINSIGHT TABLESPACE ("FACT_TBSPC_2007") USE TSM ON
SOURCE HOST TAKEN AT 20120331080240;
SELECT * FROM GOSALESDW.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/pipe.%{source_node}")
FORMAT DEL
;
-- 2,10 specifies partitions 2 and 10 (continues to 8 and 16)
UNLOAD TABLESPACE PART(2,10)
USING BACKUP DATABASE CSTINSIGHT TABLESPACE ("FACT_TBSPC_2007") USE TSM ON
SOURCE HOST TAKEN AT 20120331084220;
SELECT * FROM GOSALESDW.CUSTOMER;
OUTFILE("%{source_host}:/work%{source_node}/HPU/pipe.%{source_node}")
FORMAT DEL
;

```

2. Create named pipes on each data node as specified in the control file.

The names in this example have the format /workN/HPU/pipe.XXX, where XXX is the appropriate database logical node number.

The named pipes were created for the database partitions to be processed by the unload block in the control file. Example 8-7 illustrates the commands used to create the named pipes in the test environment.

Example 8-7 Sample control file to unload data from disk

```

-- Data Node 1
mkfifo /work1/HPU/pipe.001
mkfifo /work2/HPU/pipe.002
mkfifo /work3/HPU/pipe.003
mkfifo /work4/HPU/pipe.004
mkfifo /work5/HPU/pipe.005
mkfifo /work6/HPU/pipe.006
mkfifo /work7/HPU/pipe.007
mkfifo /work8/HPU/pipe.008

-- Data Node 2 (Symbolic links for /work1 to /work8 created)
mkfifo /work1/HPU/pipe.009
mkfifo /work2/HPU/pipe.010
mkfifo /work3/HPU/pipe.011
mkfifo /work4/HPU/pipe.012
mkfifo /work5/HPU/pipe.013
mkfifo /work6/HPU/pipe.014
mkfifo /work7/HPU/pipe.015
mkfifo /work8/HPU/pipe.016

```

3. Execute the Optim High Performance Unload command that specifies the required control file.

Example 8-8 Sample command line to issue db2hpu with control file

```
db2hpu -f CUSTOMER_disk_pipes.ct1
```

4. Start the DB2 load operation.

Example 8-9 shows the load commands.

Example 8-9 Sample LOAD command to complete recovery

```
-- Issue LOAD commands for all database partitions to be recovered
db2 "LOAD from pipe OF DEL INSERT INTO GOSALESDW.CUSTOMER PARTITIONED DB
CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work1/HPU
OUTPUT_DBPARTNUMS(1,9)"
db2 "LOAD from pipe OF DEL INSERT INTO GOSALESDW.CUSTOMER PARTITIONED DB
CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work2/HPU
OUTPUT_DBPARTNUMS(2,10)"
```

8.5.3 Recover data using non-production database and backup images on TSM

In this scenario, the contents of a column in a production table had been compromised. Backup images of the production database were located on the TSM server. A non-production system was used to recover and prepare the data before it was reapplied to the production system.

The production environment contained two data nodes and a total of 16 database partitions. The non-production system contained one data node and eight database partitions. There were two tape drives available.

Optim High Performance Unload was installed and configured on the non-production system only and the non-production system was configured to use TSM. A copy of the backup image for the catalog partition on the production system was made available on the non-production system. Because only a single column was compromised, Optim High Performance Unload was used to unload just the primary key and the contents of the affected column.

The control file for this scenario unloaded data from each backup image in sequence. The TSMNODE parameter in the control file was used to identify the correct TSM node where the backup image file exists.

The TARGET ENVIRONMENT and TARGET KEYS control file parameters direct output data for each target database partition to a separate file. The

TARGET ENVIRONMENT parameter value specifies the instance name, database name, and the server name where the database catalog resides on the non-production system. The TARGET KEYS parameter value specifies the database partitions on which the table exists in the non-production system.

Each data node on the non-production system used /workN/HPU directory for the output files. The APPEND parameter in the OUTFILE clause ensured that each subsequent unload block did not overwrite the output data file already created.

Example 8-10 illustrates the control file for unloading data from TSM to an alternate system.

Example 8-10 Sample control file to unload data from TSM to alternate system

```
-- Global Block specifying catalog partition backup image on disk
GLOBAL USING BACKUP CATALOG CSTINSIGHT from "/work0/HPU/catalog_BACKUP" TAKEN
AT 2012033108;
;
-- Unload Block specifies backup image for database partition 1 and 2 --- from
TSMNODE tsm_datanode1
UNLOAD TABLESPACE
PART(1,2)
USING BACKUP DATABASE CSTINSIGHT TABLESPACE ("FACT_TBSPC_2007") USE TSM TSMNODE
"tsm_datanode1" TAKEN AT 2012033109;
-- Select Blocks specifies customer table and output files
SELECT * FROM GOSALESDW.CUSTOMER;
TARGET ENVIRONMENT (INSTANCE "db2inst1" ON "perfbwadm_test" IN CSTINSIGHT)
TARGET KEYS(CURRENT PARTS(1:8))
OUTFILE
("%{target_host}:/work%{target_node}/HPU/GOSALESDW.CUSTOMER.file.%{target_node}
") FORMAT DEL
;
-- Unload Block specifies backup image for database partitions 9 and 10 -- from
TSMNODE tsm_datanode2
UNLOAD TABLESPACE
PART(9,10)
USING BACKUP DATABASE CSTINSIGHT TABLESPACE ("FACT_TBSPC_2007") USE TSM TSMNODE
"tsm_datanode2" TAKEN AT 2012033109;
-- Select Blocks specifies customer table and output files
SELECT * FROM GOSALESDW.CUSTOMER;
TARGET ENVIRONMENT (INSTANCE "db2inst1" ON "perfbwadm_test" IN CSTINSIGHT)
TARGET KEYS(CURRENT PARTS(1:8))
OUTFILE
("%{target_host}:/work%{target_node}/HPU/GOSALESDW_CUSTOMER.file.%{target_node}
" APPEND) FORMAT DEL
;
;
```

Example 8-11 shows the DB2 command to load data into the non-production system.

Example 8-11 Sample command line to load data into non-production database

```
-- Issue LOAD commands for all database partitions to be recovered
db2 "LOAD from GOSALESDW.CUSTOMER.file OF DEL INSERT INTO GOSALESDWCUSTOMER
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work1/HPU
OUTPUT_DBPARTNUMS(1)"
db2 "LOAD from GOSALESDW.CUSTOMER.file OF DEL INSERT INTO GOSALESDWCUSTOMER
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART PART_FILE_LOCATION /work2/HPU
OUTPUT_DBPARTNUMS(2)"
```

Example 8-12 shows the control file for loading data from the non-production system to files in preparation for transfer to the production system. Only the column to be recovered is specified in the select block.

Example 8-12 Sample control file to extract just the data needed for restore

```
GLOBAL CONNECT TO CSTINSIGHT;
UNLOAD TABLESPACE
PART(1:8)
SELECT customer_address_id,ca_zip FROM gosalesdw.customer WHERE ca_zip LIKE
'NA%4';
OUTFILE("/work1/HPU/customer_recovered_data.csv") FORMAT DEL
;
```

The data was applied to the production system by first being loaded into a temporary table and then applied to the target table through an UPDATE statement.

8.6 DB2 Recovery Expert as part of your recovery strategy

24/7 data availability is crucial in an operational BI environment. A DBA needs many tools to analyze, perform, and respond to recovery scenarios as efficiently as possible and with minimum effect on the availability of data to the business.

DB2 Recovery Expert for Linux, UNIX, and Windows has a number of features that help you analyze database transactions and assess the recovery position of a database.

Similar to Optim Performance Manager, DB2 Recovery Expert uses an embedded web application server to support a web interface. Configuration and

session data for the databases monitored are maintained in a local datastore. Core components are installed on each monitored database.

Use DB2 Recovery Expert in a data warehouse environment when:

- ▶ Determining the most efficient technique for a recovery situation.
- ▶ Rolling back specific unwanted data changes while the system remains online.
- ▶ Analyzing database logs, providing text reports, and generating optional executable redo or undo SQL using user-specified filters and report range parameters.

Avoid using DB2 Recovery Expert to offset ETL application shortcomings. Log analysis of extended time periods where high volumes of data have been ingested into the data warehouse is not advisable.

8.6.1 Schema level repository

DB2 Recovery Expert uses a schema level repository to maintain details of changes made to database objects. This repository is referenced to generate DDL and DML for the recovery of objects.

You create and refresh the schema level repository through the schema level repository web interface. The initial build of this repository is based on a user selected backup image. You can then update this repository based on subsequent backups; DB2 Recovery Expert keeps a record of changes which can be thought of as a history of your system catalog.

The schema level repository “data store” can be located in a separate DB2 database or can be placed within an existing database as a separate schema.

8.6.2 Log analysis

DB2 Recovery Expert analyzes each archived transaction log to generate Log Analysis reports detailing changes that have occurred within specified time periods. Report filters allow you to include tables, table spaces, database partition groups, or schemas. Log analysis can use the schema level repository to reference the structure of objects that might have been dropped to build a recovery plan for the dropped table.

Redo or Undo SQL statements can be generated from the Log Analysis report and used to replicate or reverse changes that were previously made to database objects.

Log Analysis Report

Data warehouse environments, and fact tables in particular, are typically characterized by volumes of INSERT statements. UPDATE statements against fact table entries, although not unusual in some industries, can be considered rare. The scenario in this section shows how the Log analysis feature can be used to determine where an UPDATE statement was issued against the CSTINSIGHT.CUSTOMER_TXN table in error.

Figure 8-2 shows the objects page where the CUSTOMER_TXN table has been specified for inclusion in the report. In the next step, the Filters page, we only checked the box for Updates as we simply want to report on UPDATE statements issued against the fact table.

Tip: Avoid performing a Log Analysis for INSERT statements during a high volume ETL time period. Instead, choose the time period and specific statement you want to report on.

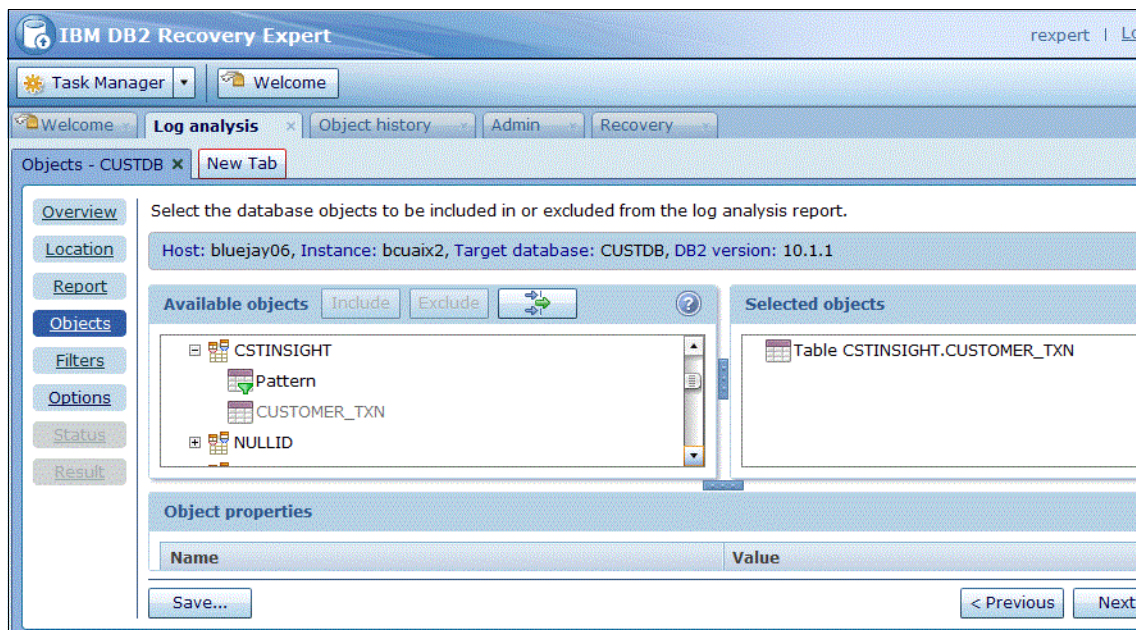


Figure 8-2 Selecting objects to be included in the Log Analysis report

When the filter is selected, click **Run** to start the Log Analysis report for the criteria entered. Figure 8-3 on page 316 shows the completed Log Analysis statistics. Two update statements were identified for the CUSTOMER_TXN table. Click **Report** to see the time each update occurred, the database partition the

updated rows resided on, and other related details. The Log Analysis results can help determine the type of recovery scenario and plan you need implement to restore or Undo the UPDATE statements.

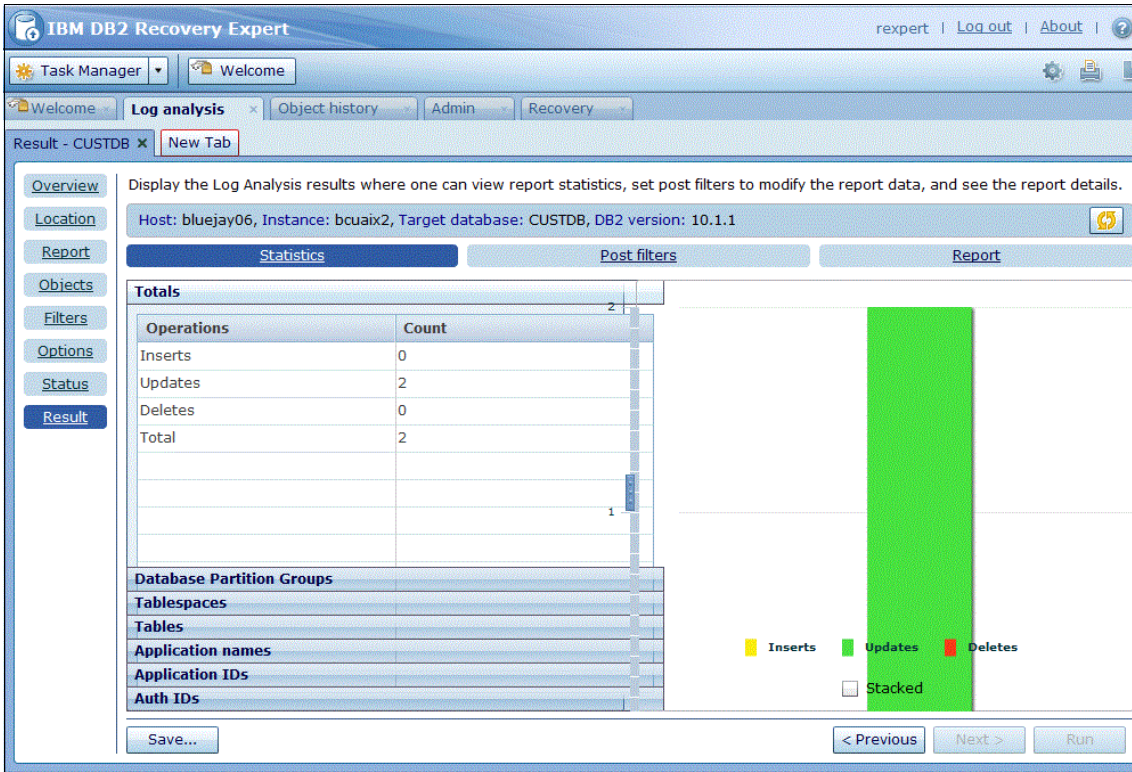


Figure 8-3 Log Analysis report filtered to report on update statements in a data warehouse environment

To UNDO the two UPDATE statements identified, we can run the Log Analysis **Summary+Detail** report with a SQL Type of **Undo SQL** as shown in Figure 8-4.



Figure 8-4 Run the Log analysis report with the Undo SQL parameter set to generate Undo SQL for the statements identified

When the Log analysis report completes, click **Export** and select the **Export to SQL** option. A dialog box is presented as shown in Figure 8-5 from which you can generate and save the SQL.

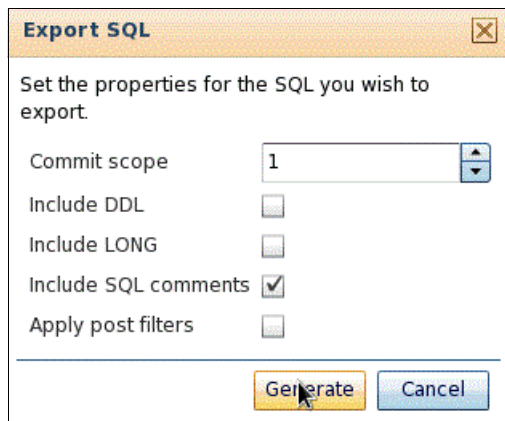


Figure 8-5 Export SQL dialog box

The generated SQL can then be reviewed, modified, tested, and run against the target database to complete the task. Example 8-13 shows the generated SQL for the 2 UPDATE statements referenced.

Example 8-13 Generated Undo SQL from Log analysis report

```
-- FILE CREATION TIMESTAMP: 2012-08-30 08:59:03
-- LOG ANALYSIS REPORT OPTIONS
-- DATABASE: CUSTDB
-- USER: bcuaix
-- DATASTORE DATABASE: DATASTOR
-- DATASTORE DATABASE USER: bcuaix
-- DATABASE PARTITION LIST: 0 1 2 3 4
-- UNDO DIRECTION SQL
-- TABLE LIST: CSTINSIGHT.CUSTOMER_TXN, CSTINSIGHT.CUSTOMER_TXN,
CSTINSIGHT.CUSTOMER_TXN, CSTINSIGHT.CUSTOMER_TXN, CSTINSIGHT.CUSTOMER_TXN
-- TID FID MAP: !0
-- ACTION: u
-- TRANSACTION STATE: cpru

-- UNDO UPDATE "CSTINSIGHT"."CUSTOMER_TXN" IN TXID 0000000DFBD8 AT LSO 000000028739FF41
-- TOP: 3 0 4
UPDATE "CSTINSIGHT"."CUSTOMER_TXN" SET "COL2" = 1001 WHERE "COL1" = 1 AND "COL2" = 1111;
COMMIT;
-- UNDO UPDATE "CSTINSIGHT"."CUSTOMER_TXN" IN TXID 0000000DFBE0 AT LSO 0000000284CC0C81
-- TOP: 3 0 2
UPDATE "CSTINSIGHT"."CUSTOMER_TXN" SET "COL2" = 1002 WHERE "COL1" = 2 AND "COL2" = 1111;
COMMIT;
```

8.6.3 Recovery

If you are performing a table space recovery and the table space already exists, you can select “End of logs” to recover the selected database objects by restoring data and processing the recovery logs through the end of the available log data. Use this option to fix a problem where the log data was correct but the database contents became physically corrupted.

To use DB2 Recovery Expert, the monitored database must be recoverable; the `logarchmeth1` database configuration parameter must be set to disk or storage manager.

The following scenario illustrates how DB2 Recovery Expert can be used to recover a dropped table. The schema level repository retains a copy of system catalog tables each time you refresh the schema level repository following backup operations. Using the Recovery option from the Task Manager menu, you can browse through and select the objects that you want to recover.

In our example, as shown in Figure 8-6 on page 319, we want to recover the schema `CSTINSIGHT` and the dropped table `CUSTOMER_TXN`. DB2 Recovery Expert correctly identifies the dependencies, table space `TA`, involved in the recovery process.

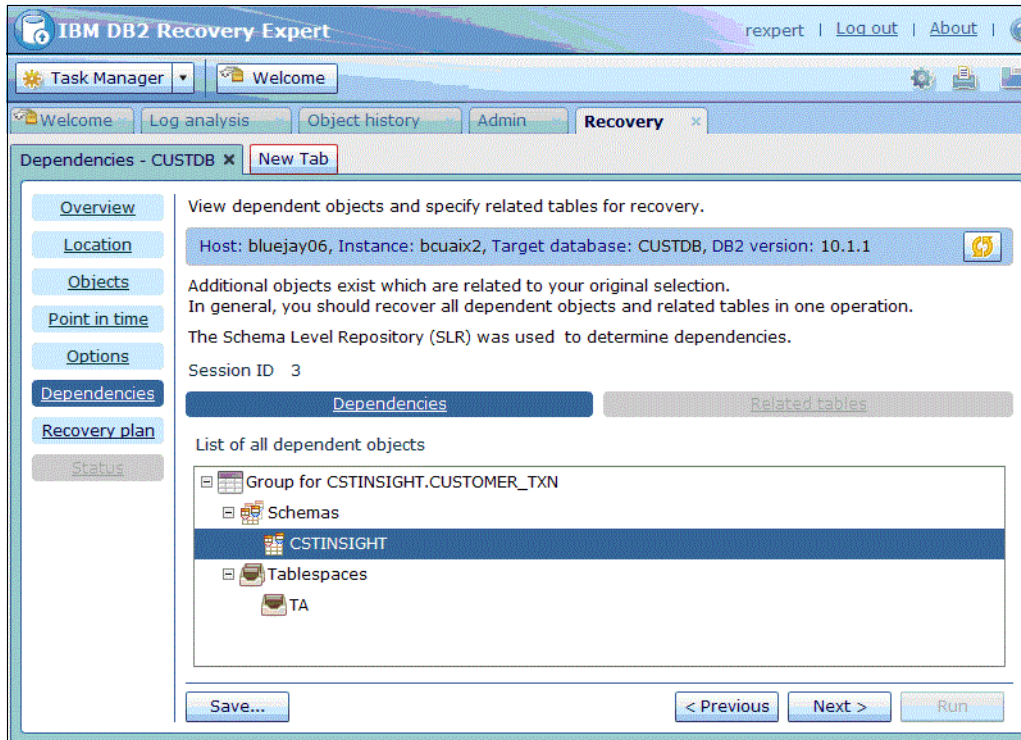


Figure 8-6 Identifying dependencies in a recovery scenario

The Next step in the process is to generate the recovery plan for the dropped tables from the DB2 Recovery Expert.

Figure 8-7 on page 320 shows the recovery plan generated for the recovery of the dropped table. The active scenario displayed looks to “Recreate object definitions from the SLR and generate Redo SQL”.

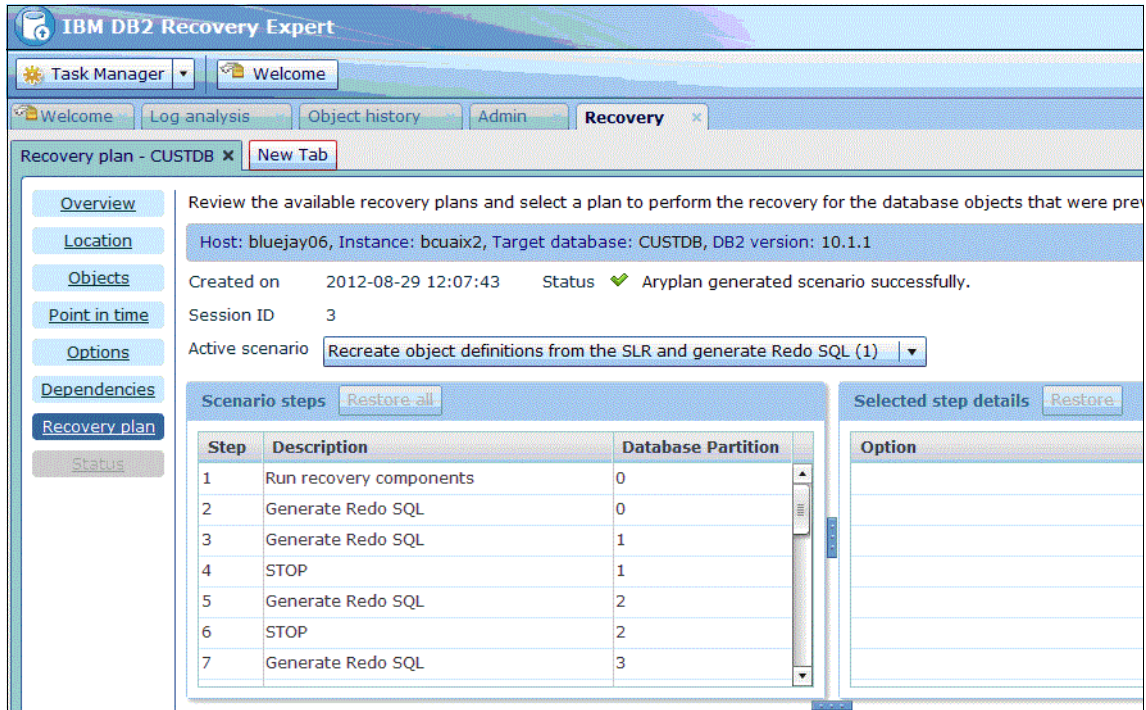


Figure 8-7 Selecting the recovery plan to restore the dropped table

When you click **Run** to select and execute the recovery plan, DB2 Recovery Expert performs the recovery steps listed to complete the recovery of data.

Figure 8-8 on page 321 shows the status page and the successful completion of the recovery operation for the CSTINSIGHT.CUSTOMER_TXN table. The status message window shows detailed, step-by-step progress through the recovery plan.

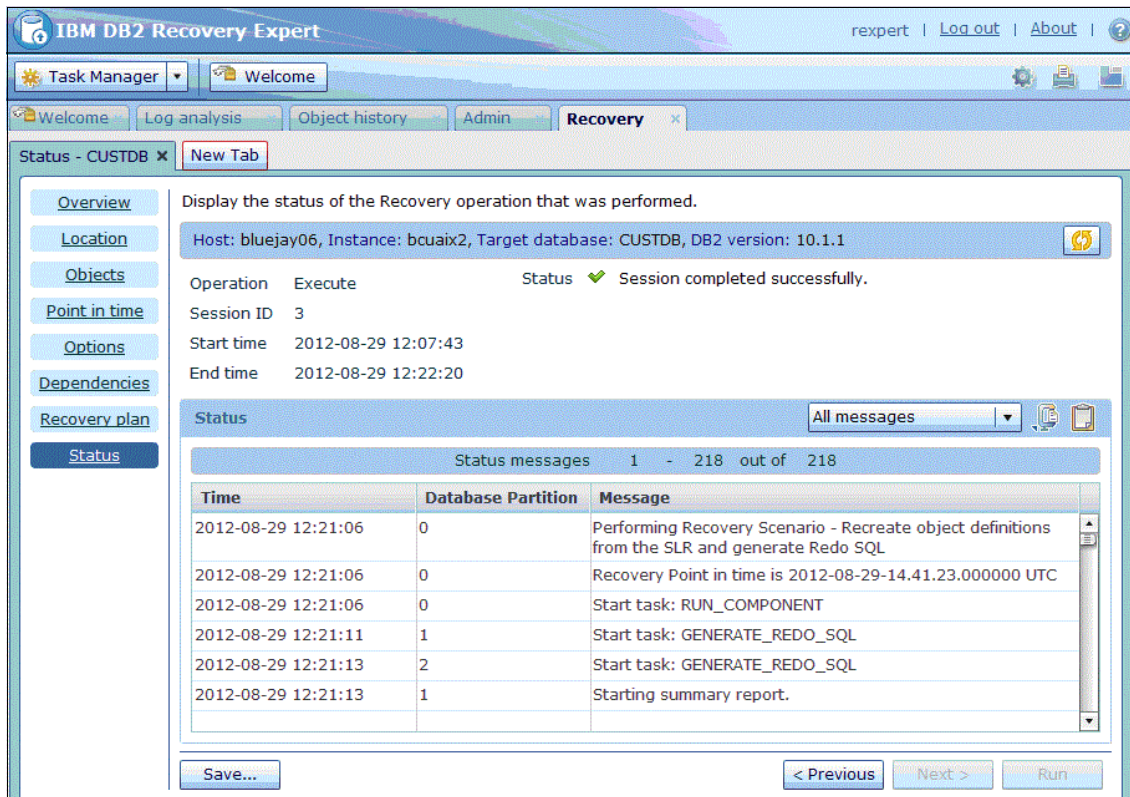


Figure 8-8 Completion of recovery operation for dropped table recovery

8.7 DB2 Merge Backup as part of your backup strategy

In this section we describe how to use DB2 Merge Backup as part of a backup strategy to reduce backup traffic and provide more flexibility in recovery scenarios. DB2 Merge Backup can create a new merged full backup image by merging a previous full backup image and any subsequent incremental, delta, and table space backup images. DB2 Merge Backup is compatible with the DB2 backup and restore commands, and all backup images create by DB2 Merge Backup are registered in the DB2 recovery history file.

The examples in this section focus on creating merged database backup images.

DB2 Merge Backup must have access to a full backup image to create a new backup image. This “base” backup image can be a merged backup.

Figure 8-9 shows how DB2 Merge Backup creates a merged backup image by merging the previous full backup image with subsequent incremental backup images.

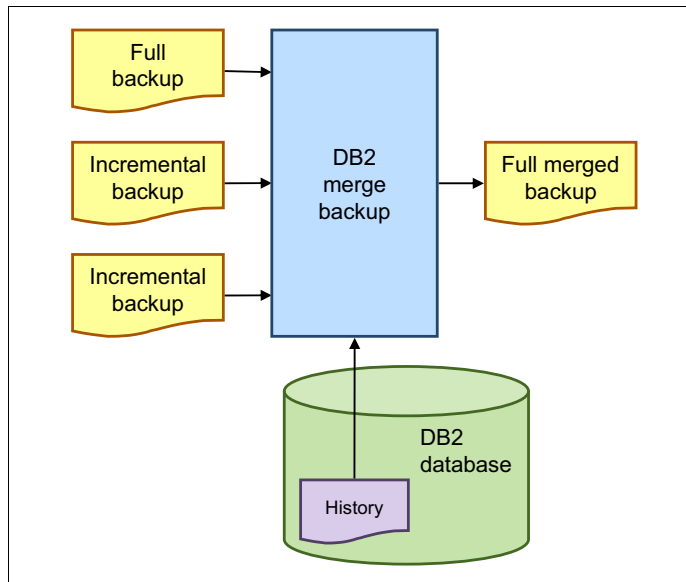


Figure 8-9 DB2 Merge backup database process

This approach means that you retain the original backup image in addition to having the merged backup image. When the merge process has completed, you can remove previous backup images as part of your database administration policy for pruning recovery history.

When you merge online and offline backups, DB2 Merge Backup determines the output backup type based on the latest input backup type. For example, if the last incremental delta backup is an online backup, DB2 Merge Backup creates a full online merged backup.

Tip: DB2 Merge Backup determines the backup type as that used for the last database backup included in the backup inventory. Where an offline backup is required prior to a system upgrade, perform an offline incremental backup and then continue to create a merged offline backup.

Table space backups are used where possible when gathering an inventory of backup images to use when creating a new merged backup.

When creating a merged backup image, DB2 Merge Backup does not support a mix of backups on disk and TSM.

DB2 Merge Backup process each database partition on a data node in sequence, but can process each data node in parallel.

8.7.1 DB2 Merge Backup as part of your infrastructure

Similar to Optim High Performance Unload, the DB2 Merge Backup tool can work in parallel across data nodes and this provides speed, scalability, and efficient use of network capacity in creating merged backup images.

The merged backup image size incorporates the size of the base full backup image plus the size of backup images to be merged. Where online backup images with logs included are merged, only the logs from the latest backup image are carried forward.

Tip: When using local disk backups, place data, log and back up files on separate storage to isolate I/O activity and increase data security.

For local disk backup strategies you must have disk storage capacity to accommodate:

- ▶ Base full backup image for each database partition.
- ▶ Merged full backup image for each database partition.
- ▶ All incremental and delta backup images to be merged for each database partition in sequence.
- ▶ All table space backup images to be merged for each database partition in sequence.

Where all backup images are on local disk, staging space needs are limited to inventory data. Inventory data is related to the number and size of backup image objects and database objects.

Tip: DB2 Merge Backup creates a backup chain for each table space in the database. By incorporating table space backups into your backup strategy, you can reduce the resources needed to complete a merged backup and increase your recovery options if data loss occurs.

For TSM backup strategies you must have disk storage capacity to accommodate the following items:

- ▶ All incremental and delta backup images to be merged for each database partition in sequence.

- ▶ All table space backup images to be merged for each database partition in sequence.
- ▶ New merged backup image is prepared on disk prior to being written back to TSM.

Where the base backup image resides on TSM, it does not need to be staged to local disk.

8.7.2 Install and configure db2mbk in partitioned database environment

Use the following guidelines when installing and configuring DB2 Merge Backup in a partitioned database:

- ▶ Install DB2 Merge Backup on each data node in the partitioned database, starting with the administration node and including each standby node.

Create a staging directory using the same directory name on each node. Only one stage directory per data node is supported. DB2 Merge Backup defaults to the /tmp file system if no staging directory is specified.

- ▶ db2mbk.cfg is the configuration file used to determine default parameter values.

The configuration file shown in Example 8-14 was used in our test environment.

Example 8-14 Contents of db2mbk.cfg in /opt/IBM/DB2TOOLS/MergeBackup11/cfg

```
# db2mbk default configuration
db2instance=db2inst1
nbcpu=1
netservice=db2mbkdm11
stagedir=/stage/stage_db2merge/
```

Where:

- nbcpu

The nbcpu configuration parameter determines the maximum number of processor threads that the DB2 Merge Backup process can use when creating a merged backup image for a database partition. DB2 Merge Backup is designed to process data as efficiently as possible using all available resources. To avoid any unintended effect on resource usage, set the nbcpu value to 1. Override where required through individual control files.

Tip: Use the `db2mbk.cfg` configuration file to set default values for parameters that help avoid any unwanted impact through not specifying these parameters in an individual control file.

- `inventory_memory`

DB2 Merge Backup gathers an inventory of backup recovery history across each database partition in the database. This parameter determines the maximum amount of memory that can be used to gather inventory data. When the maximum amount of memory has been used, DB2 Merge Backup reverts to disk and uses the staging directory specified.

- `stagedir`

Specifies the staging directory to use on each data node.

- `log_file`

Use this parameter to specify a different path for the DB2 Merge Backup log files. When you specify this parameter, make sure that the directory exists and that the user that runs DB2 Merge Backup has writing rights in that directory. By default, the DB2 Merge Backup installation directory is used.

- `sequential_access`

Use this parameter if you want to prevent DB2 Merge Backup from reading and writing on the target device at the same time. Set this parameter to YES to stage the merged backup before writing it to the target device.

8.7.3 DB2 Merge Backup command line and control files

Similar to Optim High Performance Unload, DB2 Merge Backup can be invoked directly from the command line on the coordinator node, or you can create a control file to determine the parameters to use.

Default configuration values can be overridden within an individual control file.

To build a merged full backup image from the command line for the database CSTINSDB in the instance `db2inst1`, issue the following command (as instance owner or equivalent):

```
db2mbk -i db2inst1 -d CSTINSDB
```

When performing the same task using a control file, the command looks as shown in Example .

```
# db2mbk -i db2inst1 -f create_CSTINSDB_full_backup
```

The control file would have the contents as shown in Example 8-15.

Example 8-15 Contents of control file referenced in Example 9-14

```
-- Control file contents
MERGE DATABASE CSTINSDB
OUTPUT TO "/stage/merged_backup/"
nbcpu=4
compress=yes
```

DB2 Merge Backup performs the following steps to create a backup image:

1. Access the database recovery history file to gather an inventory of backup and restore operations images available since the last base backup image.
2. Create a backup chain for each table space in the database, choosing the most efficient path.
3. Begin staging data from each backup image to be merged.
4. Create a new backup image file and begin streaming data from staging to the new backup image.
5. Repeat steps 3 and 4 for each database partition.
6. Record the merged backup image in the DB2 recovery history file.
7. Clean up the staging files used.

The recovery history file can be queried from within the database by accessing the DB_HISTORY administrative view. Example 8-16 shows a sample query that looks for all backup operations.

Example 8-16 Querying the DB2_HISTORY system administration view

```
SELECT LOCATION, DBPARTITIONNUM, SEQNUM, OPERATION, START_TIME FROM
SYSIBMADM.DB_HISTORY WHERE OPERATION = 'B' with ur
```

DB2 Merge Backup also recognizes restore operations in the recovery history file and uses these to optimize the backup chain for each table space. This helps reduce the elapsed processing time for the merged backup to be created.

Backup compression

Backup compression uses processor resources to create a compressed backup image. This has the effect of increasing processor resources required to decrease I/O; backup time increases to reduce I/O.

DB2 Merge Backup supports backup compression and by default, uses the same compression library that was used to create the backup. Use the compress keyword in the control file to determine whether the merged backup is to be compressed.

8.7.4 DB2 Merge Backup with local disk backups

In this section we discuss a simple scenario where a merged database backup was created using a base full backup image, and subsequent incremental backup and delta backup images.

You only have to perform a regular full database backup one time for your database. From this point forward, DB2 Merge Backup can refer to the previous merged backup image.

Example 8-17 shows the initial regular full database backup and also shows a backup cycle for the week that incorporates database and table space backups.

Example 8-17 Creating a merged backup strategy

```
-- Sunday. Previous full backup or merged full online backup image
db2 -v "backup db CSTINSDB on dbpartitionnums (0 to 4) online to
'/db2bkfs/full' include logs"

-- Tuesday/Thursday. Create an incremental backup image.
db2 -v "backup db CSTINSDB on dbpartitionnums (0 to 4) online incremental to
'/db2bkfs/incr' include logs"

-- Monday/Wednesday/Friday. Incremental delta backup image.
db2 -v "backup db CSTINSDB on dbpartitionnums (0 to 4) online incremental delta
to '/db2bkfs/delta' include logs"

-- Daily. Hot Table Space backups.
db2 -v "backup db CSTINSDB on dbpartitionnums (1 to 4) tablespace
(tsfactdata_2011q1) online to '/db2bkfs/hot_ts' include logs"

-- Sunday. Merge backup (Uses Full, Incr, Delta, Hot TS backup images)
db2mk -i db2inst1 -d CSTINSDB
```

In Example 8-17, a merged backup was created on Saturday.

DB2 Merge Backup uses the recovery history file to determine that the following backup images were required to create a merged backup:

- ▶ Full online database backup image from the previous Sunday
- ▶ Incremental backup image from Thursday
- ▶ Incremental delta backup image from Friday
- ▶ Table space backup for table space tsfactdata_2011q1 from Saturday

The merged backup image is an online backup image that contains the included logs from the Friday delta image only.

Recovery to end of backup and included logs is available with this backup. Archived and active transaction logs from after the last incremental delta backup image are required to complete a roll forward to end of logs.

The output from DB2 Merge Backup is verbose and lists each backup image identified as required for the merged backup. Example 8-18 shows the output for the scenario discussed in this section.

Example 8-18 Sample output from db2mbk command for disk based merge process

```

MBKM031I DB2 Merge Backup for Linux, UNIX, and Windows 01.01.100(120405) 64 bits
07/10/12 (AIX bluejay06 1 6 00CE23234C00)
-----1-----2-----3-----4-----5-----6-----7-----8-----
-9-----10-----11-----12-----13--
000001 merge database custinsdb
000002 PART(ALL)
000003 --START FROM "/stage_db2merge/" TAKEN AT 20120710065509
000004 output to "/stage_db2merge/"
000005
MBKB005I MBK control step start : 08:59:08.913.
MBKB005I [0] MBK control step start : 08:59:08.954.
MBKB006I [0] MBK control step end : 08:59:08.970.
MBKM031I [bluejay07] DB2 Merge Backup for Linux, UNIX, and Windows 01.01.100(120405) 64
bits 07/10/12 (AIX bluejay07 1 6 00CD8C944C00)
MBKB005I [bluejay07] [1] MBK control step start : 08:58:53.996.
MBKB006I [bluejay07] [1] MBK control step end : 08:58:54.074.
MBKB005I [bluejay07] [2] MBK control step start : 08:58:54.074.
MBKB006I [bluejay07] [2] MBK control step end : 08:58:54.122.
MBKB005I [bluejay07] [3] MBK control step start : 08:58:54.122.
MBKB006I [bluejay07] [3] MBK control step end : 08:58:54.169.
MBKB005I [bluejay07] [4] MBK control step start : 08:58:54.169.
MBKB006I [bluejay07] [4] MBK control step end : 08:58:54.226.
MBKB006I MBK control step end : 08:59:10.095.
MBKB053I MBK run step start : 08:59:10.095.
MBKB007I [0] MBK inventory step start : 08:59:10.095.
MBKB021I [0] The partition 0 backup image taken at 20120710085843 is involved in the
merge (type DELTA ONLINE DATABASE, device DISK)
MBKB021I [0] The partition 0 backup image taken at 20120710085322 is involved in the
merge (type INCREMENTAL ONLINE DATABASE, device DISK)
MBKB008I [0] MBK inventory step end : 08:59:10.109.
MBKB009I [0] MBK merge step start : 08:59:10.109.

```

```

MBKB028I [0] The utility will build the partition 0 backup image taken at 20120710085844
(type FULL ONLINE DATABASE, device DISK)
MBKB021I [0] The partition 0 backup image taken at 20120710084707 is involved in the
merge (type FULL ONLINE DATABASE, device DISK)
MBKB007I [bluejay07] [1] MBK inventory step start : 08:58:54.325.
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710085843 is
involved in the merge (type ONLINE TABLESPACE, device DISK)
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710085843 is
involved in the merge (type DELTA ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710085322 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device DISK)
MBKB008I [bluejay07] [1] MBK inventory step end : 08:58:54.343.
MBKB009I [bluejay07] [1] MBK merge step start : 08:58:54.343.
MBKB028I [bluejay07] [1] The utility will build the partition 1 backup image taken at
20120710085844 (type FULL ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710084707 is
involved in the merge (type FULL ONLINE DATABASE, device DISK)
MBKB010I [0] MBK merge step end : 08:59:24.175.
MBKB010I [bluejay07] MBK merge step end : 08:59:23.570.
MBKB007I [bluejay07] [2] MBK inventory step start : 08:59:23.570.
MBKB021I [bluejay07] [2] The partition 1 backup image taken at 20120710085843 is
involved in the merge (type ONLINE TABLESPACE, device DISK)
MBKB021I [bluejay07] [2] The partition 2 backup image taken at 20120710085322 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device DISK)
MBKB008I [bluejay07] [2] MBK inventory step end : 08:59:23.579.
MBKB009I [bluejay07] [2] MBK merge step start : 08:59:23.579.
MBKB028I [bluejay07] [2] The utility will build the partition 2 backup image taken at
20120710085844 (type FULL ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [2] The partition 2 backup image taken at 20120710084707 is
involved in the merge (type FULL ONLINE DATABASE, device DISK)
MBKB010I [bluejay07] MBK merge step end : 08:59:54.900.
MBKB007I [bluejay07] [3] MBK inventory step start : 08:59:54.907.
MBKB021I [bluejay07] [3] The partition 1 backup image taken at 20120710085843 is
involved in the merge (type ONLINE TABLESPACE, device DISK)
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710085322 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710085843 is
involved in the merge (type DELTA ONLINE DATABASE, device DISK)
MBKB008I [bluejay07] [3] MBK inventory step end : 08:59:55.092.
MBKB009I [bluejay07] [3] MBK merge step start : 08:59:55.092.
MBKB028I [bluejay07] [3] The utility will build the partition 3 backup image taken at
20120710085844 (type FULL ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710084707 is
involved in the merge (type FULL ONLINE DATABASE, device DISK)
MBKB010I [bluejay07] MBK merge step end : 09:00:24.093.
MBKB007I [bluejay07] [4] MBK inventory step start : 09:00:24.094.
MBKB021I [bluejay07] [4] The partition 1 backup image taken at 20120710085843 is
involved in the merge (type ONLINE TABLESPACE, device DISK)
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710085843 is
involved in the merge (type DELTA ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710085322 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device DISK)
MBKB008I [bluejay07] [4] MBK inventory step end : 09:00:24.284.
MBKB009I [bluejay07] [4] MBK merge step start : 09:00:24.284.

```

```
MBKB028I [bluejay07] [4] The utility will build the partition 4 backup image taken at
20120710085844 (type FULL ONLINE DATABASE, device DISK)
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710084707 is
involved in the merge (type FULL ONLINE DATABASE, device DISK)
MBKB010I [bluejay07] MBK merge step end : 09:00:50.544.
MBKB054I MBK run step end : 09:01:06.417.
MBKI441I MBK successfully ended
```

8.7.5 DB2 Merge Backup with Tivoli Storage Manager

DB2 Merge Backup supports Tivoli Storage Manager (TSM) and DB2 software integration with TSM. This allows DB2 Merge Backup to interface with TSM when creating a merged backup image. The merged backup image is written back to TSM. Your implementation of TSM policy determines the behavior; discuss this with your TSM administrator.

DB2 Merge Backup locates and stages each backup image required to create the merged backup on disk before writing back to TSM. To help avoid contention for tape drives, DB2 Merge Backup accesses the backup images on TSM using the same pattern in which they were created. This is done in an attempt to avoid (tape drive) contention on TSM, and assumes that the original backup pattern took the most efficient approach to resource usage on TSM. This approach also helps ensure an efficient use of staging area disk storage capacity.

Avoid contention for tape drives by scheduling DB2 Merge Backup tasks at a separate time to that of other operations, such as incremental backup, that write to tape.

Tip: Avoid contention for tape drives when using DB2 Merge Backup in conjunction with TSM; ensure that no other backup or restore operations are running at the same time as db2mbk.

Note that where TSM volume read and write is not enabled, DB2 Merge Backup has to stage the new full database backup image to disk before writing back to tape on TSM. Discuss this configuration with your TSM administrator.

Compressed backup images on TSM must be uncompressed before they can be merged with the new merged backup.

Example 8-19 on page 331 shows the initial regular full database backup to TSM and also shows a backup cycle for the week.

Example 8-19 Creating a merged backup

```
-- Saturday. Initial Full database backup (which could also be the last merged
backup)
time db2 "backup database CUSTINSDB on all dbpartitionnums online USE TSM "

-- Incremental delta backup. Sunday/Monday/Tuesday/Thursday/Friday.
time db2 "backup database CUSTINSDB on all dbpartitionnums online incremental
delta USE TSM include logs"

-- Incremental backup. Wednesday.
time db2 "backup database CUSTINSDB on all dbpartitionnums online incremental
USE TSM include logs"

-- Merge backup. Saturday
db2mbk -i db2inst1 -f merge_TSM_backup.ctl
```

Example 8-20 shows the contents of the control file required to create a new merged online database backup using TSM.

Example 8-20 Control file for merged TSM backup

```
-- Control file contents
MERGE DATABASE CSTINSDB
OUTPUT USE TSM
```

Example 8-21 shows the output data from db2mbk when creating the merged backup for the Example 8-19 scenario.

Example 8-21 Creating merged backup using TSM

```
MBKM031I DB2 Merge Backup for Linux, UNIX, and Windows 01.01.100(120405) 64 bits
07/10/12 (AIX bluejay06 1 6 00CE23234C00)
-----1-----2-----3-----4-----5-----6-----7-----8-----
-9-----10-----11-----12-----13--
000001 merge database CUSTINSDB
000002 OUTPUT USE TSM
000003
MBKB005I MBK control step start : 09:46:45.653.
MBKB005I [0] MBK control step start : 09:46:45.690.
MBKB006I [0] MBK control step end : 09:48:12.937.
MBKB006I [bluejay07] [1] MBK control step end : 09:47:57.297.
MBKB005I [bluejay07] [2] MBK control step start : 09:47:57.297.
MBKB006I [bluejay07] [2] MBK control step end : 09:49:20.486.
MBKB005I [bluejay07] [3] MBK control step start : 09:49:20.486.
MBKB006I [bluejay07] [3] MBK control step end : 09:50:33.054.
MBKB005I [bluejay07] [4] MBK control step start : 09:50:33.054.
MBKB006I [bluejay07] [4] MBK control step end : 09:51:52.536.
MBKB006I MBK control step end : 09:52:08.409.
```

```

MBKB053I MBK run step start      : 09:52:08.409.
MBKB007I [0] MBK inventory step start : 09:52:08.409.
MBKB007I [bluejay07] [1] MBK inventory step start : 09:51:52.636.
MBKB021I [0] The partition 0 backup image taken at 20120710094435 is involved in the
merge (type DELTA ONLINE DATABASE, device TSM)
MBKB021I [0] The partition 0 backup image taken at 20120710093737 is involved in the
merge (type INCREMENTAL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710094435 is
involved in the merge (type DELTA ONLINE DATABASE, device TSM)
MBKB008I [0] MBK inventory step end   : 09:52:09.392.
MBKB009I [0] MBK merge step start    : 09:52:09.392.
MBKB028I [0] The utility will build the partition 0 backup image taken at 20120710094436
(type FULL ONLINE DATABASE, device TSM)
MBKB021I [0] The partition 0 backup image taken at 20120710092603 is involved in the
merge (type FULL ONLINE DATABASE, device TSM)
MBKB008I [bluejay07] [1] MBK inventory step end   : 09:51:53.661.
MBKB009I [bluejay07] [1] MBK merge step start    : 09:51:53.661.
MBKB028I [bluejay07] [1] The utility will build the partition 1 backup image taken at
20120710094436 (type FULL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [1] The partition 1 backup image taken at 20120710092603 is
involved in the merge (type FULL ONLINE DATABASE, device TSM)
MBKB010I [bluejay07] MBK merge step end      : 09:54:20.247.
MBKB007I [bluejay07] [2] MBK inventory step start : 09:54:20.269.
MBKB021I [bluejay07] [2] The partition 2 backup image taken at 20120710094512 is
involved in the merge (type DELTA ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [2] The partition 2 backup image taken at 20120710093809 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device TSM)
MBKB008I [bluejay07] [2] MBK inventory step end   : 09:54:21.830.
MBKB009I [bluejay07] [2] MBK merge step start    : 09:54:21.830.
MBKB028I [bluejay07] [2] The utility will build the partition 2 backup image taken at
20120710094513 (type FULL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [2] The partition 2 backup image taken at 20120710092755 is
involved in the merge (type FULL ONLINE DATABASE, device TSM)
MBKB010I [0] MBK merge step end      : 09:54:39.960.
MBKB010I [bluejay07] MBK merge step end      : 09:56:09.898.
MBKB007I [bluejay07] [3] MBK inventory step start : 09:56:09.920.
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710094546 is
involved in the merge (type DELTA ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710093843 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device TSM)
MBKB008I [bluejay07] [3] MBK inventory step end   : 09:56:10.993.
MBKB009I [bluejay07] [3] MBK merge step start    : 09:56:10.993.
MBKB028I [bluejay07] [3] The utility will build the partition 3 backup image taken at
20120710094547 (type FULL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [3] The partition 3 backup image taken at 20120710092928 is
involved in the merge (type FULL ONLINE DATABASE, device TSM)
MBKB010I [bluejay07] MBK merge step end      : 09:57:59.978.
MBKB007I [bluejay07] [4] MBK inventory step start : 09:58:00.002.
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710093910 is
involved in the merge (type INCREMENTAL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710094613 is
involved in the merge (type DELTA ONLINE DATABASE, device TSM)
MBKB008I [bluejay07] [4] MBK inventory step end   : 09:58:01.160.
MBKB009I [bluejay07] [4] MBK merge step start    : 09:58:01.160.

```

```
MBKB028I [bluejay07] [4] The utility will build the partition 4 backup image taken at
20120710094614 (type FULL ONLINE DATABASE, device TSM)
MBKB021I [bluejay07] [4] The partition 4 backup image taken at 20120710093103 is
involved in the merge (type FULL ONLINE DATABASE, device TSM)
MBKB010I [bluejay07] MBK merge step end : 10:00:53.354.
MBKB054I MBK run step end : 10:01:09.263.
MBKI441I MBK successfully ended
```

The production database backup strategy is now reduced to incremental and delta database operations and strategic table space backup operations. Tape drive contention is avoided by scheduling the merge operation on a separate day from the backup operations. Because the backup operation is recorded in DB2, Optim High Performance Unload can now be used to generate data extracts from the full online database backup image.

Simple merge backup using a new output location

The merged backup image can be redirected to an alternate storage path. This can be useful if storage capacity on existing paths is a concern. For example, backup images on TSM can be merged and written to disk storage as shown in Example 8-22.

Example 8-22 Redirecting output

```
MERGE DATABASE CSTINSDB
USE TSM OPEN 2 SESSIONS
OUTPUT
TO "/db2bkfs/new0", "/db2bkfs/new1"
```



Managing data lifecycle with InfoSphere Warehouse

In this chapter we describe how to manage data through the data lifecycle so that service objectives can be adhered to in a cost-effective manner.

The data lifecycle relates to how data is stored, maintained, accessed, and archived while relevant to your business. The frequency with which data is accessed and why data is accessed changes as the data ages within the database. The volume of activity associated with data is referred to as *data temperature* and the temperature of data cools as it ages.

The value of data as an asset to the business depreciates over time. In this context the cost of maintaining and accessing data as it ages should decrease so that the total cost of ownership for your data warehouse is effectively managed.

9.1 The value of data and its age

The most valuable data in a data warehouse is data that delivers actionable insight to the business. In an operational business intelligence (BI) environment, this is recent or low latency data. The ability to gain actionable insight from low latency data increases as the volume of low latency data increases. Because the data sample is greater, more opportunities exist to find value through identifying patterns, seeing trends, predicting outcomes with greater accuracy, comparing data sets, and applying data mining techniques.

By effectively managing the data lifecycle, you can help enable greater volumes of data to be ingested at greater speeds, which delivers an increased return on investment to your business. The challenge is to maximize the ingest rate. To meet this challenge, you must have the capability to organize, segregate, and isolate data that is “cooling off” to help ensure that priority operational queries are as efficient as possible.

As discussed in Chapter 7, “Understand and address data latency requirements” on page 267, reducing the time latency between business events occurring and the ability to take actionable insight based on data related to the event represents a significant value proposition to your business.

The technical challenge is to store and maintain data to facilitate queries in a way that maximizes performance and minimizes cost. This can be achieved by distinguishing data as it ages.

The concept of data temperature and a multi-temperature database, although unique to each environment, can be generally described by the patterns shown in Table 9-1.

Table 9-1 Data temperature patterns

Data temperature	Data temperature characteristics	Typical data age	Data maintenance
Hot	Tactical and OLTP-type data; that is, current data that is accessed frequently by queries that must have short response times, for example, high volume, small result set point queries in operational data stores (ODS).	0 - 3 months and aggregates or summaries of this data.	Data is located on the fastest storage and is updated frequently. Frequent table space backups are taken to aid fast recovery if needed.

Data temperature	Data temperature characteristics	Typical data age	Data maintenance
Warm	Traditional decision support-type data; that is, data that is accessed less frequently and by queries that most likely do not require short response times.	3 - 13 months and aggregates or summaries of this data.	As data ages it is accessed less frequently and cools; it is moved to less expensive “warm” storage. The warm data is updated less frequently, and so less frequent table space backups are taken.
Cold	Deep historical and legacy data; that is, data that is typically accessed infrequently.	13 months - 5 years.	Data is placed on the least expensive storage available on the production system. Infrequent updates to data are captured by less frequent database incremental and delta backup operations.
Dormant	Regulatory type or archival data; that is, data that is accessed infrequently and that is never updated.	Over 5 years.	Data is archived to backup, file, or to a federated database on inexpensive hardware.

There are always exceptions when determining the temperature of data and these exceptions are different in each environment:

► Active data

Certain date-based events such as the new year might result in data for that period being accessed more often; this is referred to as active data.

► Priority workloads

Some queries or applications might have a greater priority than others in your environment, even though the data accessed might not be the most current data.

The increasing role of corporate governance and regulations can dictate that data is retained long after the business use of the data has expired and this need extends the data lifecycle. Developing a strategy to archive and offline data to alternative storage points where it can be easily retrieved where required is crucial.

9.2 Manage the cost of storage with multi-temperature management

Managing a data warehouse with terabytes and petabytes of data requires clear objectives followed by careful planning to ensure that all aspects of data warehouse design are aligned. InfoSphere Warehouse provides a number of features that are strategically aligned to the objective of effectively managing storage costs. These include:

- Table partitioning

Often referred to as range partitioning, table partitioning allows you to partition your table into ranges of data. A common implementation is to partition data by date range and assign each range (data) partition to a separate table space which can be independently managed.

For example, a fact table might be partitioned by year; all data for a given year is placed in a single data partition. Each data partition for a table can be located in a separate table space and each table space can be placed on separate storage. This approach can increase performance by optimizing the number of rows read per result rows returned because the optimizer can eliminate entire data partitions where the predicate (where clause) specifies a particular year. Maintenance operations such as backup can then be targeted at active data partitions, reducing the time and cost of ownership.

- Storage groups

A storage group is a method of grouping one or more storage paths where table data is placed. You can use storage groups to label different types of storage according to storage speed, thus creating a multi-temperature database.

For example, separate storage groups can be created and given storage paths on different storage tiers: SSD storage might be assigned to a Hot storage group, magnetic disk storage to a warm storage group, and SATA drives to a cold storage group. Storage group functionality allows you to asynchronously migrate a table space from one storage group to another without affecting the availability of the data.

This approach to storage management allows you to structure your approach to how data is aged at a (data partition) table space level. Fast storage can be maintained for hot data only. Slower storage can support cooler data.

Using this approach you can infer that analytics queries that access cooler data take longer to process, and operational queries that access warmer data are more responsive.

- **DB2 Workload Manager**

DB2 Workload Manager (DB2 WLM) can be configured and used in conjunction with storage group data tags to manage the resources assigned to workloads that access data in a multi-temperature database.

For example, by creating a tag on a storage group, DB2 WLM can be configured to assign a workload to a certain service subclass depending on the storage group accessed, and then effect the resources and concurrency rules applied to that workload. Using this approach you can change the workload priority of a query based on the storage groups it touches. This gives the administrator more control over queries that might access both hot and cold data.

- **Temporal data**

Creating a temporal table allows you to pass the processing task of maintaining dimensional history over to the database engine. Deploying temporal features for dimension tables further enhances your ability to manage the data lifecycle, because temporal history data can be managed separately from active data.

A change to a data row in a temporal table generates data in the associated temporal history table. The temporal history table can be physically implemented independently of the parent table. This means that it can be partitioned, stored, and maintained (pruned) separately based on your requirements.

For example, although a parent table might be range partitioned, a history table with a smaller volume of rows does not have to be range partitioned, correlating the maintenance effort with the table volume. Similarly, data can be archived from the history table independently of the parent table.

- **Federated database**

In situations where you need to archive aged or inactive data out of the primary production database but retain data availability, you can migrate the data to a federated database from where it can be referenced.

9.2.1 Using multi-temperatures features in a sample scenario

This section describes how to implement the concepts discussed to effectively manage the cost of storage. The example uses a standard FACT table, which is typically the largest table in a data warehouse.

The target environment has three distinct types of storage available to the database which represent distinct cost groups for storage. The object of the exercise as discussed in this chapter is to place data for the fact table on the appropriate storage based on age. Figure 9-1 on page 340 shows the target

environment where SSD, SAS, and SATA storage is utilized in placing data for a single fact table across multiple storage tiers.

The approach to the data lifecycle is through the implementation of storage group, table partitioning, and table space objects for a sales transactions fact table. Each data partition and table space is designed to hold data for a calendar quarter. Data for the current quarter is always maintained on the fastest storage. As data ages, the table space associated with the data partition is moved to warm storage and then cold storage.

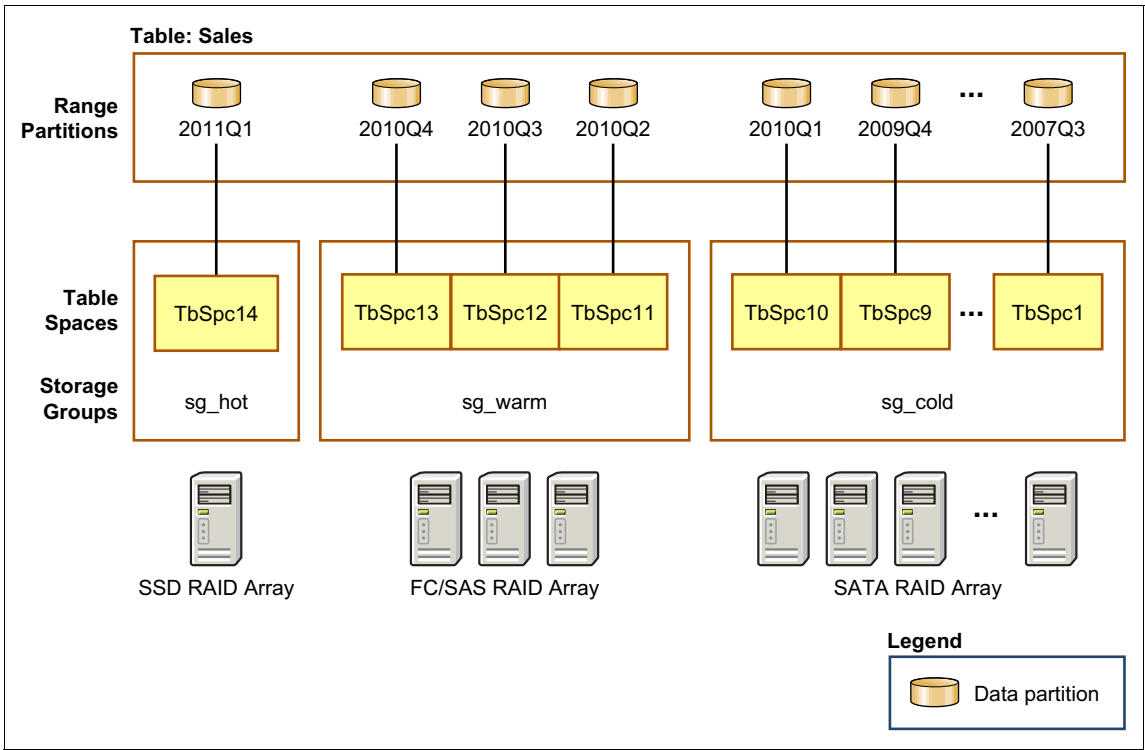


Figure 9-1 Physical implementation of a fact table to support multi-temperature storage through the data lifecycle

The following steps show how this environment was created, using the example of the sales transactions table CUSTOMER_TXN as shown in Figure 9-1. Each row in the table will be placed in a data partition, table space and storage group based on the transaction date.

1. Create a storage group.

A storage group is characterized by having one or more storage paths and performance statistics for the DB2 optimizer to reference. Using IBM Data

Studio, a separate storage group for hot, warm, and cold storage was created where the storage paths were already created on the target storage subsystems.

Each storage group is assigned an OVERHEAD and DEVICE READ RATE value. These values help the DB2 optimizer understand the speed of the underlying storage when compiling an access plan. The values for OVERHEAD and DEVICE_READ_RATE can reflect the actual storage capability, or you can alter these to reflect the temperature of data that you intend to place on that storage.

For example, in an environment where all storage devices are the same but where you still want DB2 to compile different access plans based on the temperature of data, you can create different storage groups and assign different parameter values to reflect temperature you want to assign to the data on that storage device. Refer to the DB2 Information Center for a complete description:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.perf.doc%2Fdoc%2Fc0005051.html>

The hot storage group is set as the default storage group for the database; any new table space created is assigned to the default hot storage group. Figure 10-2 shows hot, warm, and cold storage groups as created in our test environment.





	Name	Overhead	Device Read Rate	Data Tag	Default
	IBMSTOGROUP	6.725	100.0	NONE	false
	SG_WARM	6.725	100.0	3	false
	SG_COLD	7.5	75.0	5	false
	SG_HOT	0.75	350.0	1	true

Figure 9-2 Storage groups represented in IBM Data Studio

When you deploy these changes, click **Advanced Options** and include the tasks of flushing the package cache and issuing the RUNSTATS command

so that the optimizer can build a new access plan for queries that access these objects. Figure 9-3 shows an example.

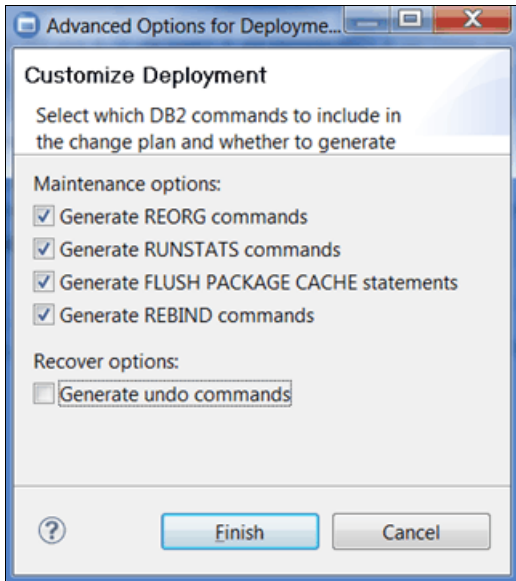


Figure 9-3 Advanced options available when generating DDL in IBM Data Studio

- 2. Create table spaces for each data partition to be created.

Using Data Studio, create a table space for each data partition to be attached to the table. Figure 9-4 shows an excerpt of table spaces created for each quarter of 2010 and Q1 2011.










Name	Buffer Pool	Page Si...	Size	Auto Resize
 SYSCATSPACE	IBMDEFAU...	4096	0	true
 SYSTOOLSPACE	IBMDEFAU...	4096	0	true
 USERSPACE1	IBMDEFAU...	4096	0	true
 TSFACTDATA_2010Q4	IBMDEFAU...	4096	0	true
 TSFACTDATA_2011Q1	IBMDEFAU...	4096	0	true
 TSFACTDATA_2010Q2	IBMDEFAU...	4096	0	true
 TSFACTDATA_2010Q3	IBMDEFAU...	4096	0	true
 TSFACTDATA_2009Q4	IBMDEFAU...	4096	0	true
 TSFACTDATA_2010Q1	IBMDEFAU...	4096	0	true

Figure 9-4 Table space list including FACTDATA table spaces for partitioned table

When creating each table space, click **Storage Options** to assign the table space to a storage group. Click **I/O Settings** to inherit Overhead, Transfer rate, and Data tag values from the storage group (Figure 9-5).

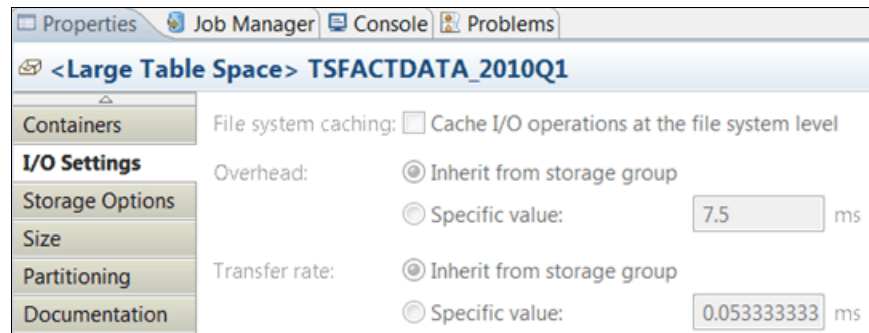


Figure 9-5 Set table spaces for partitioned table to Inherit from storage group

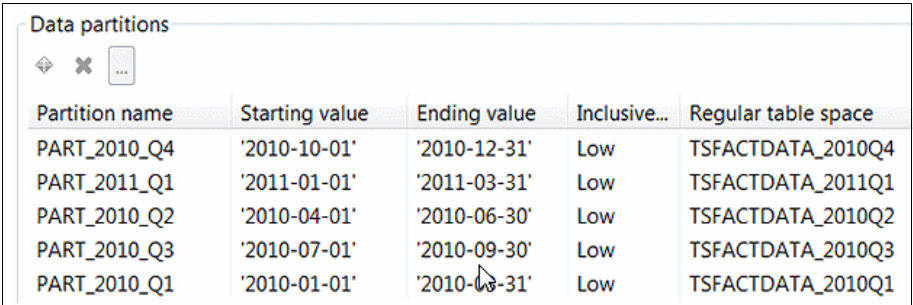
At this point, the storage group and table spaces have been created. The range partitioned table can now be created.

3. Create data partitions for the fact table.

In this example, a partitioned table is created to hold the sales transactions. In our scenario, the table already exists as a regular table and has to be migrated to a range partitioned table. We achieve this by doing the following:

- Generate the DDL for the original table.
- Modify the DDL to incorporate the data ranges you need and table spaces to be used. Use a new table name.
- Create the new table with required indexes.
- Use the INSERT with subselect statement to copy the data from the old to the new table.

Figure 9-6 show the partitioned fact table with each data partitioned assigned to a table space.



Partition name	Starting value	Ending value	Inclusive...	Regular table space
PART_2010_Q4	'2010-10-01'	'2010-12-31'	Low	TSFACTDATA_2010Q4
PART_2011_Q1	'2011-01-01'	'2011-03-31'	Low	TSFACTDATA_2011Q1
PART_2010_Q2	'2010-04-01'	'2010-06-30'	Low	TSFACTDATA_2010Q2
PART_2010_Q3	'2010-07-01'	'2010-09-30'	Low	TSFACTDATA_2010Q3
PART_2010_Q1	'2010-01-01'	'2010-03-31'	Low	TSFACTDATA_2010Q1

Figure 9-6 Partitioned fact table with each data partitioned assigned to a table space

By aligning a data partition with a table space, table space data can be moved asynchronously from one storage group to another as it ages. In addition, table space maintenance operations such as backup can be grouped by storage group so that backup frequency can be determined by data age. This reduces the cost of ownership through less maintenance time and a reduction in the amount of time and consumables required to maintain data that has not changed.

Figure 9-7 shows that Optim Performance Manager can also be used to monitor table space activity and performance.

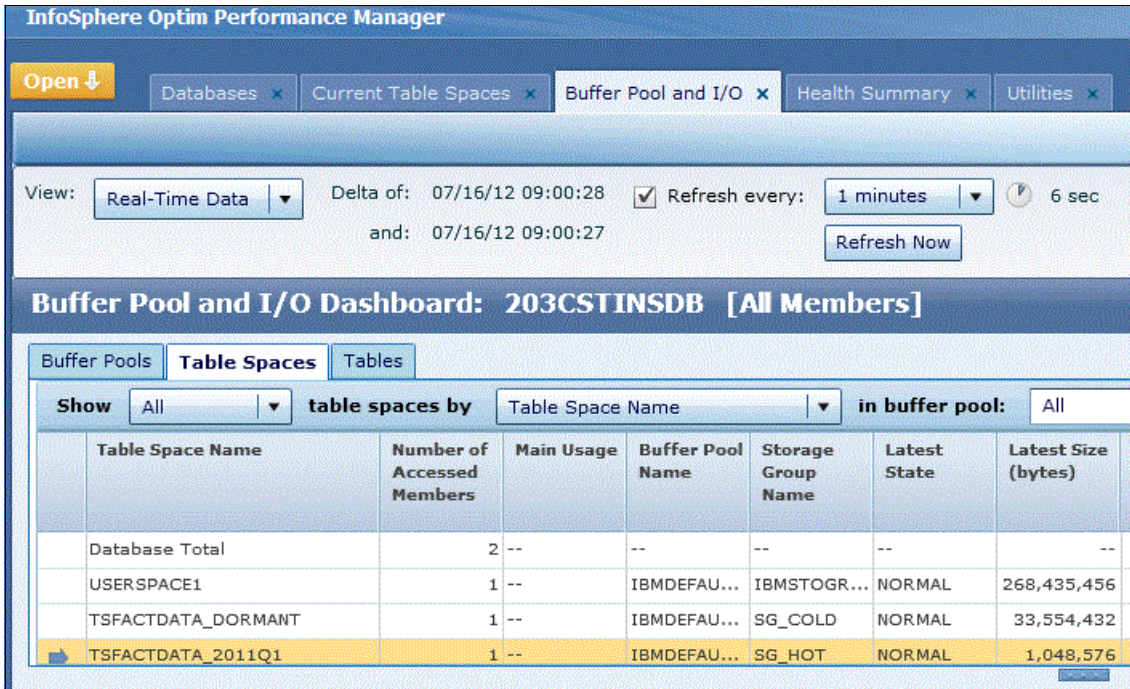


Figure 9-7 Monitoring table space performance metrics

9.3 Aging data

In Chapter 8, “Building a corporate backup and recovery strategy” on page 289, we show how Optim High Performance Unload can be used to meet your service level objectives for data. The act of moving the table space from one storage group to another is managed as an asynchronous task that can be managed by the DBA at command line level. Both IBM Data Studio and Optim Configuration Manager also provide interfaces for performing this task.

For example, you can move data for 2010Q1 from warm to cold storage using the Optim Configuration Manager interface referenced in Chapter 12, “InfoSphere Warehouse resilience with Optim Configuration Manager” on page 413 or from the command line as shown in Example 9-1.

Example 9-1 Rebalancing data by moving from one storage group to another

```
ALTER TABLESPACE tsfactdata_2010q1 USING STOGROUP sg_cold
```

Although the movement of data is an asynchronous task, the data remains available to applications during the process. Because the TSFACTDATA_2010Q1 table space inherits its media attributes from its storage group, its media attributes now are based on storage group SG_COLD.

Monitor the progress and the status of the operation using the MON_GET_REBALANCE_STATUS table function as shown in Example 9-2.

Example 9-2 Monitoring progress of the table space rebalance process

```
SELECT tbspace_name, rebalancer_status, rebalancer_extents_remaining,  
rebalancer_extents_processed FROM TABLE (MON_GET_REBALANCE_STATUS(NULL, -1))
```

The operation can be suspended or resumed where required, as shown in Example 9-3.

Example 9-3 Monitoring progress of the table space rebalance process

```
-- Suspend the rebalance process  
ALTER TABLESPACE tsfactdata_2010q1 REBALANCE SUSPEND  
-- Resume the rebalance process  
ALTER TABLESPACE tsfactdata_2010q1 REBALANCE RESUME
```

The movement of data from one storage group to another can be throttled as a utility using the UTIL_IMPACT_LIM and UTIL_IMPACT_PRIORITY configuration parameters. Throttling database operations using these configuration parameters throttles CPU processor usage; it does not control I/O usage.

Avoid moving too many table spaces at any one time because there is an effect on both the source and target storage group paths.

9.3.1 How active is the data

We have discussed how data ages and how to manage and order data by age. However, it is equally important to understand the temperature of data by how often it is used.

By understanding how active your data is, you can further manage the physical implementation of your data warehouse. Ingesting and maintaining inactive data is a poor value proposition.

The SQL statements dashboard in Optim Performance Manager has two tabs, the Top Individual Executions tab and the Execution summary tab. The

Execution Summary tab provides a summary of data for each unique query issued for the time period selected, including:

- ▶ Number of executions
- ▶ CPU time
- ▶ Rows read
- ▶ FCM traffic metrics

Figure 9-8 shows the Execution Summary tab on the SQL Statements dashboard. The Actions menu is also shown. From the Actions Menu you can switch views to see individual executions of the selected SQL Statement.

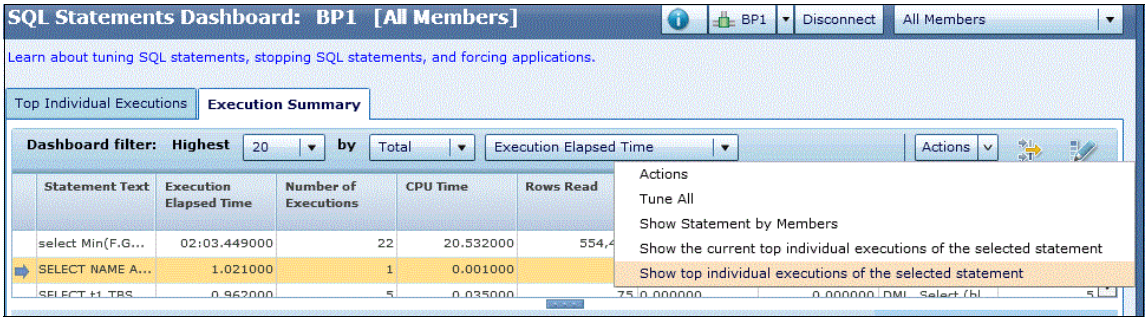


Figure 9-8 Execution Summary tab with Actions menu options

You can add additional columns as needed to aid analysis and filter by columns to analyze grouped data. Data on the grid can be exported to a spreadsheet by right-clicking the data grid and selecting **Copy to clipboard**. On the clipboard, you can perform these tasks:

- ▶ Manipulate and perform further analysis by the FROM and WHERE clause to get active data information by table or date
- ▶ Review total number of executions, CPU, and elapsed execution time by query text

9.4 Optim High Performance Unload to archive or migrate cold data

Chapter 8, “Building a corporate backup and recovery strategy” on page 289 demonstrates how IBM InfoSphere Optim High Performance Unload can be used to meet your service level objectives for data recovery.

Here, we build on the concepts introduced earlier and describe how Optim High Performance Unload can be used to archive data and migrate data from one database to another.

When data is no longer needed in the data warehouse, or where the cost of storage and maintenance has exceeded the value to be gained from the data, it is advisable to archive the data. In archiving data, it is important that you have a clear and simple method for reinstating the data if required.

Maintaining database or table space backup images, tapes and history can be prone to error. IBM Optim High Performance Unload can be used to quickly and effectively unload the data to be archived to a file or files. When completed, the files can be retained and the database object removed, which frees space for warmer data.

9.4.1 Archive cold data

Data archiving is discussed in the context of archiving data out of the data warehouse. The target destination can be a file, backup image, or a federated server if data access to the archived data is required.

The goals when archiving data are to have a minimal effect on the production server, and to be able to realize the storage capacity made available as a result of the archiving process.

By using storage groups in conjunction with table partitioning and table space usage as described earlier, a data partition can be detached without affecting the data availability of the remaining data partitions in the table. After it is detached, the data partition can be dropped along with the associated table space. Data in table spaces can then be asynchronously migrated from other storage groups.

Data can be archived in many ways. The traditional method is to perform and keep a full database or table space backup. An alternate method is to use Optim High Performance Unload to unload data for the individual data partition and store the data in flat files for future reference or for migration to another database.

The following example shows how to use the Optim High Performance Unload utility to unload data from a table space container for a single data partition.

The following steps were completed in our test environment to archive and drop a data partition:

1. Detach the data partition from the partitioned table.

Detaching the data partition from the range partitioned table reverts the data partition to a separate regular table with the new table name you specify.

Example 9-4 shows the statement to detach data partition PART_2009Q1 into table ARCHIVE_PART_2009Q1. The new table remains in the same table space.

Example 9-4 Detach a data partition

```
-- Detach a data partition from a partitioned table
ALTER TABLE CSTINSIGHT.CUSTOMER_TXN DETACH PARTITION part_2009q1 INTO TABLE
archive_part_2009q1
```

2. Create the Optim High Performance Unload control file.

Create the control file required to unload data from the table space container to a single file on disk.

Example 9-5 Sample control file used to unload data

```
-- archive_part_2009q4.ct1
GLOBAL CONNECT TO cstinsdb
DB2 NO LOCK NO QUIESCE NO;
UNLOAD TABLESPACE
SELECT * FROM cstinsight.archive_part_2009q4
;
output("/stage/filename.del")
format del;
```

Where:

- The directory /stage is accessible on all data nodes.
 - QUIESCE NO is sufficient here because the DETACH process is a system catalog operation and not a physical data move.
 - /stage/filename.del represents a single file for data from all database partitions.
3. Execute Optim High Performance Unload from the command line as shown in Example 9-6.

Example 9-6 Issuing db2hpu from the command line with control file

```
db2hpu -i db2inst1 -f archive_part_2009q4.ct1
```

4. Drop the table.

The data partition has been detached from the partitioned table and the content of the table has been archived to file on disk. At this point, the file can be archived and backed up to TSM or another storage manager.

The table can now be dropped using the standard DROP TABLE statement as shown in Example 9-7.

Example 9-7 Drop archived table

```
DROP TABLE cstinsight.archive_part_2009q4
```

5. Drop the data partition table space.

The dedicated table space for the data partition can now be dropped as shown in Example 9-8.

Example 9-8 Drop archived table

```
DROP TABLE SPACE tsfactdata_2009q4
```

9.4.2 Migrate data from production to an alternate database

Optim High Performance Unload uses named pipes and the DB2 LOAD utility to migrate data from one database to another without the need to stage the data to disk. This feature is particularly useful for the following tasks:

- ▶ Offloading data to data marts
- ▶ Generating data subsets for direct population of downstream systems
- ▶ Populating development, test, or other non-production systems
- ▶ Repartitioning data
- ▶ Migrating cold or dormant data to a federated database

The global block and unload block used in the control to unload data to output files or named pipes is introduced in Chapter 8, “Building a corporate backup and recovery strategy” on page 289.

In this chapter, we introduce the *migrate block* which is used to automatically migrate data from one database to another without the need for you to manually create the named pipes or issue the LOAD command.

Migrate block

The migrate block provides the db2hpu utility with the details of the table to be migrated and the source and target environments involved.

In Example 9-9, we migrate data for the data partition PART_2009Q4, part of the table CSTINSIGHT.CUSTOMER_TXN, from a source to the target database environment. The source database has 16 database partitions on two data nodes. The target database in this example has just four database partitions on a single data node, similar to an IBM Smart Analytics System 5710.

Example 9-9 Issuing db2hpu from the command line with control file

```
GLOBAL CONNECT TO <target_dbname>;

--Migrate Block
MIGRATE TABLESPACE
PART(ALL)
DB2 NO LOCK YES QUIESCE YES
TARGET ENVIRONMENT(INSTANCE "target_instance" on "target host" IN
"target_dbname")
WORKING IN("/work1")

SELECT * FROM CSTINSIGHT.CUSTOMER_TXN;
DATAPARTITION_NAME (part_2009q4)
TARGET KEYS(current parts(1:4))
FORMAT MIGRATION INTO schema.customer_txn MODIFIED BY IDENTITYOVERRIDE
;
```

Where:

- ▶ /work1 is available on all data nodes.
- ▶ The target database table exists.
- ▶ The number of source database partitions is 16.
- ▶ The number of target database partitions is 4.
- ▶ MIGRATION INTO specifies the name of the target table name where different to the source table name.
- ▶ IDENTITYOVERRIDE indicates that the contents of the identity column on the source database table is to remain intact during the transfer.

In this example, data is repartitioned from 16 database partitions to four database partitions. Optim High Performance Unload achieves this by partitioning the data on the source system using the default round-robin partition map method. Data is then streamed to four named pipe files on the target system and the LOAD utility is initiated.

The contents of the identity column, often used to generate a surrogate key, are retained and this allows for fact and dimensional data to be transferred with foreign key (informational) constraints intact.

Where the default round-robin partition map is not in use (for example, if the database has been redistributed), you have to use a single file so that data is streamed through the coordinator partition on the target database. This is achieved by using the REPART NO keyword in the TARGET_ENVIRONMENT syntax, so the control file in Example 9-9 on page 351 will look like the statement shown in Example 9-10.

Example 9-10 Drop archived table

```
TARGET ENVIRONMENT(INSTANCE "target_instance" on "target host" IN
"target_dbname" REPART NO)
```

To identify the partition map on the source and target database, check the value for the DB2 registry parameter DB2_PMAP_COMPATIBILITY.

- ▶ The default setting of ON indicates that the distribution map size remains 4096 entries (the pre-version 9.7 behavior).
- ▶ When this variable is set to OFF, the distribution map size for new or upgraded databases is increased to 32768 entries (the version 9.7 behavior).

For example, DB2 9.5 uses 4 K maps. DB2 9.7 and later can use 32 K maps.

Optim High Performance Unload migration and the DB2 10.1 temporal table feature

Optim High Performance Unload supports temporal tables in DB2 10.1. This feature can be used to unload or migrate data based on a system or business time perspective.

Here we demonstrate how to extract a specific set of data from temporal points using the temporal table feature. Data is migrated from a partitioned fact table and from a nonpartitioned dimension table to a target database.

The PRODUCT_DIM table was altered to take advantage of the system time temporal feature based on the syntax in Example 9-11.

Example 9-11 Introducing temporal features to an existing table

```
-- Add System Start column
ALTER TABLE Bi_schema.tb_product_dim ADD COLUMN sys_start TIMESTAMP(12) NOT
NULL GENERATED ALWAYS AS ROW BEGIN;
-- Add System End column
ALTER TABLE bi_schema.tb_product_dim ADD COLUMN sys_end TIMESTAMP(12) NOT NULL
GENERATED ALWAYS AS ROW END;
```

```
-- Add Business Start column
ALTER TABLE bi_schema.tb_product_dim ADD COLUMN ts_id TIMESTAMP(12) NOT NULL
GENERATED ALWAYS AS TRANSACTION START ID;
-- Add Business End column
ALTER TABLE bi_schema.tb_product_dim ADD PERIOD SYSTEM_TIME(sys_start,
sys_end);
-- Create temporal history table
CREATE TABLE bi_schema.tb_product_dim_hist LIKE bi_schema.tb_product_dim;
-- Assign temporal history table
ALTER TABLE bi_schema.tb_product_dim ADD VERSIONING USE HISTORY TABLE
bi_schema.tb_product_dim_hist;
```

The Optim High Performance Unload control file created to perform the task contains two migrate blocks: one for the dimension table, and one for the fact table. The source database is BCUDB and the target database is TEST_BCUDB; see Example 9-12.

Example 9-12 Introducing temporal features to an existing table

```
GLOBAL CONNECT TO BCUIDB;

-- Migrate Block for Dimension table
MIGRATE TABLESPACE
part(0)
DB2 NO LOCK NO QUIESCE NO
TARGET ENVIRONMENT(INSTANCE "db2inst1" on "test_coordinator" IN
TEST_BCUDB)

-- Select statement and target keys
SELECT * from bi_schema.tb_product_dim FOR SYSTEM_TIME BETWEEN '2012-06-18' AND
'2012-06-19' WHERE product_description LIKE '%fzc%'
;
TARGET KEYS(current parts(0))
FORMAT MIGRATION INTO test_bcuidb.tb_product_dim
;

-- Migrate Block for The fact table
MIGRATE TABLESPACE
part(1:8)
-- Buffer pool is flashed and write access to the table prevented during the
Migration
DB2 NO LOCK YES QUIESCE YES
TARGET ENVIRONMENT(INSTANCE "db2inst1" on "test_coordinator" IN
test_bcuidb)

-- Select statement and target keys
SELECT * FROM bi_schema.tb_sales_fact WHERE store_id BETWEEN 101 AND 250
```

```
;
DATAPARTITION ID (2)
TARGET KEYS(current parts(0:16))
FORMAT MIGRATION INTO test.tb_sales_fact
;
```

The MIGRATE block for the dimension table specifies the following items:

- ▶ DB2_NO determines that a connection to the production database will not be made for the SELECT statement.
- ▶ LOCK NO QUIESCE NO determines that data does not need to be flushed from the buffer pool to disk because the data to be migrated is aged.
- ▶ The TARGET ENVIRONMENT instance is db2inst1, the target administration host is test_coordinator, and the target database is TEST_BCUDB.
- ▶ The SELECT statement for the product dimension retrieves data for a system time range.
- ▶ TARGET_KEYS(current parts(0)) indicates that the source table is not a partitioned table and that it resides on the coordinator node.

The MIGRATE block for the fact table specifies the following items:

- ▶ PART(1:8) determines that the target fact table is on database partitions 1 to 8.
- ▶ The SELECT statement for the fact table retrieves data for a subset of stores.
- ▶ LOCK YES QUIESCE YES will flush data from the buffer pool to disk and lock the source database table for the duration.
- ▶ DATA_PARTITION_ID(2) determines that only data from the specified data partition is to be migrated. This aligns both tables in terms of dates.
- ▶ TARGET_KEYS(current parts(1:16)) indicates that the source table is partitioned across 16 database partitions.
- ▶ MIGRATION INTO TEST.TB_SALES_FACT is the target table.



Techniques for data mining in an operational warehouse

In this chapter we provide an overview of data mining techniques and use cases in an operational data warehouse environment. We look at IBM and non-IBM data mining tools and functions for implementing data mining for operational warehousing.

The main focus of the chapter is on the integrated data mining modeling and runtime features in InfoSphere Warehouse 10.1 Advanced Enterprise Edition and Design Studio. We then consider an alternative data mining scenario using SAS Enterprise Miner to provide an integrated and complementary solution.

10.1 Data mining in an operational warehouse environment

Although there are many definitions of data mining, we use the definition provided in the IBM Redbooks publication *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813:

... 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.¹

The goal of data mining is to *discover* “potentially valuable” and “non-trivial” patterns and data relationships that exist within enterprise data assets but are not obvious and are non-trivial to locate and exploit. Because they are non-trivial and difficult to discover, a complex body of *modeling* techniques have been developed over the past decades to use advanced statistical and mathematical operations to search for, discover, and characterize these useful patterns and relationships.

Many questions might be raised from this discussion so far:

- ▶ What are these discovery and modeling processes?
- ▶ How do they work?
- ▶ What kinds of relationships and patterns do they detect?
- ▶ Why are these relationships and patterns valuable?

We seek to answer these questions and more in the following subsections. We also characterize the discussion in the context of operational data warehousing and operational BI because that is the focus of this book. We begin with an overview (or more accurately, a review) of data mining concepts and technology, and then we move into a discussion of the data mining scenarios in an operational warehouse environment. From this, the high value of data mining is clear. In the rest of the chapter, we explore the various techniques that are available for InfoSphere Warehouse 10.1 to implement and deploy a data mining solution.

10.1.1 Data mining overview

The concepts, technologies, and techniques that encompass the subject of data mining is a vast topic that itself requires at least a full-sized book to explain thoroughly. An in-depth study of data mining is useful but beyond the scope of

¹ *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

this particular document. There are several IBM Redbooks that address data mining in an InfoSphere Warehouse environment in detail, including most recently *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

For completeness, we provide a summary of the concepts and processes here.

Types of data mining

There are several data mining techniques, and they can be broadly classified into one of two categories. It is possible for a single technique fall into both categories depending on the role the technique is playing at a given time.

- ▶ Discovery techniques
- ▶ Predictive techniques

We discuss each of these in turn.

Discovery data mining and techniques

Discovery methods are designed to find patterns in the historical data without any prior knowledge of what those patterns might be. Thus, we must *discover* the patterns organically. Three discovery mining methods supported by InfoSphere Warehouse directly are listed here:

- ▶ Clustering

The clustering algorithm groups data records into segments by how similar they are based on attributes “of interest.” For example, we can choose to *profile* our clients by grouping them according to similar purchasing behavior or demographic attributes to, therefore, introduce more narrowly defined targeted marketing to specific customer *segments*. A clustering method can discover non-obvious client groupings based on analysis of these demographic and behavior attributes.

- ▶ Associations

The association method identifies links (or *associations*) among the data records of individual transactions such as a single retail purchase of multiple items, for example, in a grocery store. A form of *link analysis*, the associations method is commonly used for *market basket analysis* which finds what retail items tend to be purchased together. This knowledge enables retailers to tailor their sales and promotions according to understood buyer patterns.

- ▶ Sequences

Another form of *link analysis*, this method finds sequential patterns across multiple transactions, as in a *sequence of customer events* or purchases. Knowledge of sequential client patterns or behavior can allow retailers, for example, to tailor the shopping experience for individual customers, such as

offering targeted coupons at the register that are likely useful for the *next* shopping trip.

Discovery mining techniques are sometimes referred to as “descriptive” mining because the methods describe relationships and patterns in our data that were not detected before. These techniques are also sometimes referred to as “unsupervised learning” because the results are completely unknown in advance.

Predictive data mining and techniques

The predictive methods of data mining are designed and used to make *predictions* of future data values or events based on known past values or events. In other words, based on everything we know about past data values, we can make a reasonable *prediction* of future values. There are three primary techniques supported in InfoSphere Warehouse 10.1 which are as follows:

- ▶ Classification

The classification method *classifies* values for a record into one of a predefined set of buckets or bins. For example, classification can be used to rank bank loan applicants into risk classes of high, medium, or low.

- ▶ Regression

The regression method makes predictions on a continuous scale, typically between 0 and 1, that represent a *probability* of a particular event or outcome. For example, a regression model can predict the likelihood of a client choosing to take their business away (churn) based on one or more negative events.

- ▶ Time series analysis (forecasting)

This method is used to predict future values (or uncaptured values) of a numerical field in a record based on past values and events over time. For example, time series forecasting can be used to predict product inventory levels to optimize shipping and manufacturing.

For more detailed descriptions of the data mining methods introduced here, see *Dynamic Warehousing: Data Mining Made Easy*, SG24-7418.

The data mining process

The process and steps involved in data mining can appear to be complex and mysterious. In fact they are straightforward and methodical. With the right set of tools, they can be accomplished by a variety of business analysts and application developers.

The process is well documented in many places, including *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813. We

summarize the process here for completeness, but using that book and other relevant resources is a useful way to implement a data mining project.

The key steps in the process are as follows:

1. Understand the business problem to be solved or question to be answered.
2. Identify the data mining approach to be utilized to solve the problem.
3. Understand the data.
4. Prepare the data.
5. Build and train the data mining model.
6. Interpret and evaluate the data mining results.
7. Deploy the results.

We consider each of these steps conceptually in turn. In subsequent sections of this chapter, we discuss how these steps can be performed using a variety of tools, both within the InfoSphere Warehouse Advanced Enterprise Edition 10.1 and other IBM and non IBM data mining tools.

Understand the business problem

As with most business intelligence and analytics solutions and applications, with the data mining process you first need to understand the business problem to be solved. The business questions to be answered with data mining must be clearly articulated and analyzed to validate that they are indeed problems to be solved with data mining. Despite initial assumptions, it might be that a given business problem can be addressed with a non-data mining approach, such as with an Online Analytical Processing (OLAP) solution.

Consider, for example, that we want to direct a focused targeted marketing campaign at select segments of our customer base. It is certainly possible using multidimensional analysis models to segment our customer population based on demographic dimensional attributes such as age, income bracket, gender, geography, average purchasing history, and so on. All of these attributes are readily available (typically) and do not require any specialized modeling. However, these simple attributes might not differentiate our customer base as much as is needed to make the campaign meet the target success rates. A well-designed clustering model using data mining can be used, alternatively, to generate a highly differentiated set of customer segments. We can then direct our marketing campaign at the subset of segments that most closely match the target behavior of the campaign, thus improving the chance of success.

Fully articulating and understanding the implications of the business problem are fundamental to the critical first step in the data mining solution process.

Identify the data mining approach

Now that we understand the business problem and business questions to be answered, and now that we have also determined the problems are best

addressed with data mining, we must now identify and select the best approach. We must select from the discovery and predictive methods described in “Types of data mining” on page 357.

Be aware that there are cases where more than one method is necessary to achieve the desired results. For example, consider that we might use a discovery clustering method to identify high-value customer clusters and then apply a classification method on the high-value customers to determine their likelihood of overrunning their credit card limits in terms of high, medium, or low risk.

It is also possible that the choice of data mining method might be influenced or even determined by the characteristics of the available data. For example, if the available data is at the grain of individual transactions and a clustering method is required to segment the customers, then the data must first be aggregated up to the level of one aggregated transaction per customer. This aggregation adds complexity and cost, and if this is prohibitively high then a different method such as associations or sequences that do not require transaction summarization might be used instead.

This notion of data that influences the data mining approach illuminates that understanding the data and selecting the method go hand-in-hand.

Understand the data

Along with the previous step of selecting the data mining approach, there must also be an analysis of the available data before we can proceed to the practical steps of preparing the data and building the model. This step in the process is similar to the same step that occurs in virtually every business intelligence or reporting activity development project that uses corporate or enterprise data assets.

Available data sources and their data freshness, data formats, and an understanding of the data content must all be investigated and evaluated against the business requirements to ensure that the data necessary to address the problem is available.

For many data mining solutions, the more data that is available to train and tune the models, the better and more accurate the outcomes. This makes the data warehouse an ideal data source for much data mining application activity, especially model development and tuning. There is frequently a long (or long enough) history of data records to facilitate the development of effective data mining models. Data mining has varied data needs that lend it to both the historical analytical warehouse and the more transactional operational warehouse.

There are generally two types of data used in data mining:

- ▶ Granular, transactional data

Also known as operational data, the transactional data records are generated with every transaction initiated by or for the individual. The records are generally composed of a time stamp when the transaction occurred, a target (customer) identifier and an item identifier, where the item refers to a particular product or event or other attribute of interest to the individual transaction to which it is attached. This granular transactional data is most suited to link analyses such as associations or sequences that require multiple records per entity. We typically find data of this form in the operational systems and the operational data warehouse.

- ▶ Behavioral and demographic data

This type of data tends to be at a higher level of summarization than the transactional data, aggregated to the “individual” target level, for example, the level of a single customer. This type of data is most suited for analyses for which the target individual is the “entity of interest” and there is one record per entity with many descriptive attributes per record. This type of data is most frequently consistent with the grain of the analytical data warehouse (or in some cases the operational warehouse that depends on data model employed).

Analysis of the data sources requires that we ensure the following tasks are performed:

- ▶ We have the data elements required by the data mining approach that we selected “Identify the data mining approach” on page 359.
- ▶ If the required data elements are not available, determine whether they can be obtained from an external source.
- ▶ If the required data is not in our warehouse and cannot be obtained externally, then the data mining approach must be revisited and perhaps an alternative method selected.

Completing the analysis of the available data sources then leads to the next step of preparing the data.

Prepare the data for data mining

Data preparation is one of the most important and frequently the most time-consuming parts of the data mining process. If data preparation is inadequate or incomplete, the results of the data mining exercise might be of little or no value. Virtually all of the data mining methods that we discuss in “Types of data mining” on page 357 are built on sophisticated statistical and deep mathematical procedures (and even artificial intelligence techniques), and these

methods require the input parameters to be in a particular format and ranges of values.

The data mining data preparation stage is the process of preparing the input parameters to those mining methods. This process includes understanding the required database fields (tables and columns), how the data in the database management system (DBMS) joins across multiple tables and sources, dealing with missing values, data taxonomies, data integrity and outliers, and data granularity mentioned before. All of these are aspects of the data transformation and preparation.

We discuss methods of data preparation in the sections that follow, including specific tool available in InfoSphere Warehouse Advanced Enterprise Edition 10.1 and other IBM and non-IBM data mining tools.

Build the data mining model

After the data is prepared, we can begin the process of designing and building the data model, training it with sample data, and then testing it with additional sample data. This is a interactive and iterative process that can require many cycles to tune the model to point that it is ready for production deployment.

Training and testing (validating) are required for any of the predictive mining techniques. In this process, a portion of historical data, including data that indicates the actual outcome of interest (for example, the customer did or did not close an account, did or did not act on an advertising campaign, did or did not default on a loan, and so on) is required to build and train the initial model. The model is then tested using a test data set that is statistically equivalent to, but mutually exclusive of, the training data set.

The purpose of testing a predictive model in this way its to assess its accuracy when applied to new data not used in building the model, to avoid an overstatement of the model's accuracy due to overfitting to the training set. The ability of the model to predict the known outcomes in the validation data set is the goal and can indicate a high-quality model.

It is also important to avoid “overfitting” of the model; that is, tuning it to the point that it only works perfectly on the training set but fails in practice when it is confronted by new data values or combinations that are seen in the training or validation sets. A good data mining modeler with a good set of tools can avoid this situation.

When completed, the build process produces a data mining model that can be stored in a common industry standard markup language format called Predictive Model Markup Language (PMML). This is an industry-standard XML-based markup language for describing data mining models so that they can be readily shared and deployed by any PMML-compatible applications or systems.

Interpret and evaluate the data mining results

Related to and part of the model design and training stage described previously, the results interpretation and evaluation stage typically uses visualization techniques to illustrate the results of running the mining model. There are unique visualization views tailored for each of the data mining methods that present information about the quality of the model and model results such as the individual customer clusters with descriptions. The visualization facilitates the modeler in the task of determining the quality of the model, and aids in identifying parameters that have to be tuned and manipulated to improve the quality of the model. The visualizations for each type of model are specifically designed to help the analyst interpret the mining results in the context of the business problem to be solved or question to be answered. We show examples of these visualizations in later sections.

Deploy the results

The final step involves deploying the data mining model and results into the business intelligence or reporting solution for the department or enterprise, to provide a solution to the business problems described in “Understand the business problem” on page 359. The deployment can take different forms depending on the type of mining method employed and the business requirements that are being addressed. Typical deployments include:

- ▶ **Ad hoc insight**
Data mining can be used in an ad hoc fashion to address a specific but non-recurring question or perhaps for exploration from an individual, for example, in a sandbox environment.
- ▶ **Interactive insight**
This approach involves embedding or integrating the data mining model into a business process through integration with BI tools and applications. For example, business analytics on the marketing team might start a new promotion campaign by running the customer segmentation (clustering) application, selecting one or more target segments, then running market basket analysis on the members of those segments to tailor the advertising campaign according to buying behavior. In this scenario, the data mining application is an interactive tool integrated into the business process for designing and launching the campaign.
- ▶ **Scoring, both real time and batch**
This approach involves applying a data mining model to every record in the database (either in batch or real time) and updating the record with the “value” generated by the model. The predicted value varies by model, for example:
 - For a clustering model, the “score” is the best-fit cluster for each individual.

- For association, the score is the highest-affinity item, given other items.
- For a sequence model, the score is the most likely action to occur next.
- For a predictive model, the score is the predicted value, event, or response.

Therefore, the score is used to augment each data record with new or additional information. This information can be represented in business analytics models such as dimensional models as dimensional attributes, or they can even be represented as a new dimension, depending on the analytical and reporting requirements.

As mentioned already, the scoring process can occur in batch or in real time. Again, this depends entirely on the business requirements and the nature of the data ingest into the data warehouse.

- Batch scoring

With the batch scoring approach, the scoring function is called periodically to run over all or some portion of the data records in the data warehouse. In most warehouse scenarios, this is programmed into the later stages of an extract, transform and load (ETL) process after new data is loaded into the system.

- Real-time scoring

With the real-time scoring approach, the scoring function might be started each time a new record is introduced into the data warehouse. Alternatively, the function can be started on demand during a business process. For example, when a new client requests a line of credit using an online application, this might trigger the invocation of the scoring function to determine a credit worthiness ranking dynamically.

In subsequent sections of this chapter, we consider how this data mining process can be implemented using the features of InfoSphere Warehouse Advanced Enterprise Edition 10.1. We also consider other IBM and non-IBM data mining solutions that can be deployed on top of InfoSphere Warehouse.

10.1.2 Data mining scenarios in an operational data warehousing environment

Now that we have refreshed the concept of what data mining is and how it is implemented, we can return to the discussion of specific scenarios we see for data mining in the operational data warehousing environment. We discuss the challenges and opportunities that are exposed for data mining in this environment. The following sections discuss various examples of how data mining can be used in operation warehousing scenarios. These examples are

not intended to be inclusive of all possible scenarios. Rather, they are intended to give readers a sampling of what is possible.

Online retailing and e-commerce

Two key objectives for a successful online shopping website or application are explained here:

- ▶ To provide a positive and successful shopping experience for customers, such as by making it as easy as possible for customers to find the items they want, make their purchase, and exit the site quickly. Customers lose interest and might abandon the site if they cannot find what they want, items are out-of-stock, navigation of the site is tedious (too many clicks are required), or they are confronted by too much advertising that does not pertain to their objectives.
- ▶ To maximize the revenue generated by each customer visit by not only presenting the items specifically sought, but also by presenting or suggesting additional items that are most likely to appeal to the customer and potentially increase the value of the sale.

Clearly, these two objectives have the potential to work against each other. If the retailer confronts the customer with too many ads or additional product options, then the customer might become annoyed and abandon the purchase. At the same time, if the retailer does not attempt to market additional products to the customer that might in fact appeal to them, then the customer might happily make their planned purchase, but the retailer will have missed an opportunity to generate additional revenue. The happy intersection of these two goals is the point where the customer easily finds the items they intend to purchase and are also presented with additional or alternative items for which they might be unaware and are pleased to purchase as well. In the best case scenario, both retailer and customer get more of what they want.

Data mining and some of the techniques described in previous sections can play an important role in making this scenario effective. Both discovery and predictive techniques can be applied to achieve the desired results.

- ▶ The *associations* method (discovery) can be used to perform *market basket analysis* to discover what products are frequently purchased together. A comparison to the current customer's shopping basket can reveal additional items that can be suggested to the customer for their consideration. For example, having selected to order a pair of hiking boots, the online system might also recommend a mobile GPS device for use on a hiking trip.
- ▶ The *clustering* method (discovery) can be used simultaneously with the associations model to provide more targeted recommendations based on buying patterns of customers in the same segment as our current customer.

Use of segmentation can greatly increase the effectiveness of the recommendations made to the customer.

- ▶ The *sequences* link analysis method (discovery) can also be used to identify buying patterns for the current customer over time. This information can be used, for example, to present the customer with coupons or other promotions at the time of checkout that can be used in the next purchase.
- ▶ The *regression* (predictive) method can be used in multiple ways in the online retailer scenario. For example, it can be used to make predictions (based on past behavior of this customer and other customers in the same segment determined by clustering) of how much this customer will spend in a year. This allows the retailer to categorize customers by value and offer special incentives or offers to “high value” customers.

Another example is using a regression model to predict the likelihood of a customer to abandon their shopping cart without making a purchase if two or more items that they want are out of stock. Foreseeing this outcome, the retailer can proactively offer incentives such as free shipping or discounts on purchasing a similar but more expensive alternative. Such offers might appeal to the customer to not only make a purchase in the current session, but also return for future shopping.

As we can see, there are many opportunities to use data mining in an operational environment (at the point of sale) in the retail market.

Real-time fraud detection

This scenario can apply across any industry segment, including financial services, health care, insurance, retail, manufacturing, government services, and so on. Any business process that can be manipulated intentionally for benefiting someone fraudulently provides an opportunity for data mining to assist with prevention.

Predictive methods are best suited for this type of problem, especially *classification* and *regression* methods. Consider, for example, detection of credit card fraud. Much credit card fraud is detected long after the fraud was perpetrated, so identifying and prosecuting the guilty parties is much more difficult. The problem is costing the credit card industry billions of dollars annually, \$7.6 billion in 2010², for example. If potential fraud can be detected at the point of sale *before* the transaction completes, then more fraud can be prevented and perpetrators can be identified.

² Kerber, Ross, “US Banks Losing Ground in Combating Credit, Debit Card Fraud: Report”, Huffington Post Business, http://www.huffingtonpost.com/2011/10/04/credit-debit-card-fraud-more-common-banks-lose-ground-hackers_n_994690.html, August 9, 2012.

The process for using predictive data mining to attack this problem is basically two steps:

1. The predictive model is *trained* (developed) using a large set of historical credit card transactions (ideally going back many years) that contains a statistically valid set of fraudulent and non-fraudulent examples. Include all variables about the transaction that are available at the time of the transaction in the data set, such as information about the buyer, recent account charge history, account limit, location of the transaction, amount of the transaction, and past purchasing patterns.

The model can be designed and tuned against the training data set to identify the important variables that predict fraud. A classification method then assigns a rank or risk value to each transaction based on the known variables, such as high, medium, or low risk of fraud. Next, the trained model of each method is validated against a second set of historical data with known outcomes, and the predictions are compared against the outcomes to determine the effectiveness of the model to detect fraud.

2. The validated model is then *deployed* in the operational environment. The model is started at the moment of the transaction (before it completes) to produce a “score” (runtime scoring) of either the classification “bucket” for this transaction (high, medium, or low risk), or a probability score (using regression) for the likelihood of fraud.

The credit card company establishes policy guidelines for an acceptable fraud risk threshold. If the generated score is above a set threshold, then there is a high possibility of fraud (though it might not be fraudulent at all). The transaction might either be stopped entirely or the clerk or agent at the point of sale (cash register or airline ticket counter, and so on) might be instructed to take certain actions to validate the card before proceeding.

This technique can, and does, detect and prevent much credit card fraud. It can be used, for example, to detect when a seldom-used card is suddenly used in a number of purchases over a short period of time, taking into account seasonal factors such as gift-giving holidays. It can detect when buying patterns vary widely, such as sudden frequent use for online purchases, or purchases made across great geographic distances over a short period of time. Credit card fraud continues to be a major problem, as the cited news article illustrates, so there continues to be significant opportunity for data mining to continue playing a role in detection and prevention.

Real-time risk scoring

A final common example that is seen frequently in operational warehouse environments involves online self-service auto loan or mortgage applications. These systems allow an applicant to fill out one or more forms online, collecting information about demographics, employment and salary, education, marital and

parental status, and so on. The system can then use predictive models (*classification*) to generate a credit risk score in real time that can be used as a basis for either denying the application outright, or recommending for approval. The scoring might also be used to indicate a best fit interest rate or repayment terms, depending on risk level. The process might be more nuanced in that the system makes a recommendation that is then followed up by a loan agent. The process might also be used for over-the-phone applications or account changes.

The process for designing, training, validating, and deploying the predictive data mining model in this scenario is the same as in the case of credit card fraud previously described. The scoring method must be integrated with or otherwise invocable from the online application or help-desk application, wherever the business process is being driven, either by the consumer or company representative. We discuss how these functions can be deployed for application integration in subsequent sections in this chapter.

10.2 InfoSphere Warehouse Advanced Enterprise Edition 10.1 tools and features for data mining

InfoSphere Warehouse Advanced Enterprise Edition 10.1 provides all of the tools and features required to develop, train, validate, visualize, deploy, and integrate data mining applications into an operational warehouse business intelligence solution. Through the rest of this section, we review the InfoSphere Warehouse features available to perform

- ▶ Source data exploration
- ▶ Data preparation
- ▶ Modeling in Design Studio
- ▶ Model validation and execution
- ▶ Visualization
- ▶ Scoring with other models through PMML

The data mining features of InfoSphere Warehouse is thoroughly described in other IBM Redbooks publications, most notably *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813. This section draws heavily from that document at a summary level. For more detailed explanations and examples, refer directly to *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*.

10.2.1 Source data exploration in InfoSphere Warehouse 10.1

After the business problem is identified and understood, and the data mining method is selected to solve the problem, the next step in the process is source

data exploration. It is critical to understand where the input data is coming from, how it is structured, and its granularity and relationship to other source data. InfoSphere Warehouse 10.1 Design Studio has powerful features to explore and profile the source data before modeling begins.

Database enablement

Before anything related to data mining can take place, the DB2 database must be enabled for data mining. This is done by right-clicking the database name in the Design Studio Data Source Explorer and selecting **Enable Database for Data Mining**, as shown in Figure 10-1.

After DB2 is enabled, all data mining and data exploration function can be used.

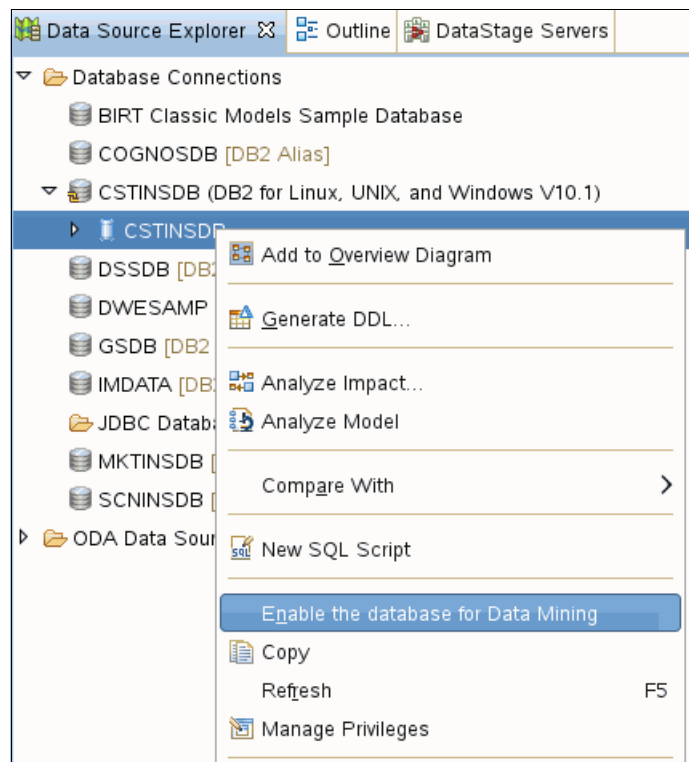


Figure 10-1 Enabling DB2 for data mining

Source data exploration

InfoSphere Warehouse 10.1 Design Studio offers several functions to aid the data mining modeling in the process of exploring the source data under consideration.

Browsing and sampling the data

In the Data Source Explorer, you can right-click a table and select to return all rows of the table (with all columns) or to a sample set of the rows (with all columns); see Figure 10-2.

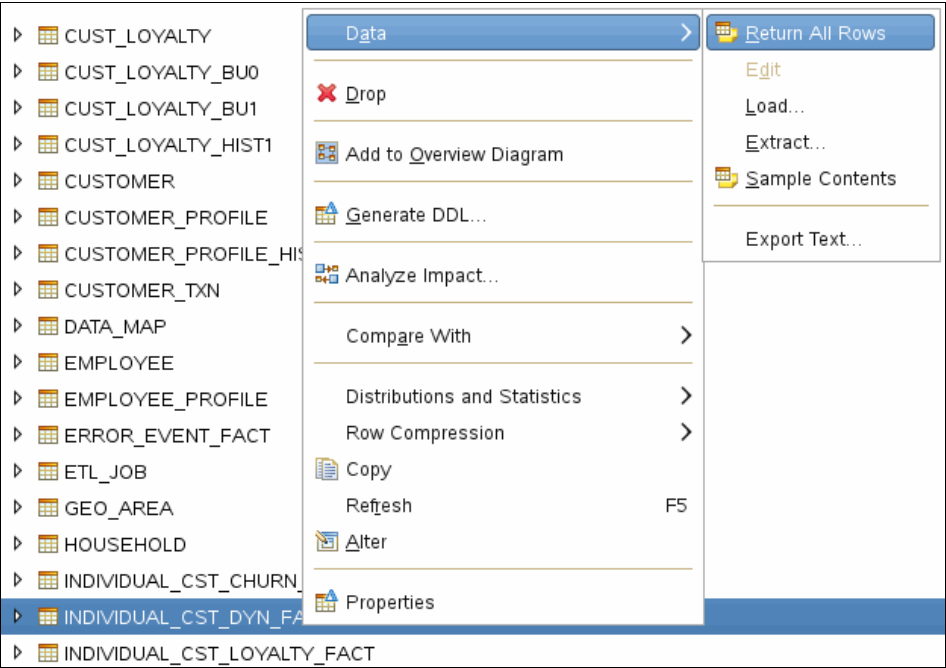


Figure 10-2 Simple data exploration techniques in Design Studio

Figure 10-3 shows a sample result set.

Status	Result1								
	CDR_DT_ID	SRO_ID	IDV_CST_PRFL_ID	CST_RSDNC_AREA_ID	NUM_OF_CST	NUM_OF_NEW_CST	NUM_OF_LOST_CST	NUM_OF_DRMT_CST	
1	20100722	1	11	5	312	4	0	0	
2	20100722	4	11	5	310	0	0	0	
3	20090522	1	11	5	272	4	0	0	
4	20100222	4	11	5	310	0	0	0	
5	20100222	1	11	5	302	4	0	0	
6	20101214	1	11	5	276	0	0	0	
7	20101214	4	11	5	310	0	0	0	
8	20101221	4	11	5	320	0	0	0	
9	20101221	1	11	5	310	3	6	0	

Figure 10-3 Sample data results in Design Studio

Multi-table exploration

You can use general purpose SQL queries to browse and explore data in the source tables using Design Studio. Right-click the database and select **New SQL Script**, as shown in Figure 10-4.

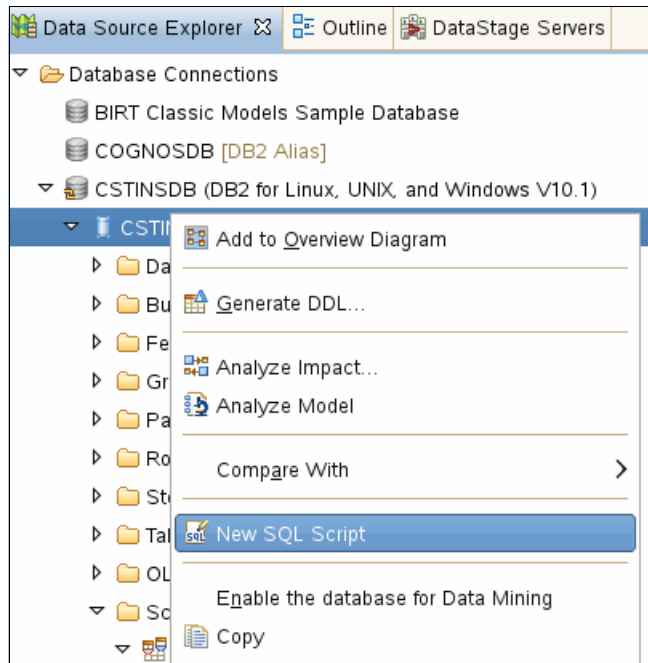


Figure 10-4 Using general-purpose SQL in Design Studio

This operation opens a script editor in which one or more semi-colon-delimited (;) SQL statements can be entered. The results are viewed in the SQL Results tab just like the results shown in Figure 10-3 on page 370.

Record value distribution analysis techniques

InfoSphere Warehouse 10.1 Design Studio also contains more sophisticated data exploration techniques that use statistical analysis functions. The analysis functions investigate value distributions on a column-by-column basis, and illuminate relationships between values of columns. The following analyses are available:

- **Univariate analysis**

Univariate analysis shows both graphical and detailed value distribution statistics for every column in the table. You can drill into the details for any given column for information regarding the loyalty program participation for an individual customer (from the Customer Insight example). See Figure 10-5 on

page 372. Details are also provided regarding the average, mean, minimum value, maximum value, and standard deviation for the values of each column.

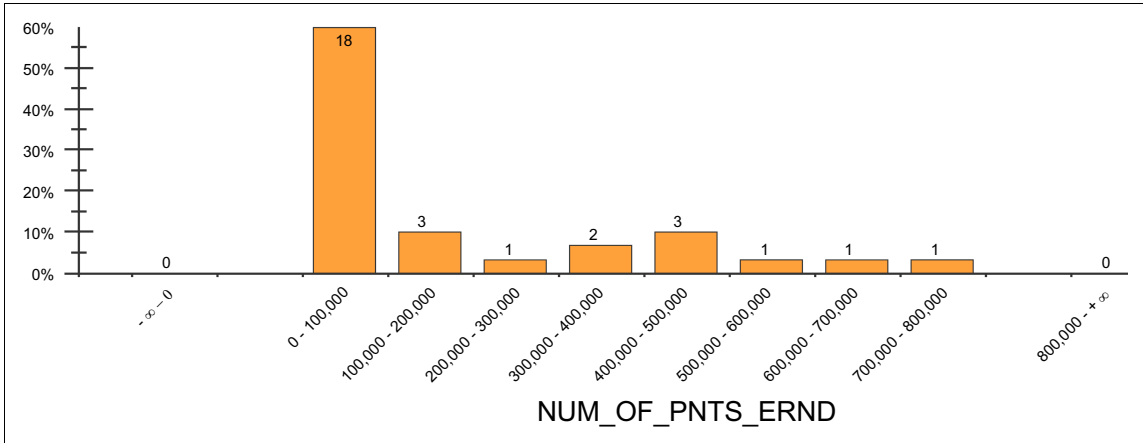


Figure 10-5 Univariate analysis - graphic example

► Bivariate analysis

Bivariate analysis shows the graphical and detailed value distributions of a single table column *relative* to values in the other column, for each of the other columns. Figure 10-6 shows an example of the output.

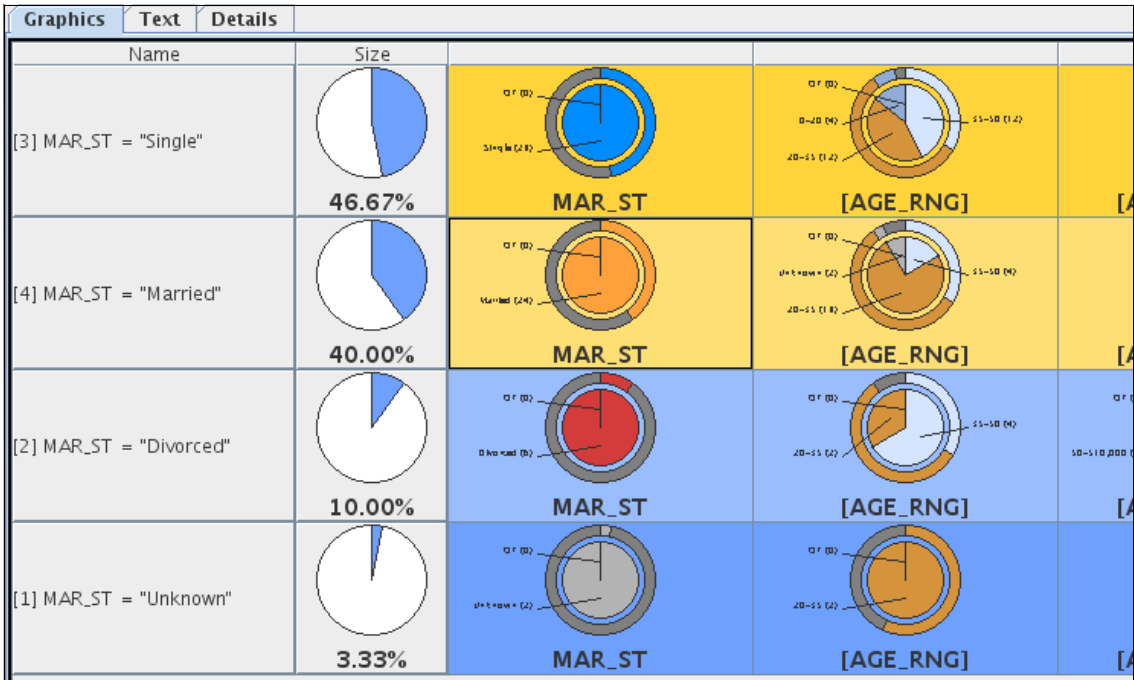


Figure 10-6 Bivariate analysis - graphic example

Notice that there is a row for each MARITAL_STATUS value, then it shows value distributions in other columns, such as AGE_RNG (age range). What we are seeing is the result of Clustering analysis on the marital status column. The data mining method for clustering in InfoSphere Warehouse is used to analyze how the marital status values fit into clusters of other column values.³

► Multivariate analysis

Multivariate analysis shows graphical and detailed distribution of column values and how they relate to each other for up to four columns at a time. One

³ For a detailed description of bivariate analysis, see *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

view simply shows the distribution of values across each of the four columns, as shown in Figure 10-7.

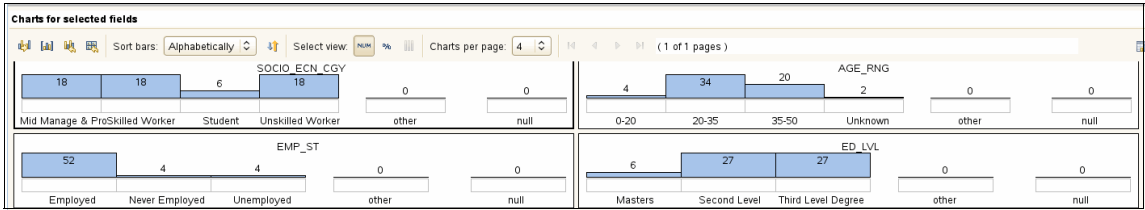


Figure 10-7 Multivariate analysis - default view

Although this information is in itself useful for data exploration, there is an added feature to explore more deeply the relationship between column values. By selecting a single column value of any of the four columns shown, the value distribution for the other columns relative to that fixed value is shown. This can be seen in here in Figure 10-8.

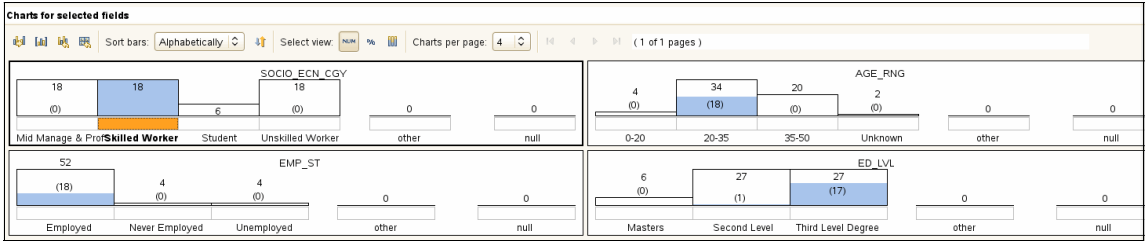


Figure 10-8 Multivariate analysis - one column fixed

In this example, we select the value of **Skilled Worker** in the column **SOCIO_ECN_CGY** (Socio-economic category). The value distributions of the other columns are shown here, limited to only one value for each column in this example, but illustrative nonetheless. The value of **EMP_ST** (employee status) is “Employed”, the **AGE_RNG** (age range) is 20-35 and **ED_LVL** (education level) is “Third Level Degree”.

In a similar manner, a value of the second column value can also be fixed, showing the value distributions of the other columns relative to the two fixed columns. As with bivariate analysis, multivariate analysis is described in detail in *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

Each of these value distribution analysis techniques can be accessed from the Data source explorer in Design Studio. Right-click the table (or even a column)

name and select the **Distribution and Statistics** menu option, then select one of the three analysis techniques described. Figure 10-9 shows the menu.

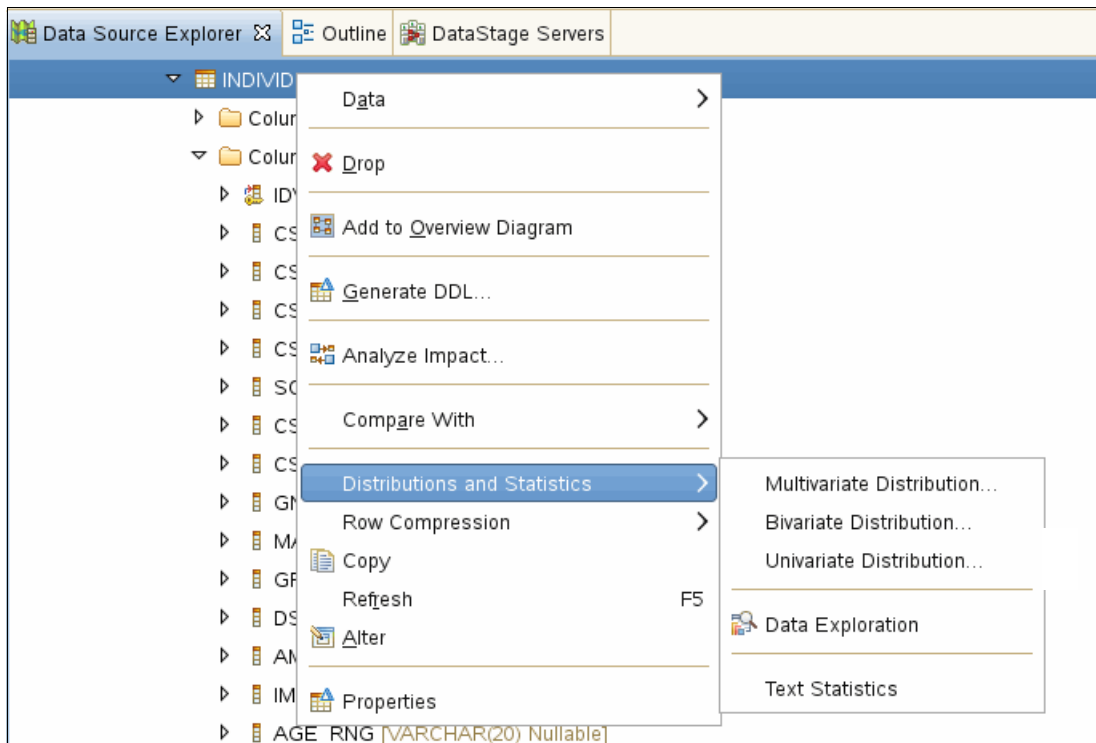


Figure 10-9 Accessing the value distribution analysis techniques in Design Studio

Finer grain exploration options

All of the distribution analysis techniques can operate on all values of all columns in the table, but there are features to affect the grain of the analysis. For example, we can choose to select only a sampling of rows or a subset of columns. SQL WHERE clause filters can optionally be applied to further subset the data under analysis. These techniques can be used to reduce the data value to make the analysis more efficient and effective.

Also, there is a Data Exploration feature under the Distribution and Statistics menu item that combines data browsing, filtering, column statistics, and multivariate analysis under a single tabbed view for convenience.

10.2.2 Data preparation

It has long been true that one of the most time-consuming stages in a data mining project is data preparation. This step involves several activities to make the source data suitable for data mining processing, including but not limited to the following tasks:

- ▶ Integrate or consolidate data from multiple sources into a single data set suitable for data mining
- ▶ Transfer data values or calculate new data values for inclusion in the data mining solution
- ▶ Align granularity (for example, transaction level versus daily summary) of data from different sources
- ▶ Eliminate or correct “bad” data values in the source data, such as null values or other errors

The result of data preparation is a data set containing all of the records required to implement a data mining model using one of the mining methods that we discuss “The data mining process” on page 358.

Within InfoSphere Warehouse 10.1 Design Studio there are two primary steps involved in data preparation, namely creating the input model and defining the data preparation profile, as explained here:

1. Input model creation

The input model defines relationships within the data used for the data mining model in terms of hierarchies and levels. This is similar to dimensional structures in an OLAP model, and in fact OLAP models can be used to guide the development of the input model.

2. Define the data preparation profile

The data preparation profile defines the focus of analysis (what aspect of the data is being analyzed, such as clients in a clustering model). It then defines the relevant properties or variables that are related to the focus of analysis. These properties might be drawn directly from table columns or calculated or transformed from one or more columns.

In summary, the data preparation stage of the data mining process is an ETL process designed to prepare source data into a single data set ready for use as input to the data mining method. As such, traditional ETL means might be used to perform these steps. The SQL Warehousing Tool (SQW) data flow features found in Design Studio can be used for this purpose. Alternatively, Version 9.7 introduced new wizards in Design Studio to aid the development of input models and data preparation profiles.

The input models and data preparation profiles can be seen in the InfoSphere Warehouse 10.1 Design Studio data project explorer folders as shown in Figure 10-10.

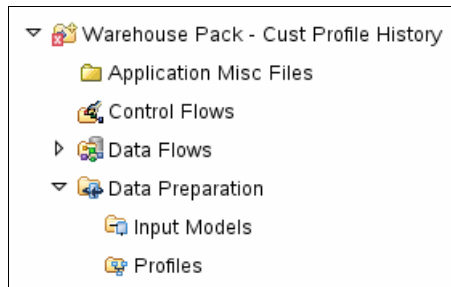


Figure 10-10 Data mining data preparation folders in design studio

To create the input model, right-click the **Input Models** folder and select **New** → **Data Preparation Input Model**. The wizard asks for a model name and then give a choice between the following two options:

- ▶ Selecting an input model based on an OLAP Cubing Services cube model
This option allows the data mining model developer to use a predefined dimensional model of levels and hierarchies as defined in a cube model. This can be a significant time saver and provide synergy between the OLAP and data mining analysis. The admissions, hierarchies, and levels are all derived from the OLAP metadata. In addition, the tables from which the dimensions are defined are selected, and the join relationships between the tables. This greatly simplifies the development of the input model.
- ▶ Selecting a database on which the input model will be based
This option allows you to develop the input model from scratch. Tables can be drawn from the warehouse model and the join relationships can be defined manually. Dimensions, hierarchies, and levels can also be defined manually. The result using this path is the same as when the OLAP model is leveraged.

The result of creating the input model is to define a set of source tables and hierarchies for the input data. See the example in Figure 10-11.

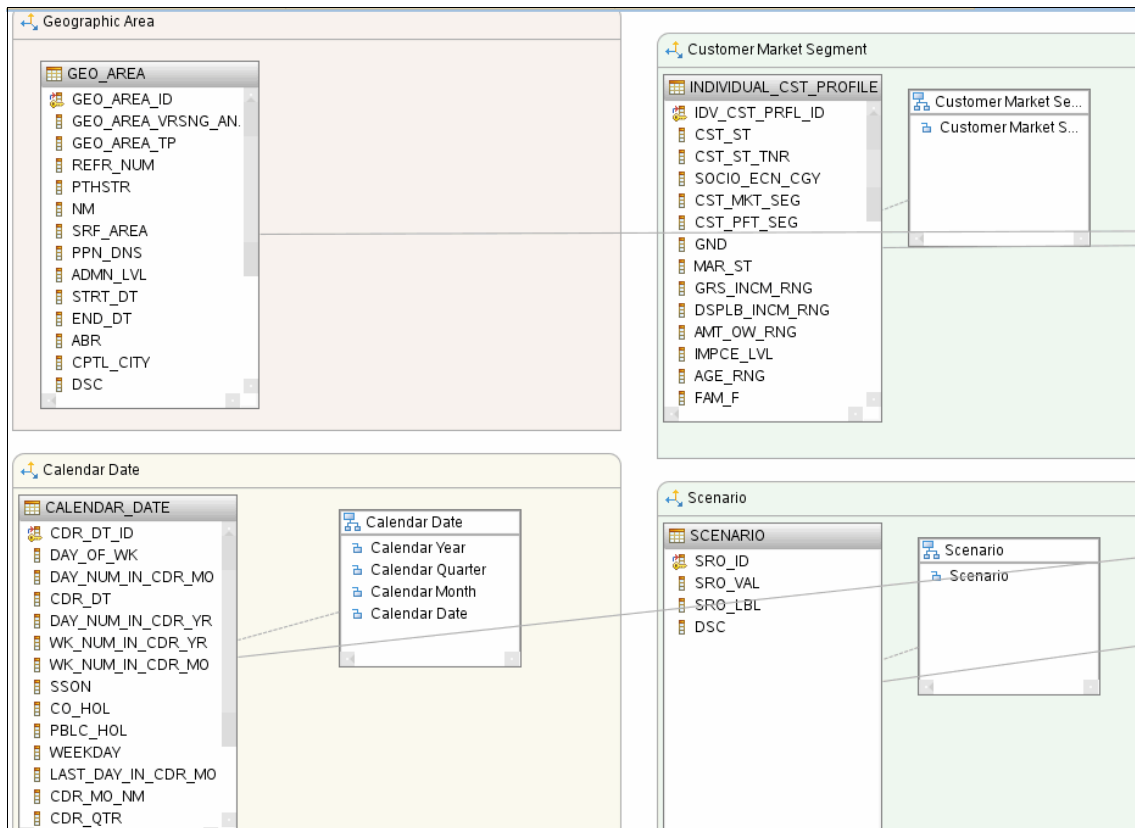


Figure 10-11 Data mining input model from OLAP cube with hierarchies

This model was created using an input OLAP cube model. We can see the different tables for each dimension and the hierarchy levels. Note the Calendar Date dimension and these levels:

- ▶ Calendar Year
- ▶ Calendar Quarter
- ▶ Calendar Month
- ▶ Calendar Date

After the input model is defined, the data preparation profile can be created and specified. Recall that the purpose of the data preparation profile is to define the transformations and calculations necessary to prepare the data set for data mining. To create the profile, select the **Data Preparation** → **Profiles** folder, right-click and select **New** → **Data Preparation Profile**. Give the profile a name

and link it to an input model (like the one created here). Figure 10-12 displays a sampling of what is shown in the data preparation profile panel.

Data Preparation Profile

Overview | **Transformations** | Result Table

Focus of analysis: ⓘ

Icons: [Add] [Edit] [Delete] [Cancel] [OK]

Name	Definition Based On
Customer	"INDIVIDUAL_CST_DYN_FACT"."IDV_CST_PRFL_ID"

Features: ⓘ

Icons: [Add] [Edit] [Delete] [Cancel] [OK]

Filter: <Select col...> Type filter text [X] Sort: <Select col...> [Up] [Down]

Name	Type	Data Type
Income Amount	Focus Attribute	DECIMAL

Properties ⓘ | SQL Results | Problems | Execution Status | Job Status

«Focus Attribute» Income Amount

General

Definition

Focus component: Customer

Result Column

Focus attribute definition based on: Single Column or Attribute

Available columns and attributes: [Filter] [Sort] [Add] [Remove] [OK] [Cancel]

Figure 10-12 Data preparation profile example in Design Studio

The profile consists of essentially two primary elements:

- ▶ Focus of Analysis is the column or columns that are the focus of the data mining analysis; for example, the customers for whom we are doing segmentation.
- ▶ Features or Focus Attributes are the variables, defined in terms of input data columns, that influence the data mining method output. The features are defined in terms of a **Name** → **Definition** → **Output Column** triplet.
 - The name is usually the name of the feature that has meaning to the developer, such as a business name.
 - The definition is in terms of a database column or columns and various functions (transformations) that can be applied to those columns in terms of calculations, aggregations, discretization⁴, and so on.

⁴ Discretization is the process of distilling a continuous numeric value (for example, income) into a series of discrete buckets, such as high, medium, and low.

- The output column designates the column name in the output table into which the results of the transformation of this feature is written. The output table is the input into the data mining model.

10.2.3 Data mining modeling in InfoSphere Warehouse Design Studio

After the source data has been prepared, the data mining model can be developed and tested. Recall that it is best to develop the mining model using a subset of the historical data available, saving an equal portion of the historical data (with known outcomes) for validation. InfoSphere Warehouse 10.1 Design Studio has two primary routes to develop the data mining model:

- The first route is to use the *mining flow editor* in Design Studio to manually build the flow of information from the source data tables into the data mining method operators and finally into an output target. Outputs can be tables or columns, or they can also be visualization methods. Mining flows are conceptually and practically similar to the SQL Warehousing (SQW) data flows described in Chapter 4, “Data modeling: End to end” on page 81.
- The second route is to use the solution plan wizards that have been defined for a subset of common data mining solutions. The wizard prompts the developer for information that will be used to produce the same mining flows as though they were developed manually, but the interface is more business analyst-oriented, seeking input in terms of the business analysis being performed instead of in terms of tables and columns.

For a more thorough discussion of the data mining modeling process and tooling in InfoSphere Warehouse Design Studio, see *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813. We consider each of these approaches at summary level in this document.

Data mining flow editor

InfoSphere Warehouse 10.1 Design Studio provides the data mining flow editor to design and test data mining models in a manner that is similar to SQL Warehousing data flows. Figure 10-13 shows an example of a data mining flow.

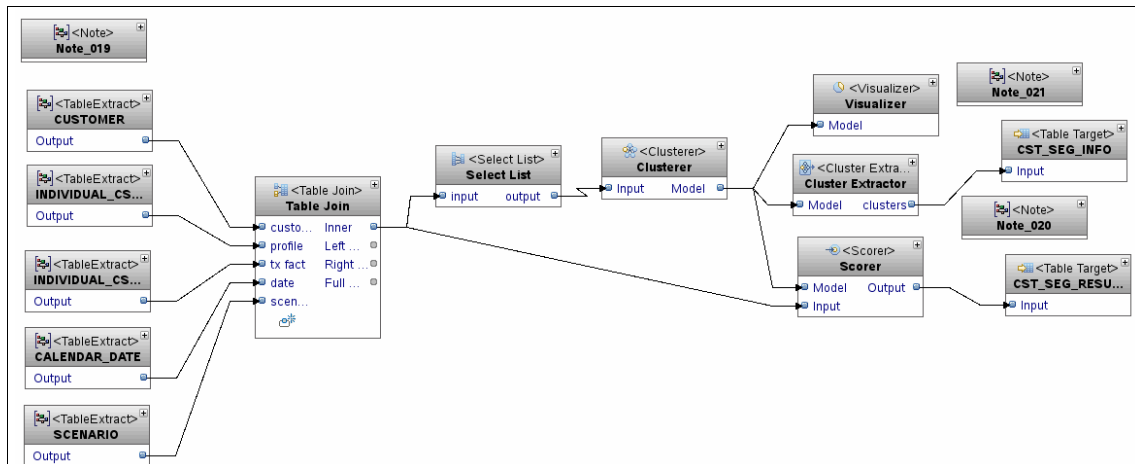


Figure 10-13 Data mining flow in Design Studio

Notice several items of interest in this mining flow:

- ▶ On the left side of the graphic there are several Table Source (extract) operators, such as Customer.
- ▶ These operators all feed into a multiway Join operator.
- ▶ The output of the Join operator feeds into a Select operator. The Select operator reduces from all of the columns in all the input tables down to only those columns required in the Clusterer operator.
- ▶ The Clusterer operator takes the input columns and performs the cluster mining method. The output of the Cluster operator goes into three operators in parallel.
- ▶ One of the operators that follows the Clusterer operator is the Visualizer. The Visualizer is most useful during model design and ad hoc mining operations. There are distinct visualizers for each data mining method. Each one is tailored to graphically display the results of the mining function. We saw examples of the cluster visualization during the discussion of data exploration and univariate and bivariate analysis.
- ▶ Another operator that follows the Clusterer operator is the Cluster Extractor. This operator extracts clusters from the Clusterer operator for insertion into a table for further analysis. It is the equivalent of performing a SELECT

statement on a table. These clusters are merely the descriptions of the clusters that were found in the data set.

- ▶ The output of the Cluster Extractor is a Table Target, which is a table that collects all of the cluster detail information.
- ▶ The third output of the Clusterer operator is the Scorer operator. The Scorer operator takes the Cluster model as one input and the data set (table) as a second input. The result is a mapping of customers in the dataset to their respective cluster. The scoring operator is what enables the model to be applied to new data records.
- ▶ The final output of the Scorer operator is another Table Target operator that collects the results of the Scorer operator, a mapping of customers to clusters.

Data mining modeling revisited

We previously provided an example data mining flow for the clustering method. In this section, we revisit the key aspects of the modeling process. This process is repeated for each mining method. The general steps of the process are as follows:

1. Extract the source data; this is shown in Figure 10-13 on page 381 as multiple tables sources feeding a multiway join operation.
2. The model operator (Clusterer, for example) operates on the input source data and generates output that can be processed in several ways.
3. The visualization operator can be used in two primary ways:
 - As a means of evaluating and validating the data mining model results for “correctness.” It can aid the developer in the process of tuning the mining model for effectiveness.
 - As a means of communicating the model results for use by users and business analysts.

The visualizer can provide an easy-to-grasp view of the results that allow important or anomalous results to stand out against the “noise” of extensive results.

There are visualizers for each data mining method in Design Studio, and they tend to flow from the mining operator.

4. The extractor operation simply distills the model results into a form suitable for insertion into a database table for standard relational query and reporting. This is an easy way to make the mining results available for integration into wider BI applications and reporting solutions, integrating with other data sources.

5. The scoring operation is what enables the developer to apply the mining model to a data set for either batch or real-time operation. Almost all operational BI scenarios use scoring.

Generally speaking, the data mining modeler uses the visualization and extractor features to design and test the model. The scoring feature can be used to validate the model against the test data set.

For an extensive discussion of data mining modeling in InfoSphere Warehouse Design Studio, see *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

10.2.4 Performing data mining scoring with other data mining models through PMML

InfoSphere Warehouse 10.1 supports the execution (scoring) of data mining models developed outside of InfoSphere Warehouse. These models can be developed on either IBM or non IBM tools using a similar process flow as that of InfoSphere Warehouse. The validated models are then exported to Predictive Model Markup Language (PMML), a markup language similar to XML. The PMML model can then be imported into InfoSphere Warehouse 10.1 and run on DB2 data in batch or real time like any other data mining model.

The process of importing the PMML model is straightforward. Locate the target database (the DB2 database on which you want the model to run) in the Source Data Explorer of the InfoSphere Warehouse 10.1 Design Studio and expand the **Data Mining Models** folder. Right-click the Data Mining Models folder and select **Import**. The dialog prompts for the location of the PMML file and imports the model into the DB2 database. After it is imported, it can be used like any data mining model in DB2. For more information about using PMML models, see *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813.

10.3 Extended data mining techniques using SAS Enterprise Miner

Thus far, we have described how data mining can be used in an operational data warehousing environment. We have also described how the data mining techniques in InfoSphere Warehouse 10.1 can be used to develop the mining models that can be used in those scenarios. In this section, we consider alternative modeling scenarios using tooling outside of InfoSphere Warehouse. Specifically, we consider predictive data mining using SAS Enterprise Miner.⁵

10.3.1 About SAS Enterprise Miner

SAS Enterprise Miner is an extensive toolset that supports and enables all of the stages of the data mining process that we have discussed in this chapter. Figure 10-14 on page 385 shows an example of the SAS Enterprise Miner workbench.

The modeling GUI looks similar to the Design Studio in InfoSphere Warehouse 10.1 and is composed of data mining flows that process and pass data from one operator to another.

SAS Enterprise Miner provides a robust data mining environment for

- ▶ Data exploration
- ▶ Data preparation
- ▶ Prediction
- ▶ Model comparison

In a traditional SAS Enterprise Miner workflow, processing and usage proceed as described here:

1. Extract a test sample data set from some source (for example, InfoSphere Warehouse). Use that test set to develop the predictive data model with SAS Enterprise Modeler.
2. Extract production data from data warehouse sources such as InfoSphere Warehouse.

⁵ Derived from a presentation “Change the game with in-database analytics: how to run SAS scoring models inside DB2” by Eileen Lin, Mattias Nicola, and Subho Chatterjee, that was given at the 2011 IBM Information on Demand® conference.

- Run the scoring model developed in SAS Enterprise Miner on that extracted production data set.

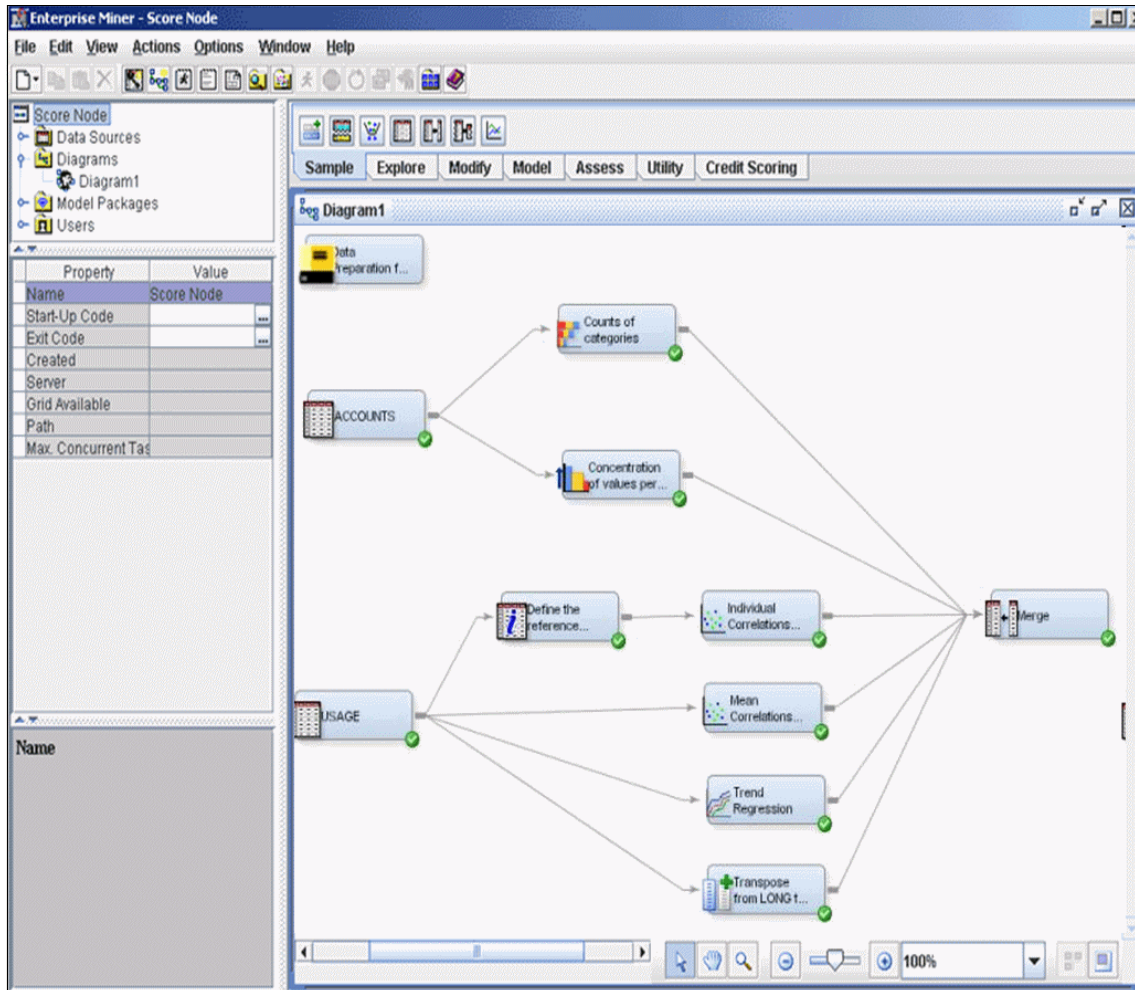


Figure 10-14 Example of the SAS Enterprise Miner workbench

This process is repeated as often as necessary, depending on data latency requirements and the rate of input of new data into the warehouse. This process is therefore problematic for the following reasons:

- ▶ Scoring is done on an extract of the data, which might already be out of date at the time of the execution.
- ▶ Scoring is processed in an outboard processor instead of near the data in the database engine.

- ▶ Data governance and data security are more difficult to manage after data has been extracted from the enterprise warehouse environment.
- ▶ Data duplication leads to increased storage requirements and costs. There are also concerns about consistency.

10.3.2 In-database scoring with SAS and InfoSphere Warehouse

A solution to the problems listed in the preceding section when using SAS Enterprise Miner and IBM InfoSphere Warehouse is to use new in-database features in DB2 from SAS.

In this scenario, the scoring models developed in SAS Enterprise Miner can be deployed using the new SAS Scoring Accelerator for DB2 that benefits from the SAS embedded process in DB2 to accelerate deployment and performance. The process is depicted in Figure 10-15.

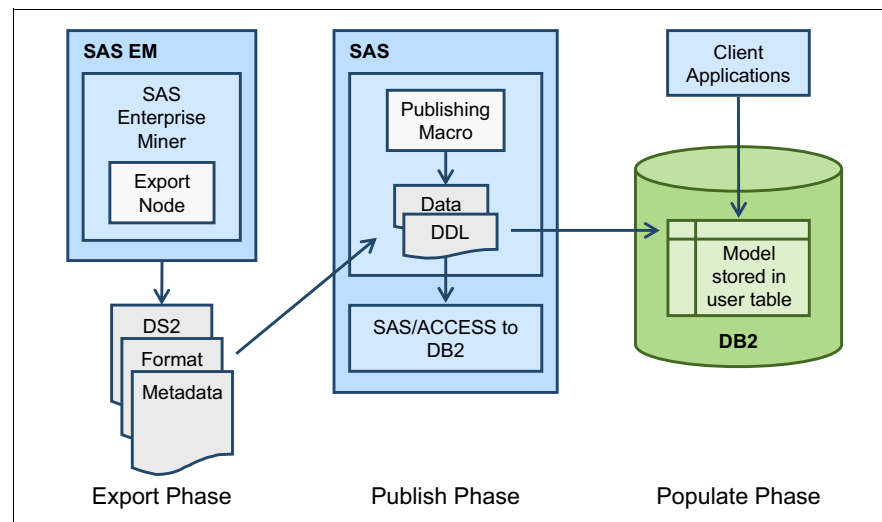


Figure 10-15 SAS Scoring Accelerator and SAS embedded process in DB2⁶

The process with the new accelerator and embedded process includes:

1. Export Phase

Export the scoring model from SAS Enterprise Miner into a DB2 mining metadata format.

⁶ Adapted from "Change the game with in-database analytics: how to run SAS scoring models inside DB2", Lin, Eileen; Nicola, Matthias; Chatterjee, Subho, Information on Demand conference 2011.

2. Publish Phase

Use the SAS server software to publish the scoring model to InfoSphere Warehouse (DB2)

3. Populate Phase

Register the SAS scoring model with the SAS embedded process in DB2, making it available natively to InfoSphere Warehouse applications.

Using the new embedded process allows for high performance execution of the scoring model in an operational warehouse environment with InfoSphere Warehouse 10.1.

10.3.3 Access the scoring model

The SAS embedded process was developed and installed by SAS, and runs as a fenced process in DB2 (db2sasep). The process is started and stopped using new **db2ida** DDL commands or can be configured to start and stop automatically with the DB2 instance. Examples of starting and stopping manually are shown in Example 10-1.

Example 10-1 Starting and stopping the SAS embedded process in DB2

```
db2ida -provider sas -start
```

```
db2ida -provider sas -stop
```

```
db2ida -provider sas -stopforce
```

After the process is running, scoring models can be invoked using the **ANALYZE_TABLE** function embedded in a SQL expression as shown in Example 10-2.

Example 10-2 Invoking the SAS scoring model

```
SELECT * FROM T1 ANALYZE_TABLE (IMPLEMENTATION 'PROVIDER=SAS;  
                                ROUTINE_SOURCE_TABLE=USER1.SOURCE_TABLE;  
                                ROUTINE_SOURCE_NAME=REGRESSION;');
```

Note the following:

- ▶ Scoring input (ROUTINE_SOURCE_TABLE) can be a table, view, or common table expression.
- ▶ The ROUTINE_SOURCE_NAME refers to the name of the model, which is REGRESSION in this example.

10.3.4 Additional scoring examples

In Example 10-2 on page 387, all columns of table T1 are used as inputs to the REGRESSION function invoked in the ANALYZE_TABLE function, and it retrieves every column of output.

In Example 10-3, only the ID and SIZE columns of table T1 are used as input to the scoring function, and all columns of output are retrieved.

Example 10-3 Scoring function with limited inputs

```
WITH INPUT AS (SELECT ID, SIZE FROM T2)
SELECT *
FROM INPUT ANALYZE_TABLE (IMPLEMENTATION 'PROVIDER=SAS;
                           ROUTINE_SOURCE_TABLE=USER1.SOURCE_TABLE;
                           ROUTINE_SOURCE_NAME=REGRESSION;');
```

Discover which columns are returned with EXPLAIN

The DB2 describe function can be used to determine which columns are returned by the scoring function as shown in Example 10-4.

Example 10-4 Discovering output columns using DB2 EXPLAIN

```
DESCRIBE SELECT * FROM T1
  ANALYZE_TABLE (IMPLEMENTATION 'PROVIDER=SAS;
                               ROUTINE_SOURCE_TABLE=USER1.SOURCE_TABLE;
                               ROUTINE_SOURCE_NAME=REGRESSION;');
```

Column Information

Number of columns: 3

SQL Type	Type Length	Column Name	Name Length
-----	-----	-----	-----
497 INTEGER	4	ID	2
453 CHARACTER	32	PROBABILITY	2
481 DOUBLE	8	CLASSIFICATION	2

This example shows that the ID, PROBABILITY, and CLASSIFICATION columns are returned by the scoring function REGRESSION with the given data types and lengths.

Joining scoring data with other table data

Although the scoring functions on their own are valuable, they are most valuable when the results are joined with other data in the data warehouse. For example, we can join the results of a scoring function called FRAUD with contents of a

Customer table to identify the relative RISK level of a particular customer. The SQL for this query is shown in Example 10-5.

Example 10-5 Joining scoring output with other data warehouse data

```
SELECT C.CUSTOMER_ID, C.RISK, L.LOAN_ID, L.LOAN_AMOUNT
FROM CUSTOMERS ANALYZE_TABLE (IMPLEMENTATION 'PROVIDER=SAS;
                                ROUTINE_SOURCE_TABLE=USER1.SOURCE_TABLE;
                                ROUTINE_SOURCE_NAME=FRAUD;') AS C,

    LOANS AS L,
WHERE C.CUSTOMER_ID = L.CUSTOMER_ID
AND C.RISK > 50
ORDER BY C.RISK DESC;
```

This SQL query returns an ordered list (by RISK from highest to lowest) of customers. Each record will list the customer, the risk score, the loan account number and the loan amount. The business analyst or loan officer can then take action on the high risk customers as is appropriate per business policy.

10.4 Deploy and visualize mining results with IBM Cognos 10 Business Intelligence

Regardless of the data mining technique used in the operational data warehouse, the data mining and scoring results must be presented to and visualized by users in some way to have business value. The results can be leveraged in a variety of ways including:

- ▶ Ad hoc visualization of the mining or scoring results.
- ▶ Scoring results can be used to augment dimensional or fact table records.
- ▶ Scoring results can be used to drive or trigger business processes or to influence decisions in business processes.

Visualization can take many forms, and we have seen examples of visualizations in the previous sections. We saw examples of the built-in visualizations in the InfoSphere Warehouse 10.1 Design Studio that are customized for each data mining method. We also saw examples of integration of scoring results using InfoSphere Warehouse and SAS using standard SQL techniques.

IBM offers IBM Cognos 10 Business Intelligence to provide the most robust end user and business analyst visualization and interaction with the data mining results. The Customer Insight InfoSphere Warehouse Model Pack includes a practical example of using Cognos to use data mining results. The Customer Insight Pack contains an example of customer segmentation using the clustering

method previously described. The clustering model operates on the sample data in the Customer Insight Pack and groups (segments) customers according to similar characteristics such as demographic aspects such as age, gender, and so on.

The data mining flow for the customer segmentation model example in the Customer Insight Pack is shown in Figure 10-16.⁷

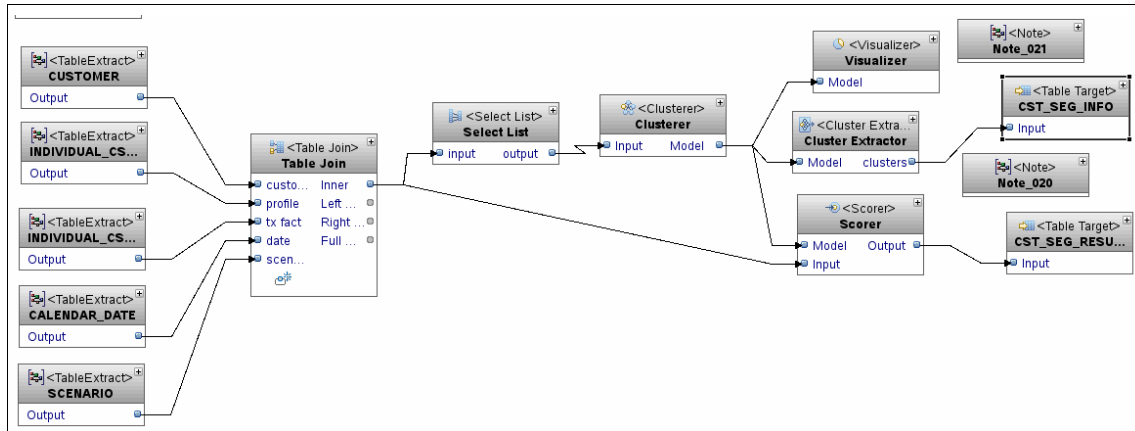


Figure 10-16 Customer Insight Pack segmentation (clustering) model

Recall the usage of the Cluster Extractor. This operator extracts all of the cluster details and writes them into the target table. After the cluster details are in a table, they can be modeled by Cognos Framework Manager and used in Cognos 10 Business Intelligence reports and applications.

⁷ A full description of the Customer Insight Model Pack and IBM Cognos 10 Business Intelligence is beyond the scope of this book. Refer to the product documentation for these offerings for full details.

Within the Cognos Framework Manager model for Customer Insight, a Cluster dimension is modeled, as shown in Figure 10-17.

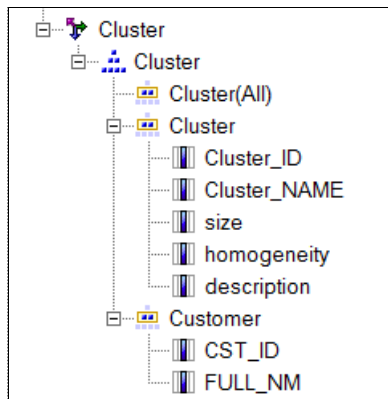


Figure 10-17 Cluster dimension in Cognos Framework Manager

Notice that in this dimension, there are two levels in the hierarchy, Customer at the leaf level and Cluster at a higher level of granularity. Customers therefore roll up into Clusters. This provides an alternative roll-up path for customers in addition to demographic attributes like age range or gender.

The Customer Insights Pack sample dashboard provides two reports that use the Cluster dimension made possible by the segmentation mining model:

- ▶ List of Clusters
- ▶ Breakdown by Cluster

10.4.1 List of clusters report

Figure 10-18 shows the List of Clusters report.

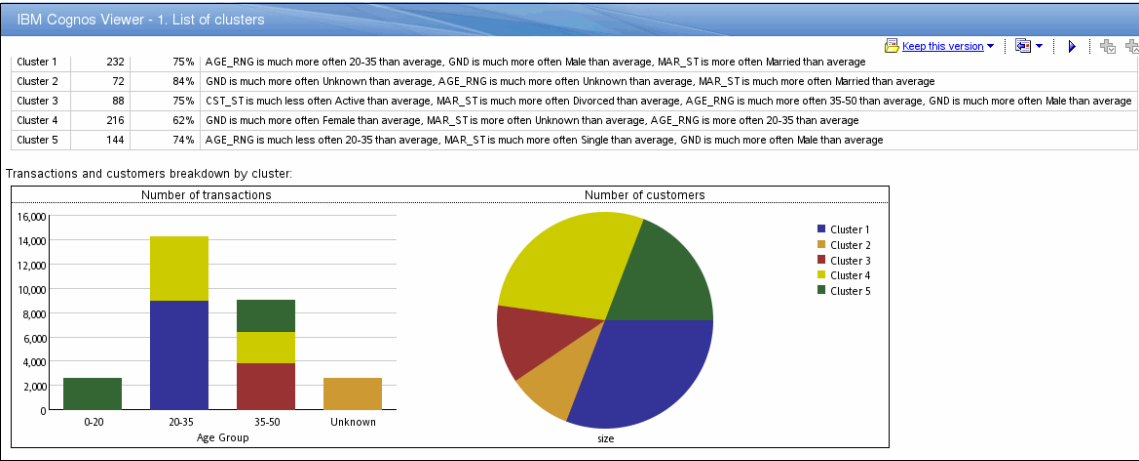


Figure 10-18 List of clusters report from Customer Insight Pack examples

The report has three main components:

- ▶ The report lists the top five clusters and provides metrics about the size and homogeneity, and a detailed description about each cluster.
- ▶ The pie chart shows the proportion of the total customer population that resides in each of the five clusters. This information clearly shows where the majority of customers fit, and can guide sales and marketing efforts to target the most customers with similar characteristics. This can increase the success rate of promotional campaigns and other customer-directed activities.
- ▶ The report also shows a bar graph that combines the cluster information with age groups, showing the number of sales transactions (or whatever transactions are relevant to our business) executed by customers in each age range (0-20, 20-35, 35-50 and “unknown”) in aggregate and also broken down by cluster. This type of chart can again make it easier to identify customer groups of interest based on their cluster characteristics *and* other demographic characteristics such as age range.

10.4.2 Breakdown by cluster report

Figure 10-19 shows a more detailed breakdown of income by cluster can be seen in the next report.

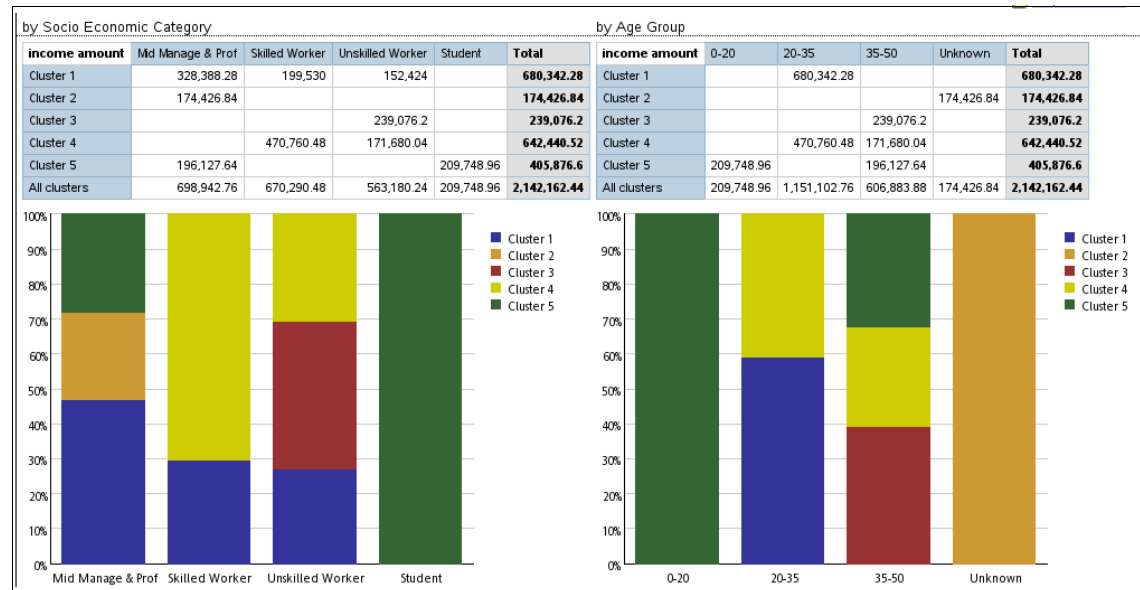


Figure 10-19 Income breakdown by cluster and profile

This report presents a further breakdown of the customer clustering by two profile categories: socio-economic category and age group. The data points for each cluster-profile category tuple is shown both in a cross-tab form and as a proportional bar graph. As with the earlier report, this type of visualization makes it easy for a business analyst or manager to quickly identify the customer group of interest to take some action or decision on that group.

The user and analyst of these reports does not see specifically the delineation of what data is generated by mining methods and what information is generated by other means or data collection. These examples demonstrate how data mining methods and results can be easily integrated into Cognos 10 BI reports and applications, thereby making the information consumable by the user or analyst without requiring them to be data mining experts.



Cognos Business Intelligence for InfoSphere Warehouse

InfoSphere Warehouse 10.1 Advanced Enterprise Edition includes licensing for IBM Cognos 10 Business Intelligence (BI) software. IBM Cognos 10 BI provides the business intelligence, analytics modeling, report authoring, application development, and dashboard deployment capabilities that help clients complete an operation data warehousing solution.

In this chapter we present a high level summary of the key features of IBM Cognos 10 Business Intelligence, including the following topics:

- ▶ An overview of the key features and interfaces
- ▶ An overview of the system architecture
- ▶ A more detailed discussion of Business Insights and Business Insights Advanced
- ▶ A drill-down on IBM Cognos Active Reports for disconnected reporting
- ▶ An overview of IBM Mobile support

11.1 Cognos Business Intelligence 10 and InfoSphere Warehouse

InfoSphere Warehouse Advanced Enterprise Edition 10.1 includes licenses of Cognos 10.1 Business Intelligence software. Cognos 10 BI provides a comprehensive business intelligence development and deployment platform that offers the ability to deliver advanced enterprise BI reporting and analytics applications and dashboards.

11.1.1 IBM Cognos 10 Business Intelligence features

IBM Cognos BI includes both web-based and Windows-based user interfaces that provide a BI experience that addresses the specific needs of various types of users. IBM Cognos BI also includes administrative tools and business modeling tools to aid in the development and deployment of BI applications. The main components are described here.¹

IBM Cognos Business Insight

With IBM Cognos Business Insight, you can create sophisticated interactive dashboards using IBM Cognos content and external data sources such as the InfoSphere Warehouse operational data warehouse and Cubing Services OLAP cubes, according to your specific information needs. You can view and open favorite dashboards and reports, manipulate the content in the dashboards, and email your dashboards. You can also use comments and activities for collaborative decision making and use social software such as IBM Lotus® Connections for collaborative decision making.

For more information about using IBM Cognos Business Insight see IBM Cognos *Business Advanced Insight User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_rptstd_fin.pdf

IBM Cognos Report Studio

IBM Cognos Report Studio is a robust report design and authoring tool. Using IBM Cognos Report Studio, report authors can create, edit, and distribute a wide range of professional reports. They can also define corporate standard report templates for use in IBM Cognos Query Studio, and edit and modify reports created in IBM Cognos Query Studio or IBM Cognos Analysis Studio.

¹ This chapter derives content from *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

For information about using IBM Cognos Report Studio see IBM Cognos *Report Studio User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cr_rptstd.pdf

IBM Cognos Query Studio

Using IBM Cognos Query Studio, users with little or no training can quickly design, create, and save reports to meet reporting needs that are not covered by the standard, professional reports created in IBM Cognos Report Studio.

For information about using IBM Cognos Query Studio, see *IBM Cognos Query Studio User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cr_qstd.pdf

IBM Cognos Analysis Studio

With IBM Cognos Analysis Studio, users can explore and analyze data from different dimensions of their business. Users can also compare data to spot trends or anomalies in performance. IBM Cognos Analysis Studio provides access to dimensional, online analytical processing (OLAP), and dimensionally modeled relational data sources. Analyses created in IBM Cognos Analysis Studio can be opened in IBM Cognos Report Studio and used to build professional reports.

For information about using IBM Cognos Analysis Studio, see *IBM Cognos Analysis Studio User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cr_pps.pdf

IBM Cognos Event Studio

In IBM Cognos Event Studio, you can set up agents to monitor your data and perform tasks when business events or exceptional conditions occur in your data. When an event occurs, people are alerted to take action. Agents can publish details to the portal, deliver alerts by email, run and distribute reports based on events, and monitor the status of events. For example, a support call from a key customer or the cancellation of a large order might trigger an event, sending an email to the appropriate people.

For information about using IBM Cognos Event Studio, see *IBM Cognos Event Studio User Guide*, which is available at the following site

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cr_es.pdf

IBM Cognos Metric Studio

In IBM Cognos Metric Studio, you can create and deliver a customized score carding environment for monitoring and analyzing metrics throughout your organization. Users can monitor, analyze, and report on time-critical information by using scorecards based on cross-functional metrics.

For information about using IBM Cognos Metric Studio, see *IBM Cognos Metric Studio User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_mm.pdf

IBM Cognos Administration

IBM Cognos Administration is a central management interface that contains the administrative tasks for IBM Cognos BI. It provides easy access to the overall management of the IBM Cognos environment and is accessible through IBM Cognos Connection.

For information about using IBM Cognos Administration, see *IBM Cognos Administration and Security*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cra.pdf

IBM Cognos Framework Manager

IBM Cognos Framework Manager is the IBM Cognos BI modeling tool for creating and managing business-related metadata for use in IBM Cognos BI analysis and reporting. Metadata is published for use by reporting tools as a package, providing a single, integrated business view of any number of heterogeneous data sources.

OLAP cubes (such as those developed in InfoSphere Warehouse Cubing Services) are designed to contain sufficient metadata for business intelligence reporting and analysis. Because cube metadata can change as a cube is developed, IBM Cognos Framework Manager models the minimum amount of information needed to connect to a cube. Cube dimensions, hierarchies, and levels are loaded at run time.

For information about using IBM Cognos Framework Manager, see *IBM Cognos Framework Manager User Guide*, which is available at the following site:

http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_fm.pdf

11.1.2 IBM Cognos 10 Business Intelligence architecture²

In this section we introduce and explore the architecture of IBM Cognos 10 Business Intelligence.

Cognos enterprise architecture

IBM Cognos BI delivers a broad range of business intelligence capabilities on an open, enterprise-class platform. All capabilities, including viewing, creating, and administering dashboards, reports, analysis, scorecards, and events, are accessed through web interfaces.

The IBM Cognos Platform is built on a web-based, service-oriented architecture (SOA) that is designed for scalability, availability, and openness. This n-tiered architecture is made up of three server tiers:

- ▶ The web tier
- ▶ The application tier
- ▶ The data tier

² This section is reproduced from Chapter 2 of *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

The tiers are based on business function and can be separated by network firewalls. Figure 11-1 shows the overall architecture.

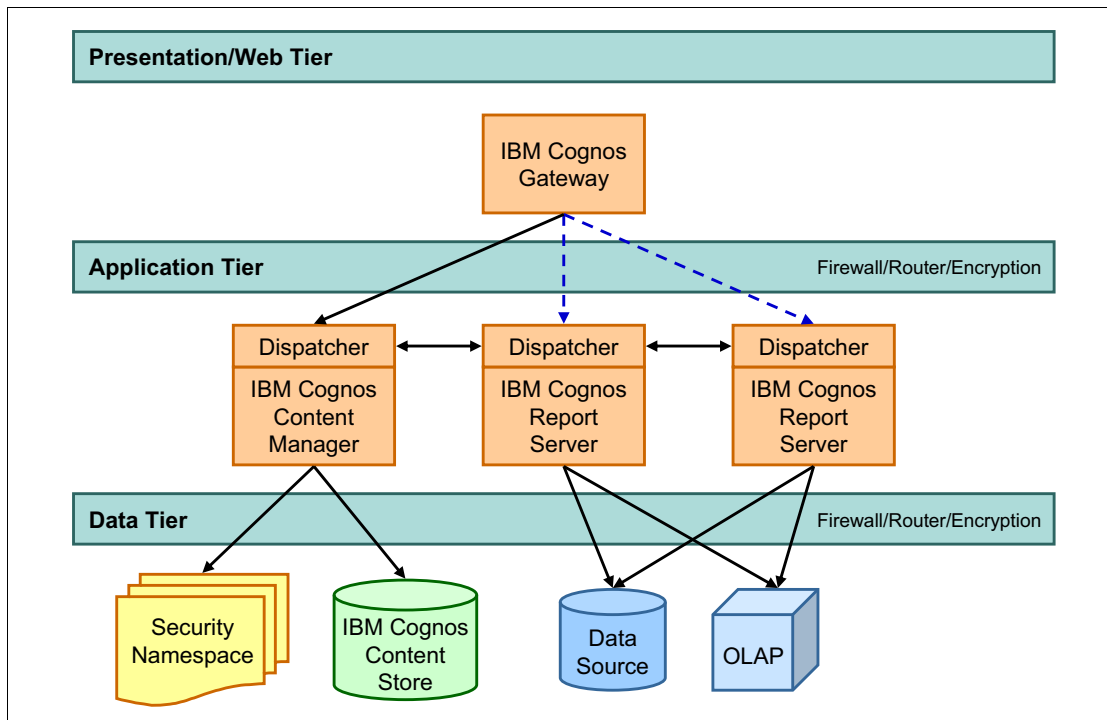


Figure 11-1 IBM Cognos Platform architecture

The main components of this architecture are described here.

Web tier: IBM Cognos Gateway

The web tier provides user session connectivity to IBM Cognos BI applications. The IBM Cognos components that fulfill this role are referred to as the IBM Cognos Gateway. The IBM Cognos Gateway component manages all web communication for the IBM Cognos Platform. The workload on the IBM Cognos Gateway server requires minimal processing resources. For high availability or scalability requirements, you can deploy multiple redundant gateways along with an external HTTP load balancing router.

Application tier: IBM Cognos Dispatcher

IBM Cognos Dispatcher performs the load balancing of requests at the application tier. The IBM Cognos Dispatcher component is a lightweight Java servlet that manages (and provides communication between) application services. At startup, each IBM Cognos Dispatcher registers locally available

services with the IBM Cognos Content Manager. During the normal operation of IBM Cognos BI services, requests are load balanced across all available services using a configurable, weighted round-robin algorithm to distribute requests. You can tune the performance of IBM Cognos Platform by defining how IBM Cognos Dispatcher handles requests and manages services.

Application tier: IBM Cognos Report Server

The main service that is responsible for application-tier processing is the report or query service. IBM Cognos Dispatcher starts IBM Cognos Report Server processes dynamically as needed to handle the request load. An administrator can specify the maximum number of processes that these services can start, and the minimum number of processes that should be running at non-peak times.

Application tier: IBM Cognos Content Manager

IBM Cognos Content Manager is the IBM Cognos Platform service that manages the storage of the following client application data:

- ▶ Security settings and configurations
- ▶ Server configuration settings
- ▶ Packages
- ▶ Dashboards
- ▶ Metrics
- ▶ Report specifications
- ▶ Report output

You use IBM Cognos Content Manager to publish models, retrieve or store report specifications, handle scheduling information and manage the IBM Cognos security name space. IBM Cognos Content Manager maintains information in a relational database that is referred to as the *content store* database.

11.2 Business modeling with IBM Cognos Framework Manager

The primary interface for data and business modeling in IBM Cognos 10 Business Intelligence is the IBM Cognos Framework Manager tool. This tool is discussed in considerable detail in Chapter 4, “Data modeling: End to end” on page 81 of this book and is thoroughly described in *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

11.3 IBM Cognos Business Insight

IBM Cognos Business Insight is the key web-based interface that allows you to create, open, edit, and interact with a business intelligence dashboard. Thus, the tool is used by both dashboard developers and report authors, and by users and analysts. It is highly interactive and intuitive. Figure 11-2 shows the essential elements of the interface.

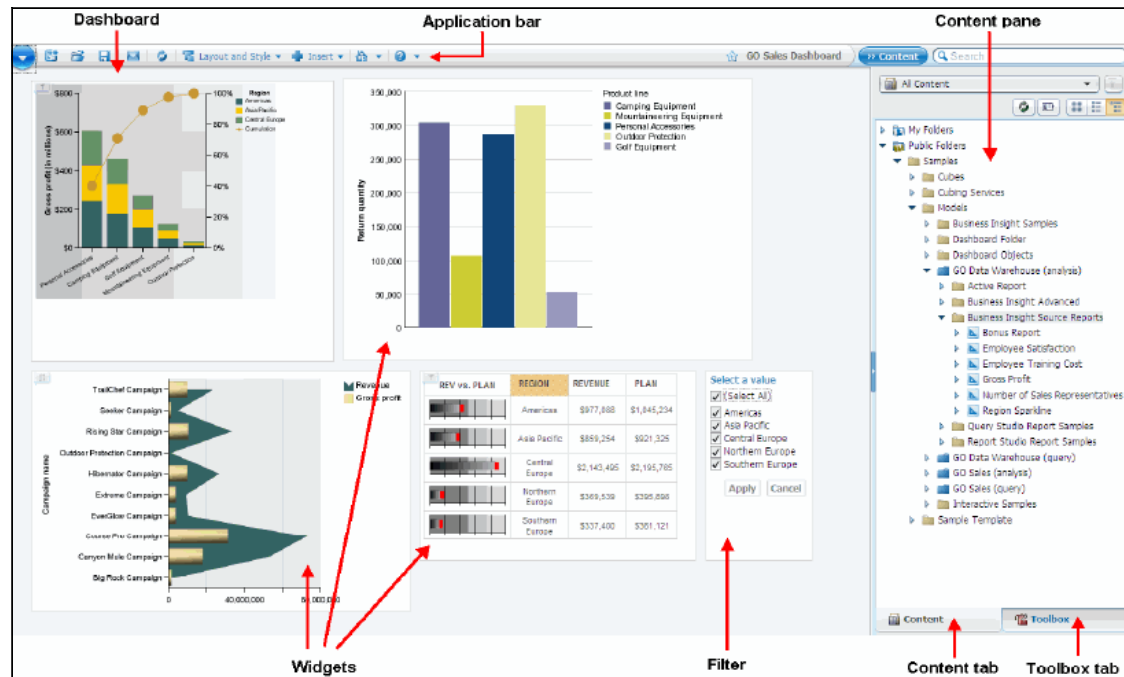


Figure 11-2 IBM Cognos Business Insights GUI layout³

Key areas of the GUI shown in the figure include the following items.

- ▶ **Application bar**

The application bar displays the name of the current dashboard and contains the icons for different actions that you can perform in the dashboard layout area.

- ▶ **Dashboard layer area**

The dashboard layer area is a workspace in which you can combine information from various sources to gain insight into your business. This is the

³ Figure derived from *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

main work area where various widgets with BI and non-BI content are arranged for optimal interaction with users.

► Content pane

The content pane contains multiple tabs of all the information that can be included in the dashboard area, including the following items:

– Content tab

The content tab displays IBM Cognos content in a hierarchy of folders and subfolders with dashboards that you can open and reports that you can add to a workspace

– Toolbox tab

The toolbox tab includes a palette of widgets that can be used in the dashboard, including:

- Widgets that can add additional content to the workspace of a business user, such as images, HTML content, or RSS feeds.
- Widgets that allow you to filter already added content, sliders, and select value filters.

► Widgets

Widgets are containers for all objects that you can add to the workspace. For business users, widgets allow the interaction and manipulation with the content that they contain, whether it is a report or a filter. There are two types of widgets provided, content widgets and toolbox widgets.

– Content widgets can contain IBM Cognos content, as listed here:

- Report widgets contain content authored in Report Studio, Query Studio, Analysis Studio, or Metrics Studio.
- Powerplay widgets contain content from IBM Cognos Powerplay Studio (OLAP).
- TM1 widgets contain content from IBM Cognos TM1® (in-memory OLAP).
- IBM Cognos Navigator is a navigation browser that displays IBM Cognos BI content such as folders, packages, and reports.

– Toolbox widgets are used for adding non-Cognos content to the workspace, either to add or filter information:

- Page widget is a toolbox widget to display HTML content in the dashboard.
- Image widget displays a single URL-accessible image in the dashboard.
- Inbox and RSS feed widgets display RSS feeds.

- Text widget displays text on the dashboard.
- Select value filter widget displays a select input panel with a list of items to select from to filter the dashboard content
- Slider filter widget displays a slider bar for filtering the dashboard content.

11.3.1 Interact with IBM Cognos Business Insight Editor

We can use the Customer Insight Pack samples to illustrate how users interact with IBM Cognos Business Insight to edit dashboards. If we open IBM Cognos Connection and navigate to a dashboard, we can open it in the Business Insights editor as shown in Figure 11-3.

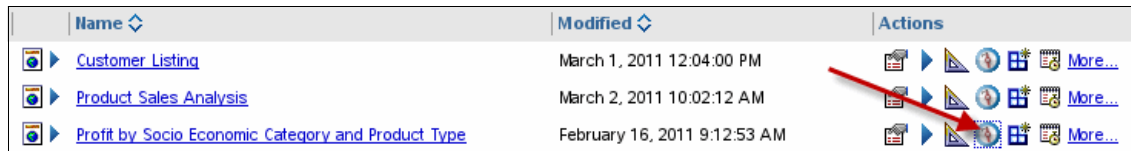


Figure 11-3 Opening a dashboard in the Business Insights editor

Clicking the icon for Business Insights as indicated in the figure presents the interface shown in Figure 11-4.

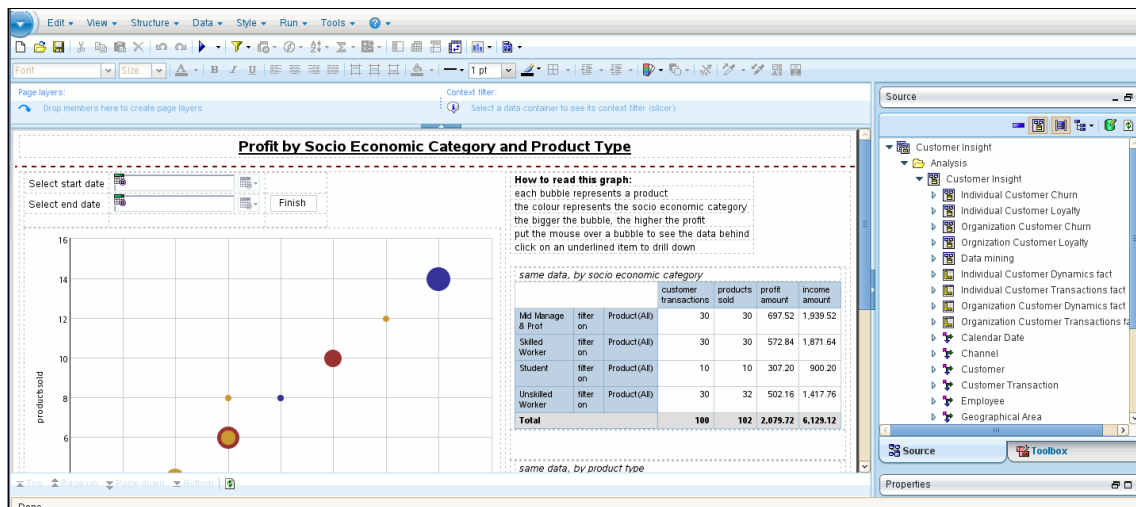


Figure 11-4 Business Insights editor for the "Profit by Socio Economic Category and Product Type" dashboard

Notice that the interface shows all the major elements we discussed in the previous section, including a set of graphical, tabular, and text widgets (including a filter widget) in the dashboard pane. The content area shows all of the business model content that can be included in the dashboard. Any user with access to the dashboard can open it in edit mode.

11.3.2 Run the IBM Cognos Business Insight dashboard

The dashboard can be run either from inside the editor or by clicking the dashboard name in the IBM Cognos Connection interface. Figure 11-5 shows the runtime dashboard from our example.

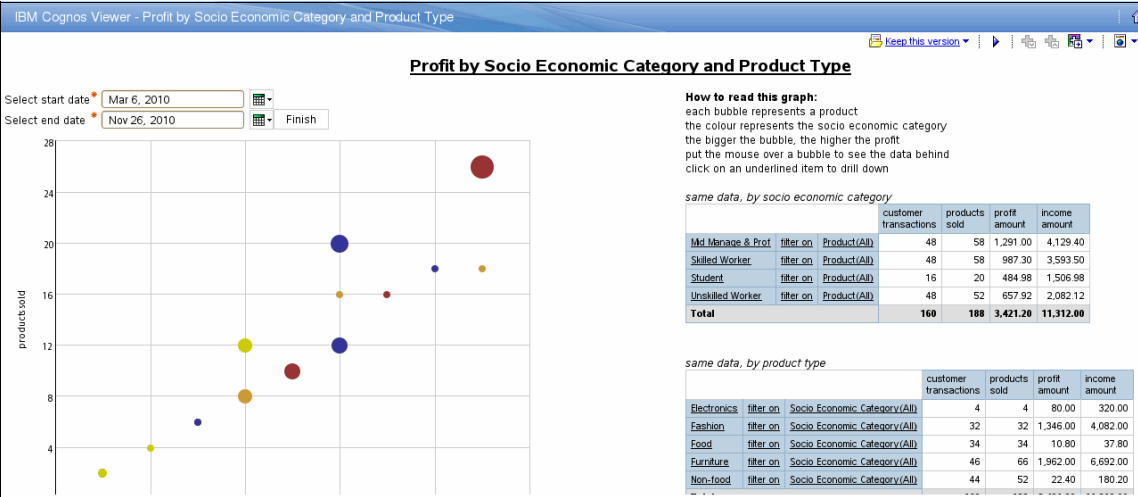


Figure 11-5 Example dashboard created with Business Insight

We can see that the dashboard appears as it did in the editor dashboard area. The dashboard is fully interactive. We can select a product in the graph at left

and drill down into that particular product. For example, if we select the large bubble in the upper right, we get the new dashboard shown in Figure 11-6.

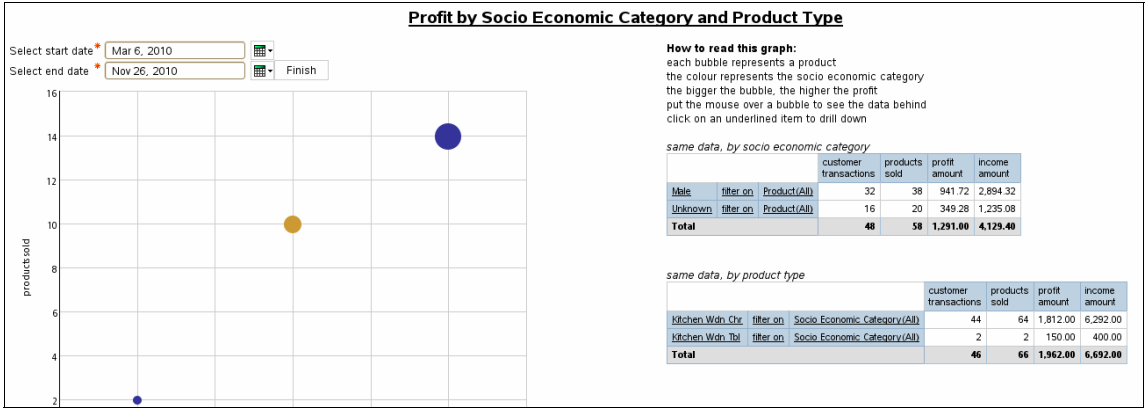


Figure 11-6 Drill down in IBM Cognos Business Insight dashboards

IBM Cognos Business Insights are described in much greater detail in *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

11.4 IBM Cognos disconnected report interaction

With IBM Cognos 10 Business Intelligence, it is possible to distribute interactive reports that are disconnected from their source data. This is a key advancement over being limited to static reports when disconnected.⁴ This feature is known as IBM Cognos Active Report.

11.4.1 IBM Cognos Active Report overview

IBM Cognos Active Report includes reports that are built for business users, allowing them to explore data and derive additional insight. IBM Cognos Active Report extends the reach of business intelligence and analytics to employees in the field and business partners. IBM Cognos Active Report can be consumed by users who are offline, making this an ideal solution for remote users such as the sales force.

You use IBM Cognos Report Studio to create active reports. IBM Cognos Active Report is a report output type that provides a highly interactive and easy-to-use managed report. Report authors build reports that are targeted at users' needs, keeping the user experience simple and engaging.

⁴ Content derived from *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

11.4.2 IBM Cognos Active Report features

IBM Cognos Active Report produces reports that are an extension of existing IBM Cognos Report Studio values and the IBM Cognos Platform. Users continue to benefit from the enterprise value of one version of the truth, and new interactive control types in IBM Cognos Report Studio serve as building blocks for creating active reports. You use active report controls to create the *layout* of an active report and to *filter* and sort data in the report.

Layout

To aid interaction with disconnected reports in IBM Cognos Active Report, Report Studio has the following layout controls available:

- ▶ Tab controls
- ▶ Deck of card controls
- ▶ Control data for hiding or showing list columns

Filtering

Filtering allows users to focus on the data that is most important to them, such as for their sales geography or product line. Filtering controls provided in IBM Cognos Report Studio support filtering include the following items:

- ▶ Lists and drop-down lists
- ▶ Chart interaction
- ▶ Radio buttons
- ▶ Check boxes
- ▶ Toggle buttons
- ▶ Push buttons

11.4.3 Enable an existing report for active reporting

An existing report can be enabled for active reporting. As an example, we can convert an existing report in the Customer Insight Pack samples into an IBM Cognos Active Report.

We navigate in IBM Cognos Connection to the folder **Public Folders** → **Customer Insight** → **Sample Reports** → **Customer Profile** → **Transaction Analysis**. After opening the report Transaction Listing in the Report Studio

editor, a window similar to Figure 11-7 is displayed. Note that the report is a simple list of products for a customer and year.

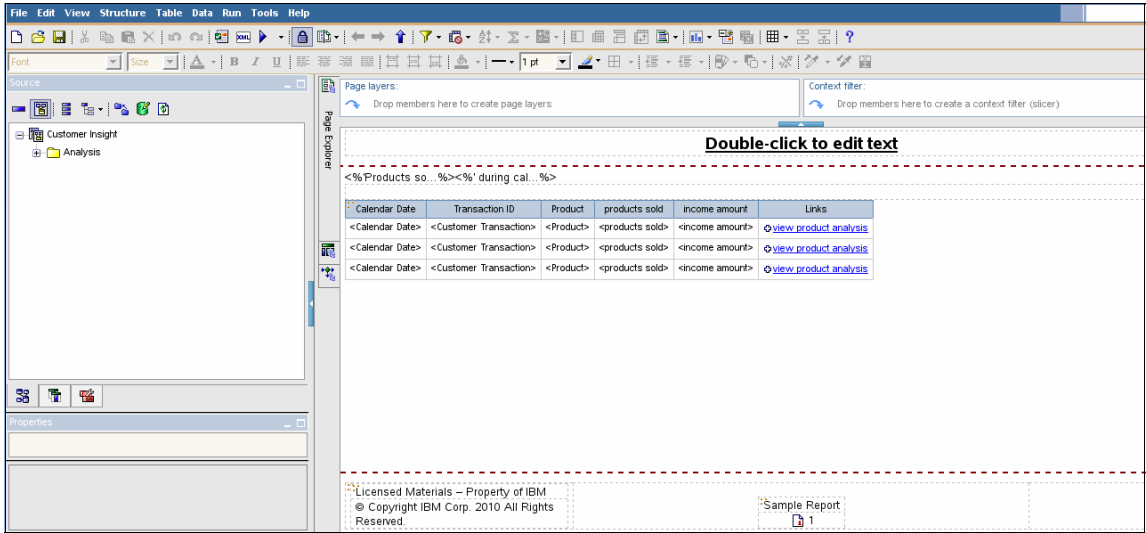


Figure 11-7 Transaction listing report before converting to active reporting

First we convert the report to an active report by selecting **File** → **Convert to Active Report**. Next we add a tab bar for Product. We insert a Data Tab Control as shown in Figure 11-8 to the left of the existing list widget.

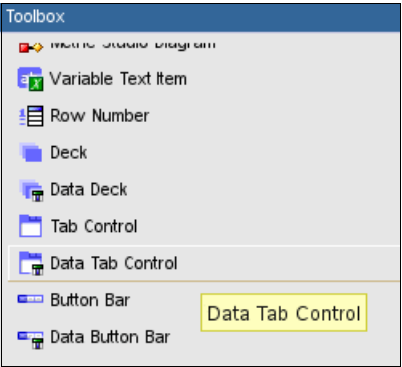


Figure 11-8 Data tab control widget

We drop the **Product** level label on the box that says “Drop Item Here.”
Figure 11-9 displays the new tab bar after insertion.

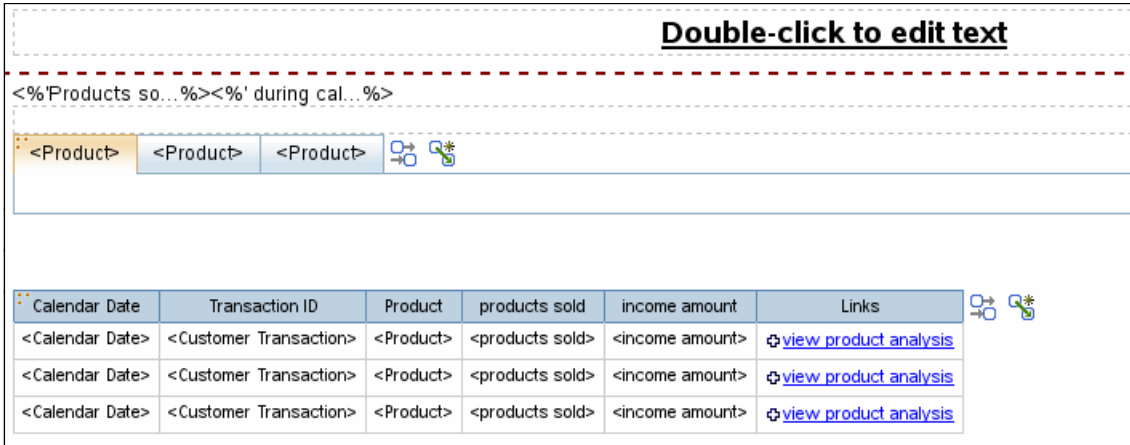


Figure 11-9 Tab control bar for product

We create a connection variable by clicking the **Interactive Behavior** icon (Figure 11-10).

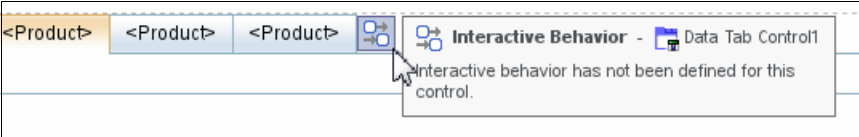


Figure 11-10 Interactive Behavior icon

We create a new variable connected to Product (Figure 11-11).

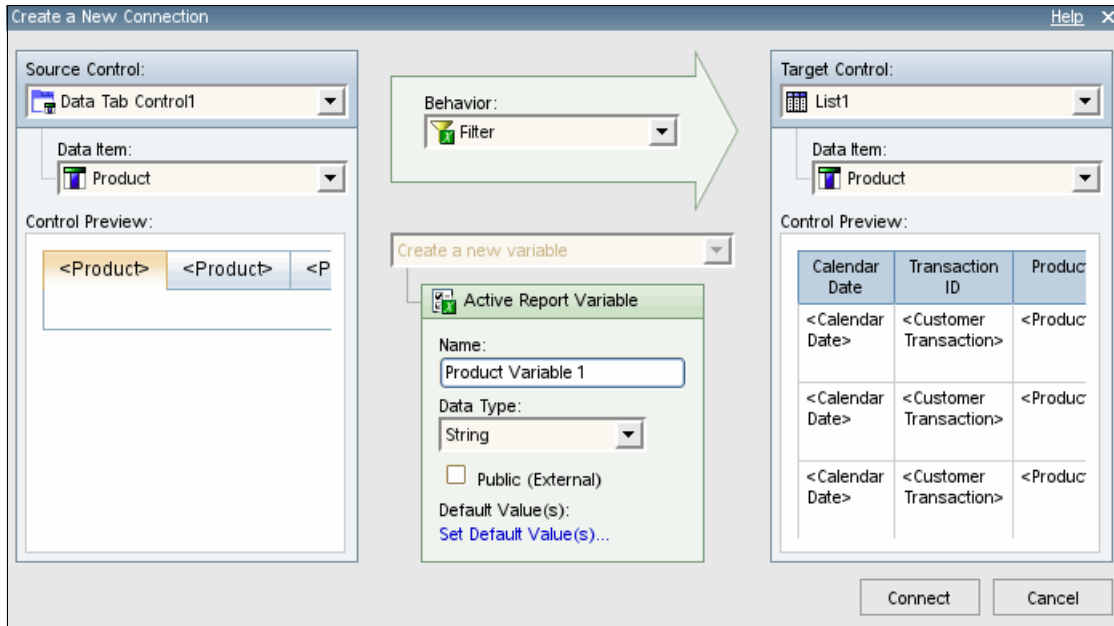


Figure 11-11 Product tab connection variable connecting to Product value

We now save and run the report, which generates a data file with an *.mht extension. The file can be sent to any user or it can be published centrally, as with other data file. It can be viewed in a browser by double-clicking it.

Note: If using Mozilla Firefox, you must install the noMHT add-on before the file can be displayed properly.

We can open the report file in a browser and navigate, as shown in Figure 11-12.

Potatoes 1kg	White Vinegar 300ml	Italian Penne 1kg	Decaff Special 200g	Frozen Corn Cobs 1kg	Low
Calendar Date	Transaction ID	Product	products sold	income amount	Links
2010-01-22	10451	Mens Jacket	1	155.00	view product analysis
2010-08-22	10381	Mens Jacket	1	155.00	view product analysis
2010-12-14	10063	Mens Jacket	1	140.00	view product analysis

Figure 11-12 Active report navigation

The report file is disconnected from the data source, but it is still interactive.

11.5 IBM Cognos access from mobile devices⁵

IBM Cognos Mobile provides timely and convenient access to IBM Cognos BI information from a mobile device. Mobile users want to take advantage of personal mobile devices while interacting with business intelligence on the device. This function provides decision makers with access to business-critical information wherever they are and whenever they need it.

IBM Cognos Mobile now supports the following mobile devices:

- ▶ Support for Apple iPhone, Apple iPad, and Apple iPod Touch

IBM Cognos Mobile now supports Apple iPhone, Apple iPad, and Apple iPod Touch devices. Users can use familiar iPhone actions to access the same business intelligence content and IBM Cognos Mobile features that are available on other devices in previous releases. In addition, users can create a list of favorites and select one dashboard or report to display automatically on the Welcome window when they start IBM Cognos Mobile.

- ▶ Support for RIM BlackBerry smartphones

IBM Cognos Mobile now supports the enhanced BlackBerry user interface for BlackBerry OS 4.2 and higher. The new user interface is easier to navigate and provides an improved overall experience when accessing IBM Cognos BI content.

- ▶ Support for Symbian S60 and Microsoft Windows Mobile 6.1 devices

IBM Cognos Mobile continues support for Symbian S60 and Microsoft Windows Mobile 6.1 devices.

IBM Cognos Mobile for the Apple iPhone and Apple iPod Touch is an HTML 5 web application. It enables the following functions:

- ▶ Rapid mass deployment to enterprise mobile users
- ▶ Faster device support to sustain the speed at which new devices enter the marketplace
- ▶ Instantaneous software updates that occur at the server

You do not need to update mobile client software, which makes deployment transparent to users.

⁵ This section is reproduced from *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.

11.6 Further information

For more detailed information about all of the authoring, reporting, analytics, and administrative features of IBM Cognos 10 Business Intelligence, see the online documentation and *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912.



InfoSphere Warehouse resilience with Optim Configuration Manager

One of the major benefits of InfoSphere Warehouse Advanced Edition is the inclusion of a number of additional tools aimed at helping to manage the advanced complex environments that make up a Data Warehouse. One such included tool is Optim Configuration Manager.

In this chapter we outline both the benefits and functionality that Optim Configuration Manager brings to your environment.

12.1 Challenges to maintaining an operational warehouse

In today's business environment, one method that companies use to maintain or increase profitability is to continually drive down costs. This usually translates to accomplishing more with fewer resources. When these aspirations are applied to the data warehouse environment, companies want to store larger amounts of data preferably on the same amount of storage, or maximize the performance of their hardware. Or even to reduce the number of administrators required to maintain the solution.

Therefore, the pressure on a database administrator to achieve these goals never diminishes. As such, DBAs must introduce and use a range of tools if they are ever to meet these targets. Optim Configuration Manager is one such tool that IBM offers to assist DBAs in improving their efficiency, while also helping them understand their complex environment.

In a production environment, it is critical that a database structure remains efficient by reducing the risk of performance degradation or more importantly, preventing system failure. There are many reasons for failures to occur, from unregistered system changes to uncoordinated application updates. Whatever the reason, these types of production issues can be costly to a company, not only in system downtime and lost production, but also in damage to the reputation of the company.

An additional benefit of implementing Optim Configuration Manager is the capability to help maintain an efficient database structure and thus reduce the risk of these types of failures.

12.2 InfoSphere Optim Configuration Manager

IBM InfoSphere Optim Configuration Manager is an advanced database management tool from the Optim family of applications that is used to assist organizations and database administrators in the performance of their roles. The tool can be purchased either individually from IBM or in a more cost-efficient solution as part of InfoSphere Warehouse Advanced Edition.

The key role of InfoSphere Optim Configuration Manager is to provide a central store and easy-to-use web interface for discovery, storage, and management of database and client configuration information. This type of information can include, for example, host names, IP addresses, user IDs, and DB2 client versions details.

The tool also provides the features to:

- ▶ Define standard client connection definitions for databases, including database location and driver properties. Storing this information in a central location within the Optim tool metadata database improves manageability and allows DBAs to track client access to data servers.
- ▶ Allow DBAs to set and modify properties of deployed database clients to improve database transaction performance and allocate data clients among data servers for the workload balance that you want by redirecting clients to alternative data servers. This feature allows users to maintain application performance and availability to meet production objectives.
- ▶ Provide DBAs with the tools required to track changes of database and client configurations to determine the root cause of performance degradation or outage of a production application. Optim Configuration Manager offers the capability to redirect clients and offending applications to the alternative high availability data servers when the production server is not available.

This same feature can also be used to redirect client connections while system maintenance is being performed on a data server, or to migrate client connections from an older data server version to a new one.

- ▶ Provide DBAs with the ability to reduce the maximum number of simultaneous connections to a driver and throttle an application to ease certain performance issues.
- ▶ Analyze database servers and their storage with the aim of making recommendations on either which compression type to apply to a data set, or in implementing a multi-temperature storage solution. These capabilities help to save costs related to additional hardware and software.

InfoSphere Optim Configuration Manager must be installed either on its own hardware platform or as part of a shared Optim tools server platform. If installed as part of a shared Optim tools server platform, a number of subcomponents of the Optim applications are only installed one time and then shared by the additional installed Optim tools.

An example is the DB2 relational database that is installed to host the metadata repository; any additional Optim tools installed will use the same repository for their schemas and tables. Figure 12-1 on page 416 shows the major architectural elements of Optim Configuration Manager, consisting of a metadata repository

and a self-contained Java application that provides both the web interface and the functionality of the tool.

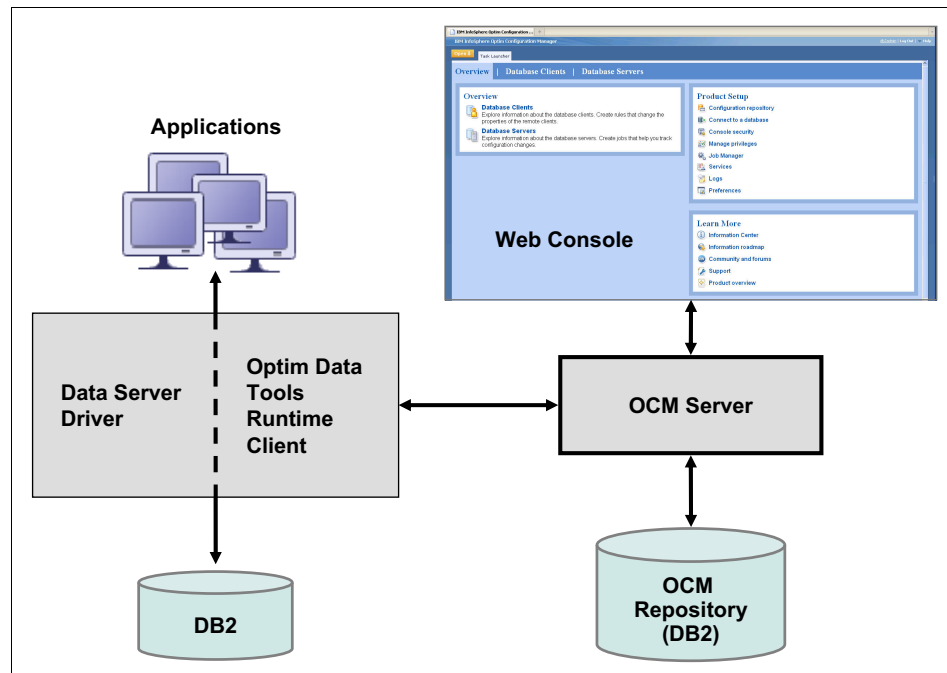


Figure 12-1 Optim Configuration Managers architecture

When installed, Optim Configuration Manager is configured to connect to various data sources within your organization. An example is the DB2 relational database that makes up your InfoSphere warehouse environment.

You can also set up an automated process that collects information such as database configuration properties, host name, IP address, instances, subsystems, operating system name and version, DB2 version, and fix pack information. This provides you with insights about how your environment was changed over time. You can now view, from a central location, the complete database configuration details of all the databases within your environment, rather than having to connect to each separately.

Optim Configuration Manager can also discover which clients and applications are connecting to a particular database and store this information within its metadata repository. Thus, from the different database servers, a complete picture of which clients and applications are connecting to which servers can be determined and stored. This type of record is of great benefit during such tasks as system upgrades or fix pack installation.

Additionally, the clients and applications can also be configured directly within the Optim Configuration Manager to allow for better management of their functionality. For Optim Configuration Manager to perform advanced client tasks such as data source redirects, performance monitoring, and system configuration changes, the advanced Optim data tools runtime client must be installed in place of the existing data driver on the client.

In this configuration, when a client connects to a data source, the Optim data tool driver first connects to the Optim Configuration Manager. This manager then informs the driver if any change such as a redirect is to be performed. If so, it provides the details of such a redirect. Otherwise, the client is directed to connect to the original data source. This extension to the data source driver to add Optim Configuration Manager awareness allows configuration information to pass back to the Optim Configuration Manager repository and the command instructions back to the client.

To access Optim Configuration Manager, open a web browser and specify a URL that consists of the host name of the server and the port specified during the installation of the tool. The default port is normally 12206, thus, for our example we enter:

`http://9.26.167.55:12206`

After the user details are entered the initial panel of Optim Configuration Manager is displayed; see Figure 12-2.

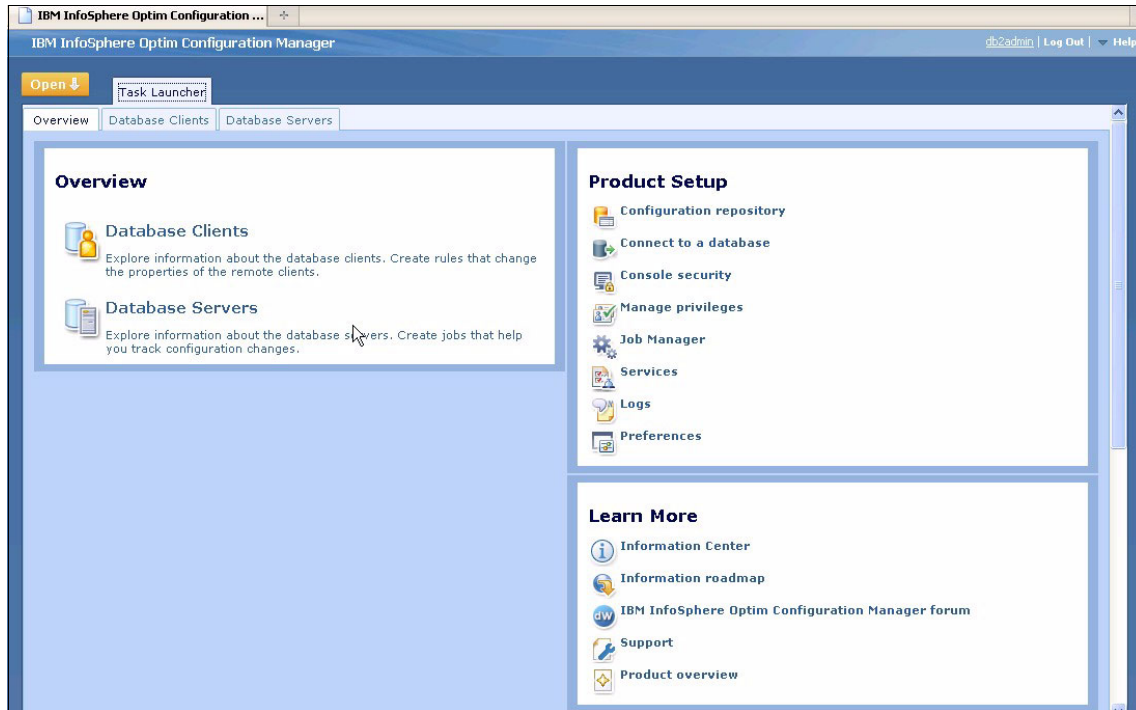


Figure 12-2 Initial page for Optim Configuration Manager

12.3 Integration of the Optim family of tools

Optim Configuration Manager is only one of a number of applications that make up the Optim family of database tools. And as mentioned in 12.2, “InfoSphere Optim Configuration Manager” on page 414, members of the Optim family of tools can share elements of their infrastructure to form a partnership solution.

This sharing is applicable to both the server and client elements of the tool set; for example, sharing the same metadata repository on the tools server. Another example is the use of the Optim data tools runtime client by multiple Optim tools.

But sharing code and saving space are not the only benefits that the integration of these tools provides. This family of Optim tools was developed to work in partnership to help solve many of the issues and problems that a single tool cannot overcome.

Consider an example of where the Optim Configuration Manager works in partnership with Optim Performance tools to help migrate the configuration and data connection details for a new distributed application that accesses DB2, from a development environment to its place in a production domain.

You can use Optim Performance Manager to ensure that the new application connects to data sources and the driver pool efficiently with the appropriate connection and driver properties by testing and adjusting as necessary. As soon as the new application is determined to be performing at its peak, which is validated using Optim Performance Manager, you can use Optim Configuration Manager to ensure the data client configuration is migrated to the production environment.

Figure 12-3 shows a scenario of Optim Configuration Manager and Optim Performance Manager integrated to deploy a new application.

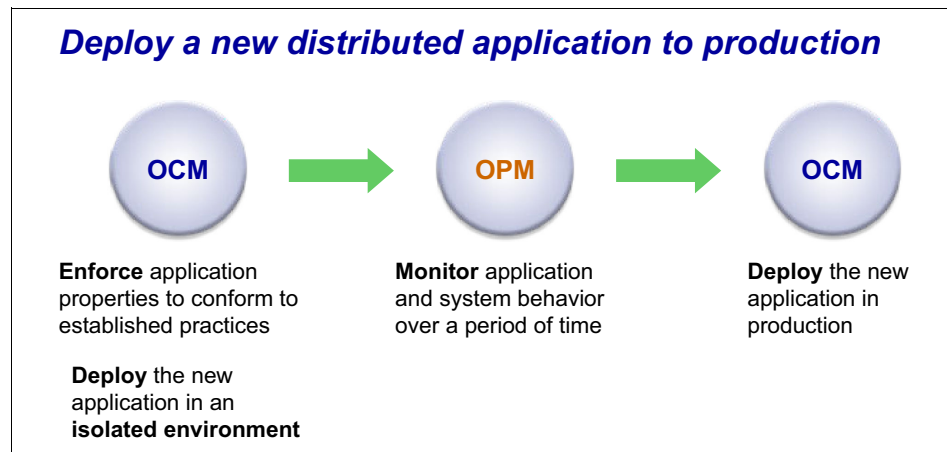


Figure 12-3 Optim Configuration Manager integration with Optim Performance Manager

Another scenario of Optim Configuration Manager working in partnership with Optim Performance Manager is for problem determination, containment, and resolution. These tools can work together to identify a misbehaving application, quickly contain the damage, find and fix the problem, and then return everything to good working order.

By using Optim Performance Manager you can determine and identify a performance degradation. To contain the damage you can use Optim Configuration Manager to isolate the problematic transactions that cause harm to the transactions of another application. You can use Optim Configuration to help diagnose whether the problem is due to a database configuration change, or you can further diagnose the problematic transaction or query with Optim Query Workload Tuner.

After a fix is instigated, you can use Optim Configuration Manager to route the previously misbehaving application transactions back to the fixed application.

Figure 12-4 illustrates Optim Configuration Manager integrated with Optim Performance to solve a performance issue.

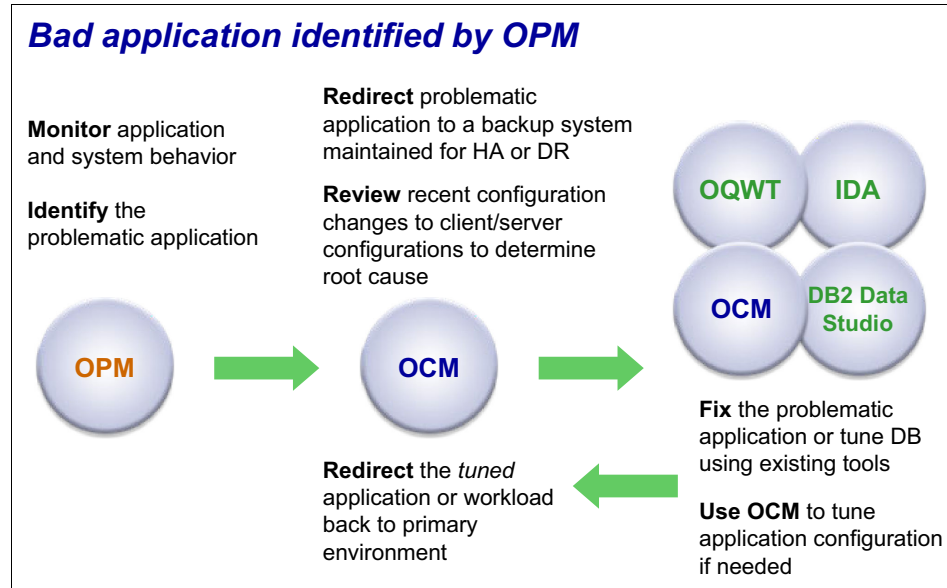


Figure 12-4 Use case 2: Optim Configuration Manager integration with Optim Performance

12.4 Monitor a DB2 database configuration

In this section we provide an example of using Optim Configuration Manager to keep track of changes made to the configuration of a DB2 database. This allows DBAs to keep both historical records and an audit trail of changes made to a database configuration. This example also illustrates how monitoring a system can be used to detract changes that can then be applied across a range of systems to help IT staff to maintain the environment synchronized.

Our data warehousing lab environment consists of two virtual IBM Smart Analytics Systems built with VMware. The IBM Smart Analytics Systems include InfoSphere Warehouse Advanced Edition 10.1 and IBM Cognos 10.

We demonstrate the Optim Configuration Manager functions with our lab systems described in Chapter 3, “Scenario: Warehouse Live Demo Set” on page 69. For more details about how to manage complex query workload, see

Chapter 6, “Managing complex query workloads in an operational warehouse environment” on page 237.

In this scenario, we create an Optim Configuration Manager job to monitor changes on the database server and the database. We then make changes on the virtual machine, database instance, and database to demonstrate that Optim Configuration Manager is also capable of identifying changes made to the configuration of database.

Follow these steps to create a configuration monitoring job in Optim Configuration Manager:

1. Create a monitoring job.

Log in to the Optim Configuration Manager web interface. Click **Open** → **Setup** → **Job Manager**. Click to **Add Job**. Under type, select **Configuration Management**, give a name to the job, and click **OK**.

Configuration Management is the Optim Configuration Manager job type that allows you to explore information about your systems, databases, database servers, and database clients.

Figure 12-5 shows adding a Configuration Management job.

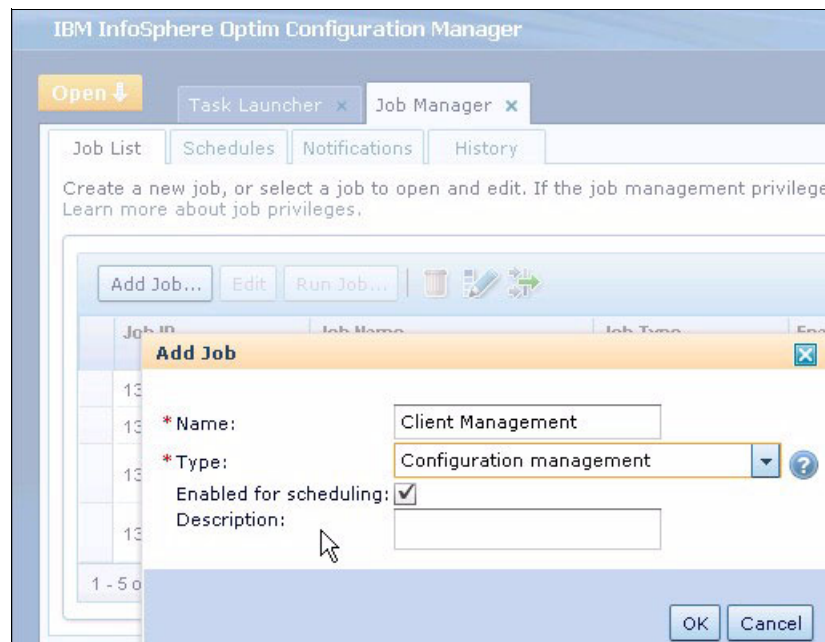


Figure 12-5 Configuration Management job

2. Specify the monitor objects.

The Job components configuration panel displays. Select the **Configuration management** tab to specify the configuration objects to be monitored. You can use the available templates (Database, Authorization, or Explore), or select the individual object you want Optim Configuration Manager to monitor. See Figure 12-6.

*Client Management x

Save All Run Job...

Job Components

- Properties
- Configuration management**
- Schedules
- Chain
- Notifications

Specify the configuration objects that you want to monitor. You can select a template represented by a template, you can select the individual objects that you want the pr

Templates

☒ Database ☒ Authorization

Database Objects

<input checked="" type="checkbox"/> Column	<input checked="" type="checkbox"/> Index
<input checked="" type="checkbox"/> Table	<input checked="" type="checkbox"/> Table space

Authorization Objects

<input checked="" type="checkbox"/> Column Authorization	<input checked="" type="checkbox"/> Database Aut
<input checked="" type="checkbox"/> Role Authorization	<input checked="" type="checkbox"/> Routine Autho
<input checked="" type="checkbox"/> Table Authorization	<input checked="" type="checkbox"/> Table space A

Explore Objects

☒ Client, Server, and System Objects

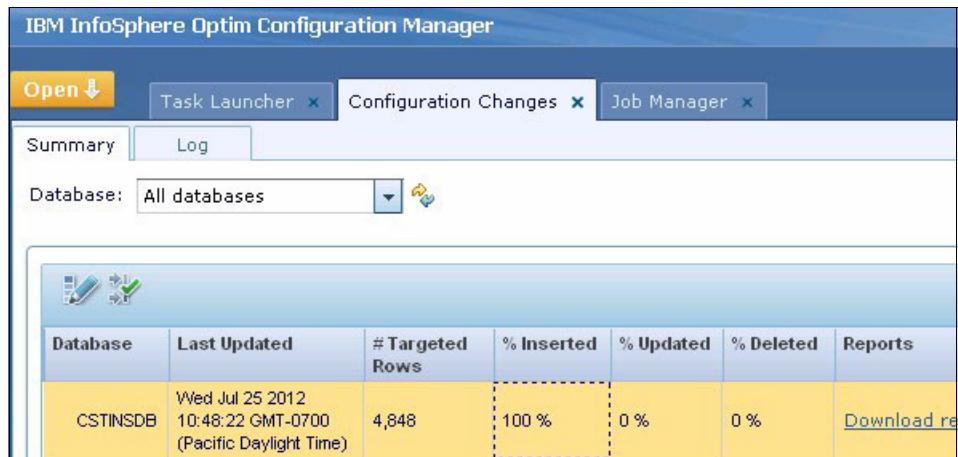
Figure 12-6 Properties of the configuration job

In the component configuration panel you can also specify other characteristics for the configuration monitoring job, such as creating a schedule, creating a notification rule to send email notifications, or configuring a job chain. A job chain allows you to string jobs to be executed in sequence, or to trigger an error handling routine if your job fails.

3. Save and run the monitoring job.

Still in the Optim Configuration Manager web interface, click **Open** → **Explore** → **Configuration Changes**. This shows a report of what Optim Configuration Manager found in both system and database. Because this is the first run, all the data that Optim Configuration Manager found is considered as inserted items. The Summary tab displays the general information about the job.

Figure 12-7 shows general information about an executed configuration job.



The screenshot displays the IBM InfoSphere Optim Configuration Manager interface. At the top, there's a title bar with the application name. Below it, a navigation bar includes an 'Open' button and three tabs: 'Task Launcher', 'Configuration Changes', and 'Job Manager'. The 'Job Manager' tab is active. Underneath, there's a 'Summary' section with a 'Log' button. A 'Database:' dropdown menu is set to 'All databases'. The main area features a table with job execution details for the 'CSTINSDB' database. The table has columns for Database, Last Updated, # Targeted Rows, % Inserted, % Updated, % Deleted, and Reports. The 'CSTINSDB' row shows a last update on July 25, 2012, at 10:48:22 GMT-0700, with 4,848 targeted rows, 100% inserted, and 0% updated or deleted. A 'Download reports' link is available in the Reports column.

Database	Last Updated	# Targeted Rows	% Inserted	% Updated	% Deleted	Reports
CSTINSDB	Wed Jul 25 2012 10:48:22 GMT-0700 (Pacific Daylight Time)	4,848	100 %	0 %	0 %	Download reports

Figure 12-7 Job general information

You can download the generated reports and view the detailed information gathered by Optim Configuration Manager. The reports contain the

information related to objects that you have configured your job to explore. Figure 12-8 shows a list of output from the monitoring job.

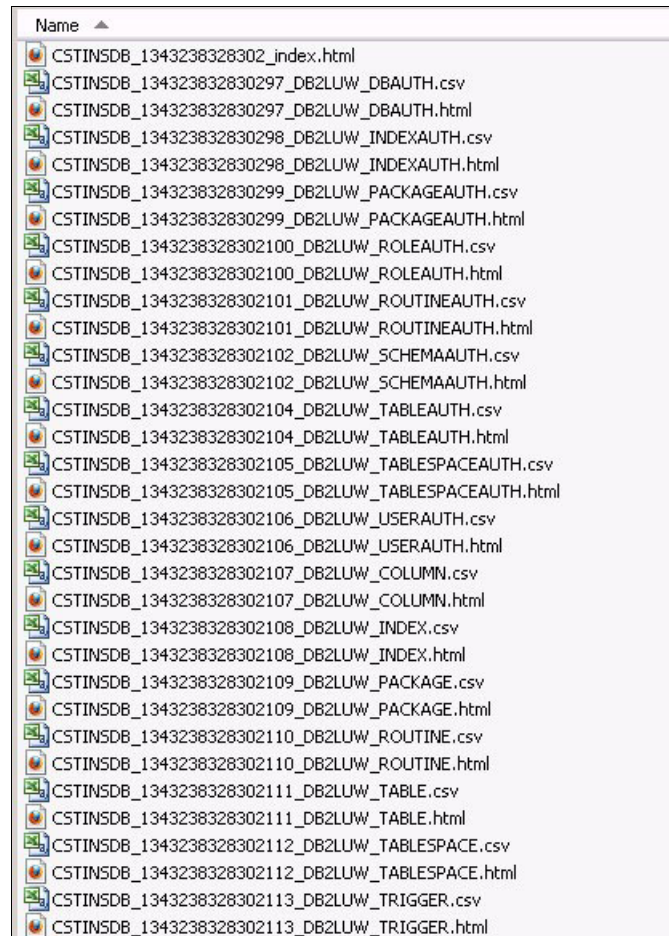


Figure 12-8 Reports generated by the configuration monitoring job

Now we increase the virtual machine memory size and add memory to the DB2 instance because memory constraint is one of the more common factors that affect system performance. MQT tables are commonly used in data warehousing systems for faster querying. So we also create two MQT tables to demonstrate how Optim Configuration Manager can keep tracking the changes in a database server operating system, database instance, and database.

We then run the Optim Configuration Manager Configuration manager jobs again. Optim Configuration Manager identifies what changes have occurred.

Under the **Log** tab, you can access all items that the configuration's Optim Configuration Manager is tracking. You can select each one you want to view from the drop-down menu as shown in Figure 12-9.

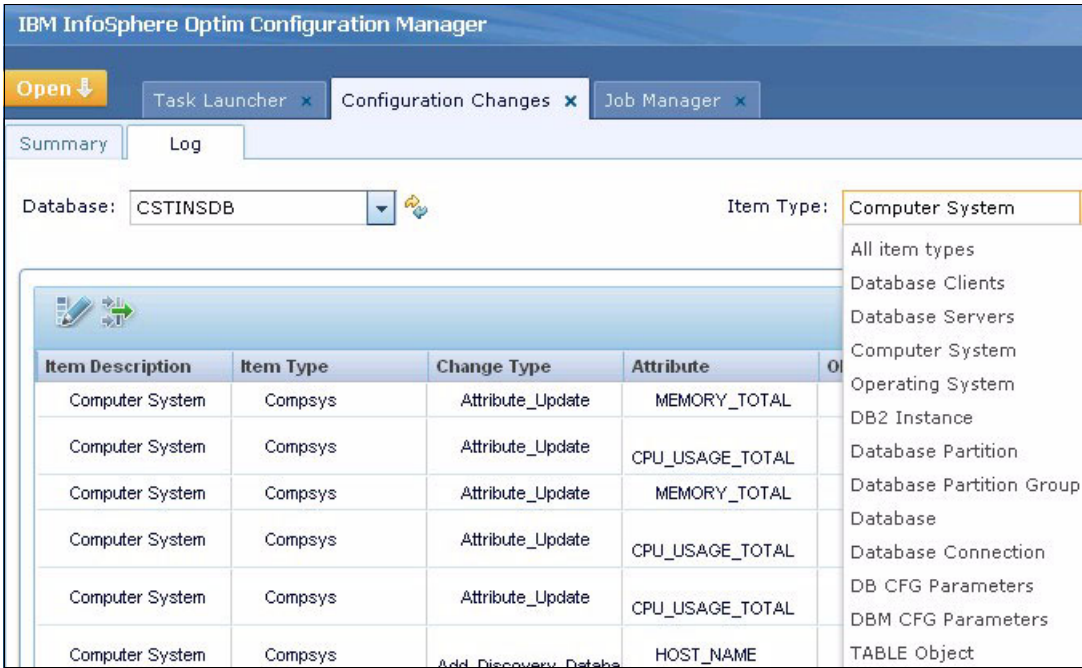


Figure 12-9 Job detail information page

When we select the Computer System type, we see the memory change we made as shown in the Old Value and New Value columns in Figure 12-10.

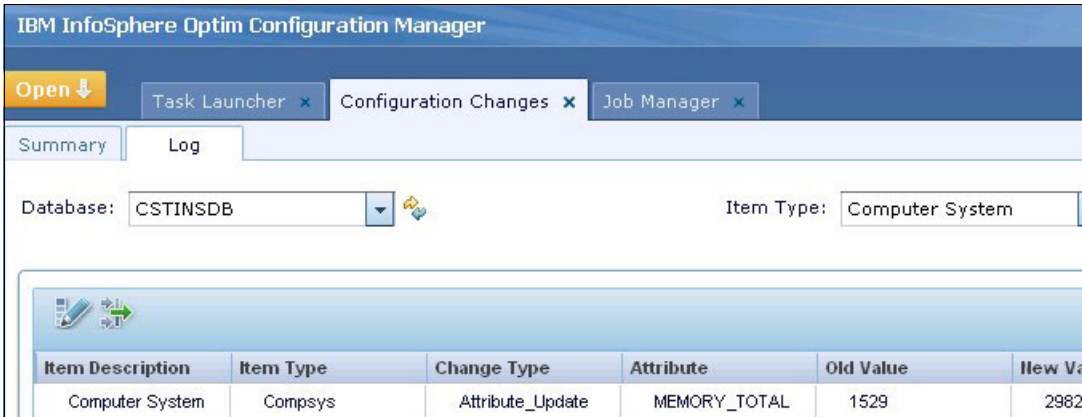


Figure 12-10 Computer System changes identified by the job

The second change that we made was to add memory to DB2 instance; see Figure 12-11.

IBM InfoSphere Optim Configuration Manager

Open ▾


Task Launcher x

Configuration Changes x



Job Manager x

Summary

Log

Database: CSTINSDB ▾ 

Item Type: DBM CFG Parameters ▾

Item Description	Item Type	Change Type	Attribute	Old Value	New Value
Instance Configuration Parameters	InstConfigParameters	Attribute_Update	instance_memory	293469	598054

Figure 12-11 Database changes identified by the job

Figure 12-12 shows MQT that we created have been identified by the configuration monitoring job.

Views Summary / Database: CSTINSDB / Job ID							
Object type	# Rows	Inserted	Updated	Deleted	Unchanged	Start time	End time
Views .CSV	2	2 (100.0%)	-	-	-	7/25/12 12:00:14 PM PDT	7/25/12 12:00:14 PM

Views Details / Database: CSTINSDB / Job ID: 13							
#	Type	Key	DEFINER	OWNER	OWNERTYPE	QUALIFIER	
1	Insert	DB2INST1/ MQT207151830160000	DB2INST1	DB2INST1	U	DB2INST1	CREATE SUMMARY TABLE DB2INST1.MQT207151830160000 AS (SELECT SUM(C1) AS 'C1' FROM TABLE SELECT C1 FROM DB2INST1.C1 AS 'C1' WHERE C1 IS NOT NULL GROUP BY C1) WITH NO LOGGING
2	Insert	DB2INST1/ MQT207151830220000	DB2INST1	DB2INST1	U	DB2INST1	CREATE SUMMARY TABLE DB2INST1.MQT207151830220000 AS (SELECT SUM(C1) AS 'C1' FROM TABLE SELECT C1 FROM DB2INST1.C1 AS 'C1' WHERE C1 IS NOT NULL GROUP BY C1) WITH NO LOGGING

Figure 12-12 MQT created and identified by the job

With the changes identified by Optim Configuration Manager in hand, DBAs or system support can create a script or procedure to apply the same changes to other systems to keep all systems synchronized.

12.5 Optim tools runtime client

In this section, we demonstrate how to install and configure the Optim Data Tools Runtime Client that is required for the Optim Configuration Manager to perform advanced data capture and configuration tasks on the client. Without this runtime client, database clients and applications can still be discovered through their interaction with DB2 database instances. However, clients configured using this approach have limited functionality with the Optim family of tools.

The Optim Data Tools Runtime Client is available for a number of operating system types including Microsoft Windows, versions of Linux, and versions of UNIX such as AIX. Obtain the Optim Data Tools Runtime Client installation media from a valid source and use these steps to install the runtime client:

1. Decide which of the following three possible installation processes that you want to use:
 - Installation wizard
Use the installation wizard to install on a computer that has a graphical user interface.
 - Console mode
 - Use the console mode to interactively install on a computer that does not have a graphical user interface.
 - Silent installation
 - A silent installation uses the input from a response file without displaying an interface that requires your input. A silent installation simplifies the process of installing on multiple computers.

In our example we use the installation wizard.

2. Unzip the file that contains the installation image. In our example, we install the runtime client on Windows; this is the file:

`IBM_Data_Tools_Runtime_Client_windows.zip`

3. From the installation directory, find the installation image file for your operating system. This is the wizard installation program for Windows:

`IBM_Data_Tools_Runtime_Client_platform.exe`

4. Within the wizard, select to install **IBM InfoSphere Optim Configuration Manager Client**.

5. Specify the installation directory. We used the following directory:

`c:\Program Files\IBM\IBM_Data_Tools`

6. Optional: When the installation is finished, see the following log file for detailed installation information:

`Clientinstall_details_timestamp.log`

Before using the Optim Data Tools Runtime Client, you have to configure it using the configuration wizard that comes with the client. When you run the configuration wizard, it configures the following items:

- ▶ The URL of the Optim Configuration Manager server.
- ▶ The client cache directory.
- ▶ The classpaths of the JDBC drivers for the applications that will be monitored by the Optim Configuration Manager; for example, Java programs hosted within an IBM WebSphere Application Server that the configuration wizard can discover and configure.

Follow these steps to configure the Optim Data Tool Runtime Client using the tools configuration wizard:

1. When the installation of Optim Data Tools Runtime Client is finished, specify that you want to configure the tool using the configuration wizard and then click **Done**.

You can configure the Optim Data Tools Runtime client later by closing the install wizard, and then at a later time, run the configuration wizard from the installation directory. On Windows, the configuration wizard tool is `cfgtool.bat`.

2. Select the language to use and click **OK**; see Figure 12-13.

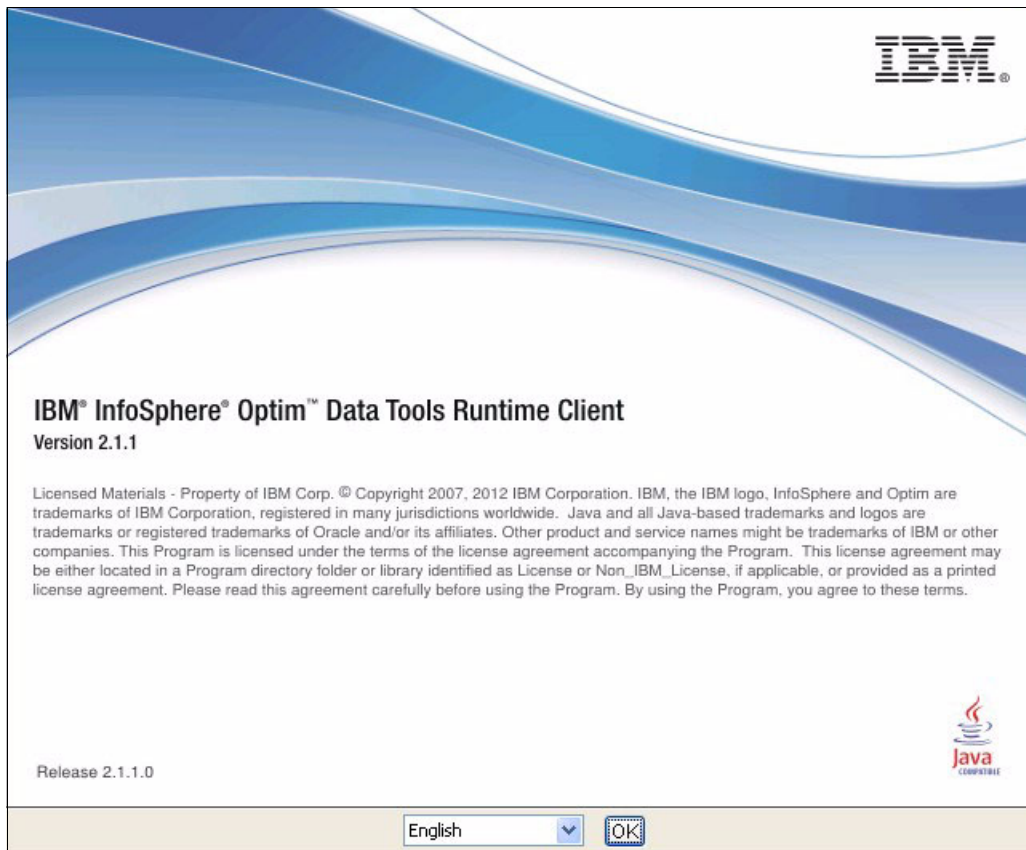


Figure 12-13 Start page of the configuration tool for Optim Data Tools Runtime Client

3. Select the type of configuration that will be required. For our example, it is for Optim Configuration Manager; see Figure 12-13 on page 429.
Then click **Next**.

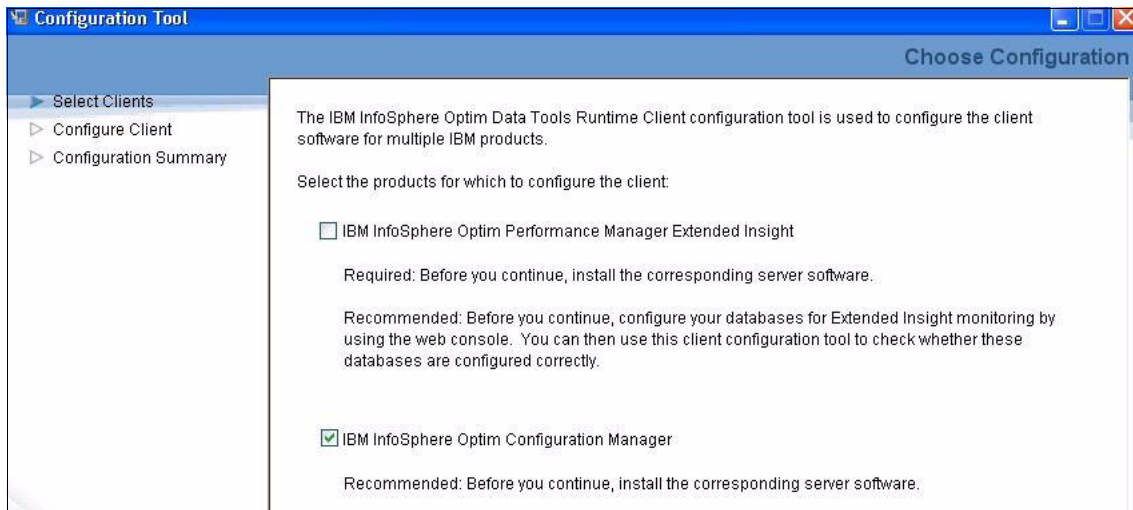


Figure 12-14 Select the type of install within the configuration tool

4. On the next panel you specify which type of application client the driver will be used for; that is, a stand-alone Java application or an application hosted within a WebSphere application server. See Figure 12-15.

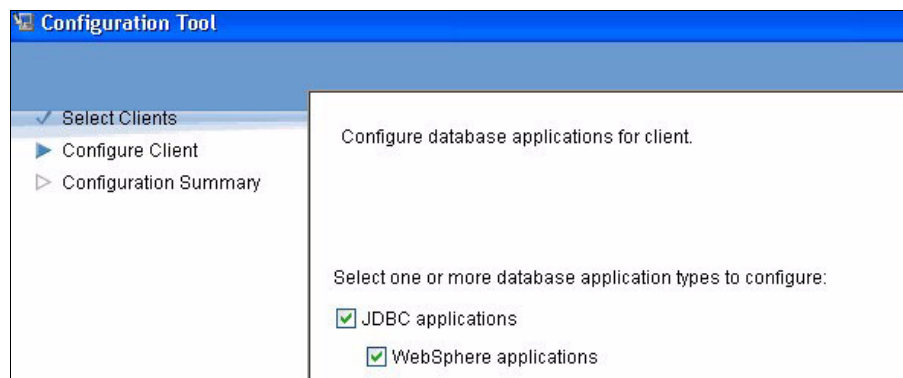


Figure 12-15 Select which type of application the client is for

5. Configure the URL of Optim Configuration Manager in the wizard by using the exact host name and port number that was configured during the installation of InfoSphere Optim Configuration Manager. In our example, we used port

12206, which is the default port issued during the install of Optim Configuration manager as shown in Figure 12-16.

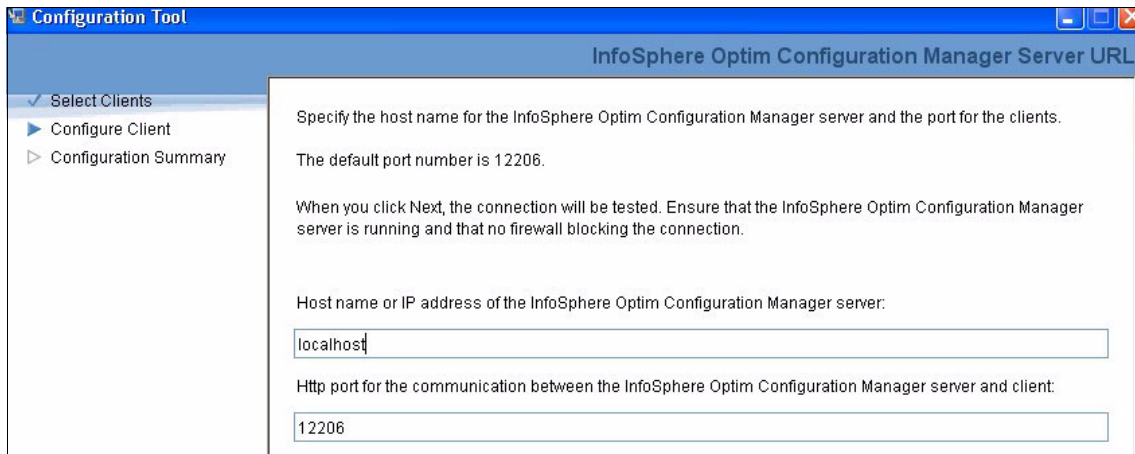


Figure 12-16 Specify both the host name and port of the Optim Configuration Manager

6. Specify the absolute path of the directory in which the client cache will be configured. In our example this is:

`c:\Program Files\IBM\IBM_Data_Tools\pureQuery`

See Figure 12-17.

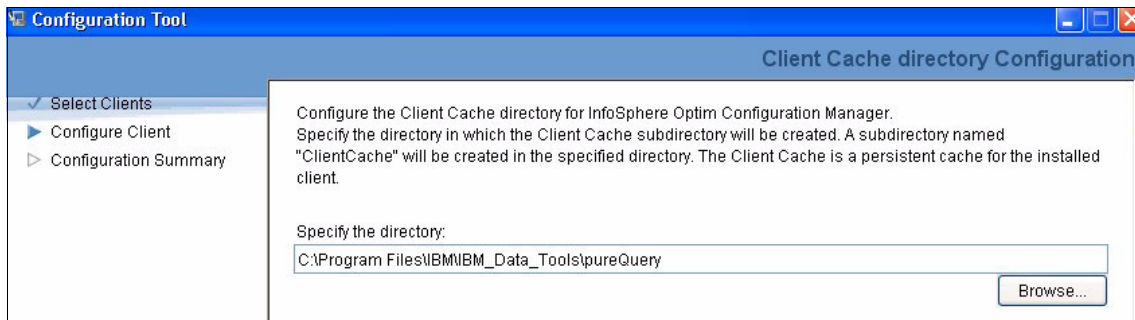


Figure 12-17 Detail the required location of the cache directory

7. Because we selected to configure an application hosted within an instance of IBM WebSphere Application server, the configuration tool searches for the

installation of WebSphere and lists the defined server profiles, as shown in Figure 12-18.

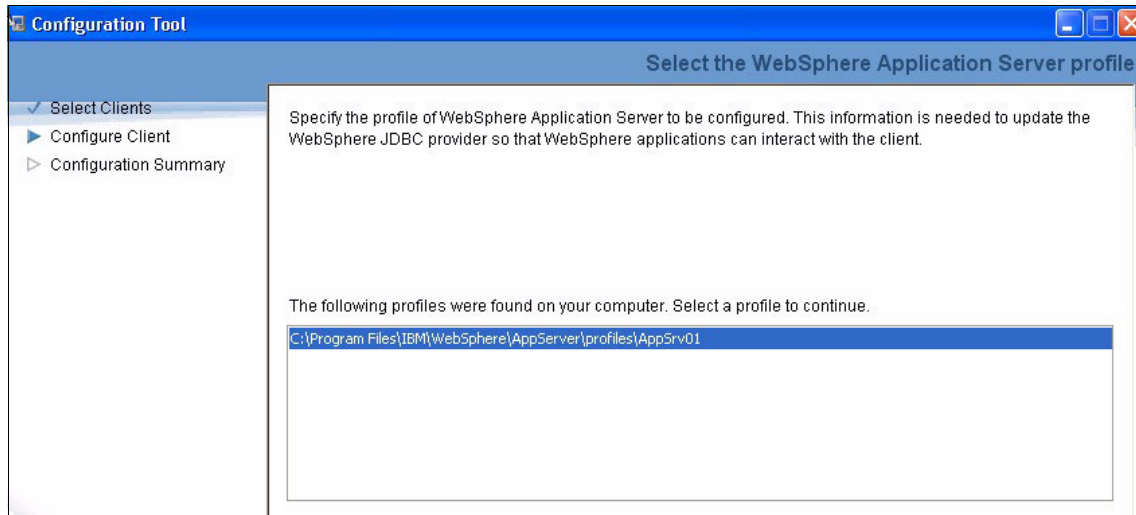


Figure 12-18 Select WebSphere Application Server profile to alter

8. The configuration tool searches this WebSphere Application Server profile for defined JDBC drivers and displays a list. Select a JDBC driver; in our example we chose the DB2 JDBC driver, as shown in Figure 12-19.

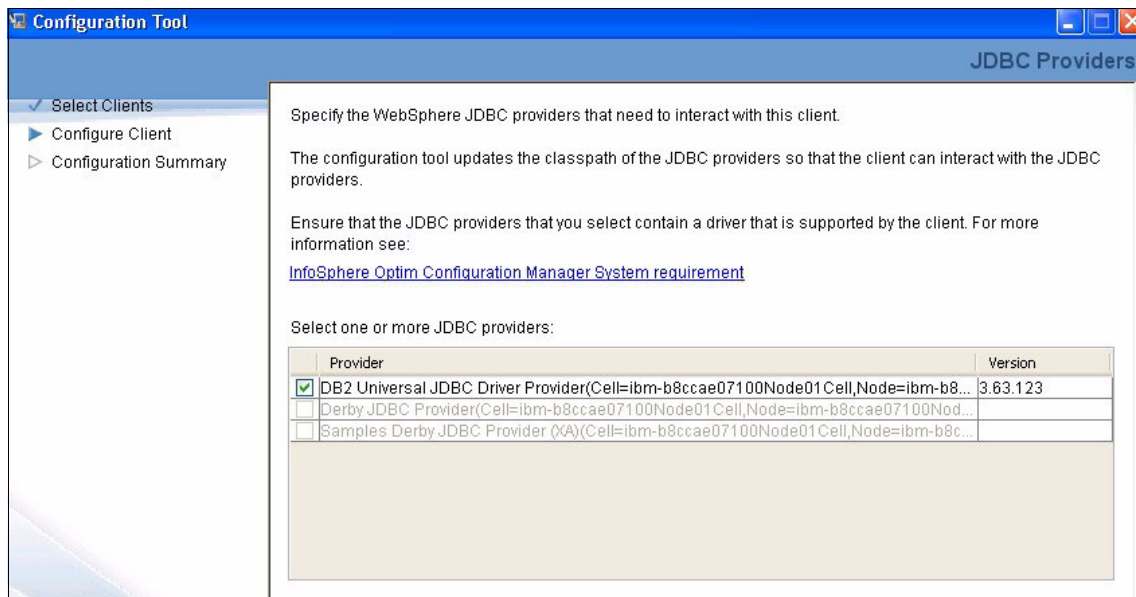


Figure 12-19 Configuration tool lists JDBC drivers installed within WebSphere Application Server profile

9. Confirm that all the input options are valid, and then select **Config** to perform the configuration. Figure 12-20 shows the summary panel.

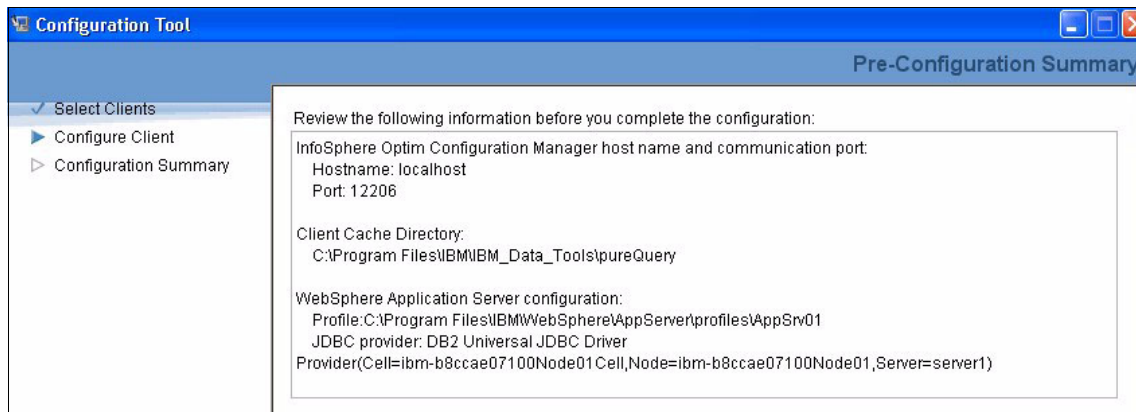


Figure 12-20 Summary page of our example, before performing configuration

10. The Configuration tool confirms that the Optim Data runtime client installation and configuration is complete. Figure 12-21 shows the confirmed panel.

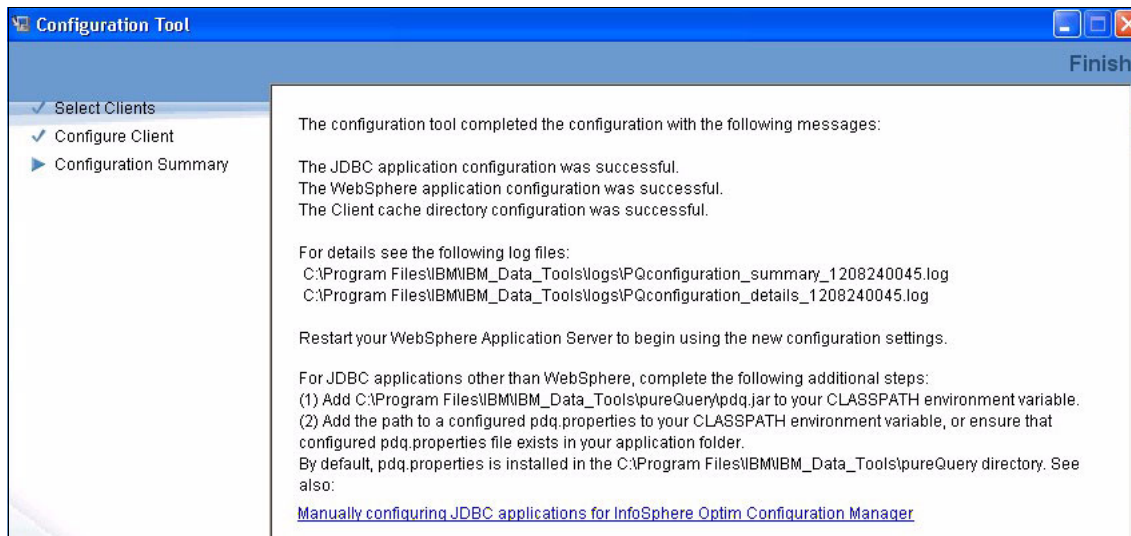


Figure 12-21 Installation confirmation

11. For applications not hosted within a WebSphere Application Server, a manual configuration of the JDBC drivers must be performed. For these stand-alone applications, the DB2 JDBC driver used has to be replaced with the special Optim Data runtime client JDBC driver file. In our example, this is db2jcc4.jar.
12. To confirm that the Optim Data runtime client has been configured and registered on the Optim Configuration Manager, open a web browser, log into

the manager, and then select the **Client** tab. The Optim Data Runtime client within the table of monitored clients is displayed, as shown in Figure 12-22.

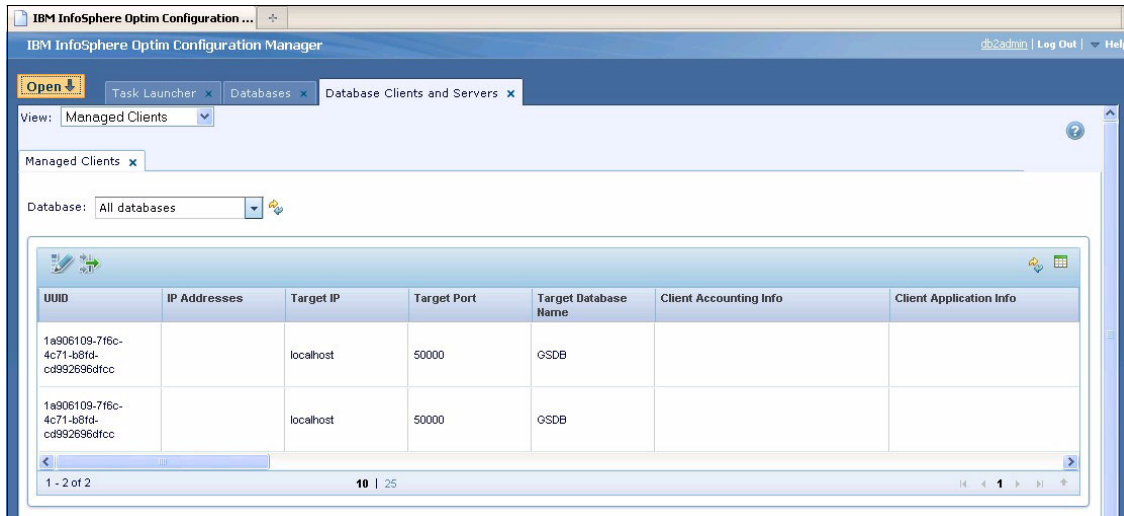


Figure 12-22 The newly registered Optim data runtime client within Optim Configuration Manager

12.6 Optimize database storage with Optim Configuration Manager

Using Optim Configuration Manager, DBAs can optimize database storage both by reducing storage consumption and by implementing a multi-temperature schema.

Optim Configuration Manager provides two mechanisms to help optimize storage:

- ▶ The first mechanism involves making recommendations for where and which type of compression can be applied to database objects to regain space. The tool also estimates how much space will be saved after implementing the recommendations.
- ▶ The second mechanism is where Optim Configuration Manager is used to locate seldom-used objects and trapped storage within a database. *Trapped storage* refers to storage that has been allocated but is not being used. This occurs when system administrators assign extra space to ensure a sufficient storage allocation. This method might lead to wasted space.

By configuring jobs within Optim Configuration Manager to monitor databases, you can detect opportunities to reclaim the trapped storage, apply compression, or drop seldom-used tables and indexes.

You can optimize the use of your storage devices by implementing a multi-temperature storage solution. With a multi-temperature storage solution, the data that you expect to access most frequently is kept on fast storage, while keeping the data that is accessed infrequently on slower and cheaper devices. Optim Configuration Manager can be configured to schedule jobs that will move your data from high performance storage to the slower devices, thereby reducing the need for DBAs to be involved in these administrative duties.

12.6.1 Example Optim Configuration Manager storage-saving job

The example provided in this section demonstrates how to create and schedule a storage monitoring job to find opportunities to reduce storage consumption.

Follow these steps to configure an Optim Configuration Manager job to monitor the storage usage of your warehouse database:

1. Define a connection to the database you want to monitor.

Start the Optim Configuration Manager web interface. Click **Open** → **Setup** → **Databases**. Click **Add** and fill the fields with the database information; see Figure 12-23.



Figure 12-23 Creating a database connection

2. Add a monitoring job.

After the database connection is created, click **Open** → **Setup** → **Job Manager**. Click **Add Job**; see Figure 12-24.

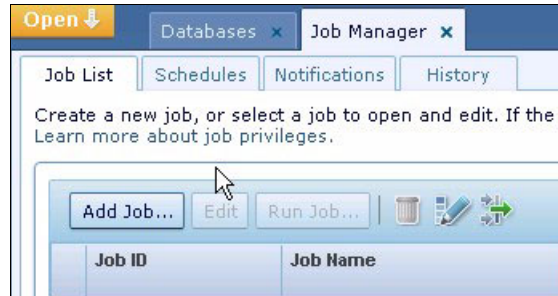


Figure 12-24 Adding a monitoring job

3. Select the job type.

Select **Storage monitoring** as the job type; see Figure 12-25.

Storage monitoring is the Optim Configuration Manager job that looks for opportunities to reclaim trapped storage in tables, identify tables and indexes where compression can be applied, and drop seldom-used databases objects (tables and indexes, for example). Click **OK**.

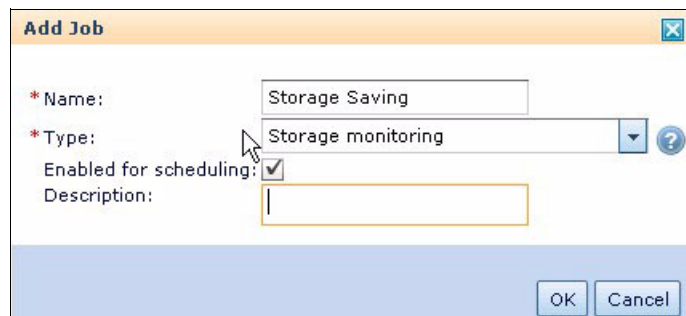


Figure 12-25 Create a Storage monitoring job

4. Select the monitoring tasks.

After clicking OK to add the job, you are redirected to the components configuration of the job automatically. Go to the **Properties** tab.

The Storage monitoring job can perform three tasks:

- Find reclaimable storage -This job looks for storage allocated but not used.

- Find compression opportunities - This job looks for tables without compression enabled.
- Track frequency of use and identify seldom used objects - This job looks for database objects that are seldom used. If these objects are no longer required, the DBA can delete them to save storage.

You can select any or all tasks; see Figure 12-26. We suggest that you select all three tasks for your monitor job because each one has certain characteristics, but are complementary to each other in the search for opportunities to optimize storage.

Select the Storage Tasks to Monitor

Select the monitoring tasks that you want to run. These tasks gather information

- ☒ Find reclaimable storage
- ☒ Find compression opportunities
- ☒ Track frequency of use and identify seldom used objects

Figure 12-26 Selecting tasks to monitor

5. Set the storage monitoring job schedule.

While still in the component configuration, click the **Schedule** tab. The Schedule tab has three subtabs:

- Schedule Details tab - In this tab, you set up the job frequency.
- The Database tab - In this tab, you select the databases on which to run the job. You can select one or multiple databases.
- Timeout Settings tab - In this tab, you specify an action to take if the script execution duration exceeds a timeout period.

Figure 12-27 shows the Schedule Details tab, where you can specify when to run the monitoring job.

Figure 12-27 Schedule for monitoring job

6. Apply your changes.

Click **Apply Changes** to save the schedule and click **Save All** to save your job.

You can click **Run Job** to run the job immediately. Be aware, however, that the job can use a significant amount of resources. Therefore, if you are running any mission-critical application, it can be impacted. Before running the storage monitoring job, run the RUNSTATS command to update the statistics for the tables and indexes you want to monitor.

When the job completes its execution, Optim Configuration Manager generates a report with all the storage savings opportunities it found. To view the report, from Optim Configuration Manager web interface, click **Open** → **Optimize** → **Storage Configuration**; see Figure 12-28 on page 440.

The Storage Configuration report has two parts. The first part presents a summary of all the savings achieved, and the second part contains the details.

The details are organized into three tabs:

- ▶ Reclaim Storage - this shows objects identified as having more space allocated than needed.
- ▶ Compression - this shows tables where compression can be enabled and how much storage can be gained.
- ▶ Seldom used objects - this shows database objects that have not been used for a while.

Database	Reclaim Savings (MB)	Static Compression Savings (MB)	Adaptive Compression Savings (MB)	Compression Savings Achieved (MB)	Last us
CSTINSDB	2.42186	8.01532	8.89033	29.19033	Thursd

1 - 1 of 1

10 | 25

Database: CSTINSDB

Reclaim storage

Compression

Seldom used objects

Object Type	Object Schema	Object Name	Parent Schema	Parent Name	Size (MB)	Estimated Savings (MB)
TABLE	CSTINSIGHT	CUSTOMER_TXN_RP			3.89062	1.20703 (31.02%)
TABLE	CSTINSIGHT	CUSTOMER_TXN			3.6875	1.07031 (29.02%)
TABLE	CSTINSIGHT	CALENDAR_DATE			0.14453	0.05859 (40.54%)
TABLE	OPM	HISTOGRAMBIN_OPMV			0.11718	0.05859 (50%)
TABLE	CSTINSIGHT	GEO_AREA			0.14062	0.02734 (19.44%)
TABLE	CSTINSIGHT	PRODUCT			< 0.1	0 (0%)
TABLE	CSTINSIGHT	CUSTOMER			< 0.1	0 (0%)
TABLE	CSTINSIGHT	PERSON			< 0.1	0 (0%)
TABLE	CSTINSIGHT	AGREEMENT			< 0.1	0 (0%)

Figure 12-28 Storage optimization opportunities

In our example we created one job with the three tasks (reclaim storage, compression opportunities, and seldom-used objects). You also can create three jobs, having each one perform one of the tasks.

12.6.2 Automate multi-temperature storage migration with Optim Configuration Manager

DB2 10.1 introduced the storage group concept. Each storage group represents a tier of storage in the data warehouse. You can logically group paths on storage system with similar attributes by using the storage group feature. By associating the automatic storage table spaces with a storage group, DBAs can place data that is accessed more frequently into faster storage. Data that is accessed less often is still available on slower storage.

Implementing storage groups can include the following benefits:

- ▶ Storing data based on the priority of accessibility
- ▶ Reducing the total cost of ownership
- ▶ Improving performance for target workloads
- ▶ Meeting business criteria for performance and reliability requirements

You can define the storage groups for the various classes of storage devices by using IBM Data Studio, Design Studio, or by issuing DDL statements. We discuss the storage group feature in detail in Chapter 9, “Managing data lifecycle with InfoSphere Warehouse” on page 335.

In this section, we demonstrate how to use Optim Configuration Manager to automate the migration process.

Figure 12-29 shows an example storage groups view from IBM Data Studio.

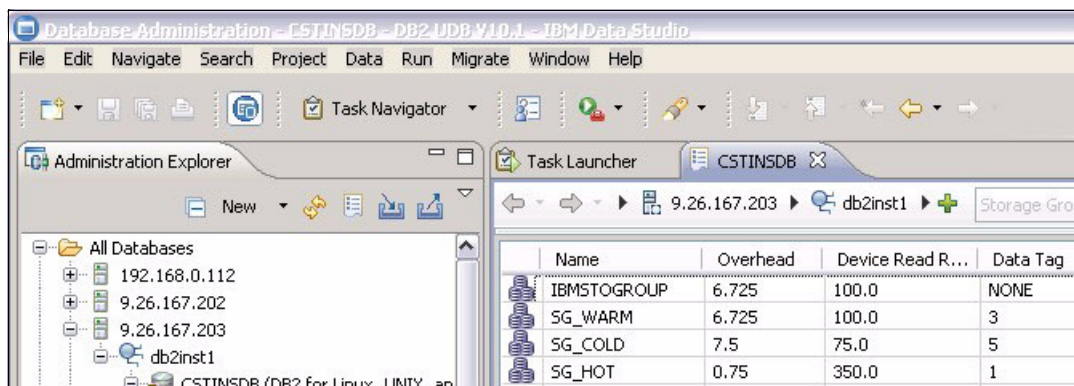


Figure 12-29 Definition of storage groups

To automate storage migration, you define and schedule a job in Optim Configuration Manager by following these steps:

1. Log on to the Optim Configuration Manager web interface.

2. Add a data migration job.

Click **Open** → **Setup** → **Job Manager**. Click **Add** and give a name to the job. Select **Data migration** as the job type; see Figure 13-12. Click **OK**.

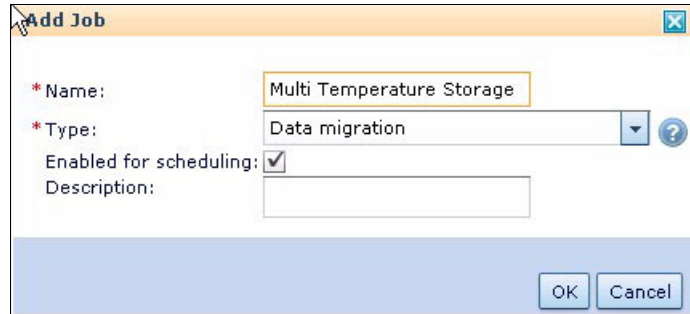
A screenshot of the 'Add Job' dialog box. It has a title bar with 'Add Job' and a close button. The dialog contains four fields: 'Name' with the text 'Multi Temperature Storage', 'Type' with a dropdown menu showing 'Data migration' and a help icon, 'Enabled for scheduling' with a checked checkbox, and 'Description' with an empty text box. At the bottom right are 'OK' and 'Cancel' buttons.

Figure 12-30 Add a data migration job

3. Select the database or tables to monitor.

In the components configuration panel, select the **Data Migration** tab to define the properties of the migration job.

The Data Migration tab has two other tabs:

- Database and Tables - this tab enables you to define an entire database or tables that you want to monitor, as shown in Figure 12-31.

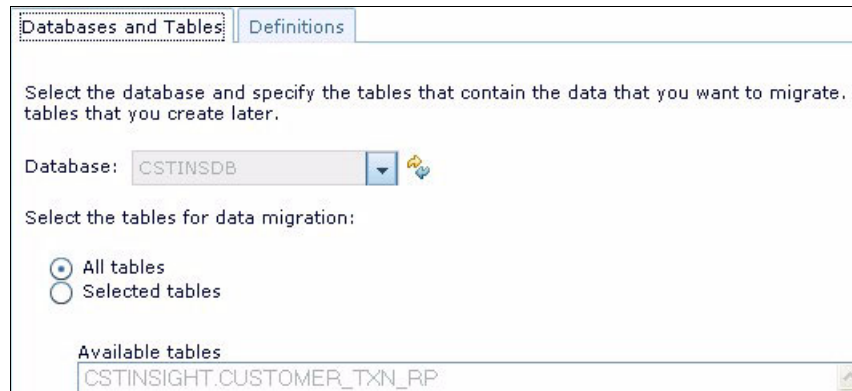
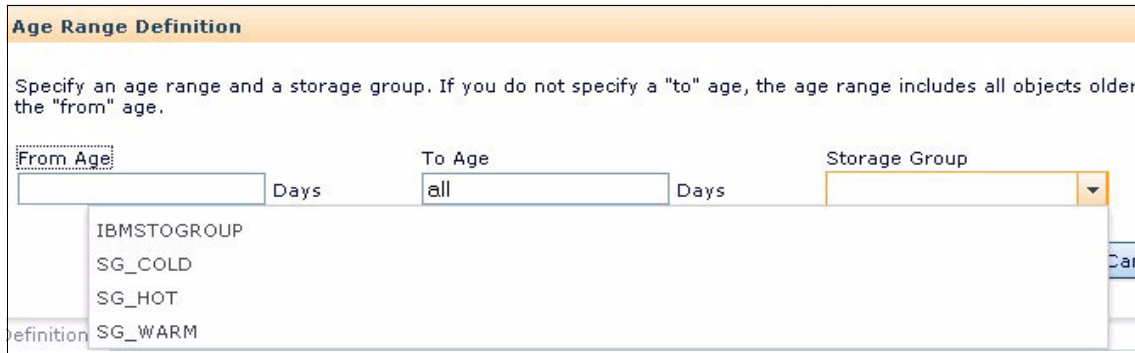
A screenshot of the 'Database and Tables' configuration panel. It has two tabs: 'Databases and Tables' (selected) and 'Definitions'. The main area contains instructions: 'Select the database and specify the tables that contain the data that you want to migrate. Select the tables that you create later.' Below this is a 'Database:' dropdown menu with 'CSTINSDB' selected and a help icon. Then, it says 'Select the tables for data migration:' followed by two radio buttons: 'All tables' (selected) and 'Selected tables'. At the bottom, there is a section titled 'Available tables' with a list box containing 'CSTINSIGHT.CUSTOMER_TXN_RP' and a small icon on the right.

Figure 12-31 Select database or tables to migrate

- Definition tab - this table enables you to define the range of your storage groups. Also, you specify the storage group to which you want to move the table spaces that contain data within the age range; see Figure 12-32 on page 443.



Age Range Definition

Specify an age range and a storage group. If you do not specify a "to" age, the age range includes all objects older than the "from" age.

From Age: Days To Age: Days Storage Group:

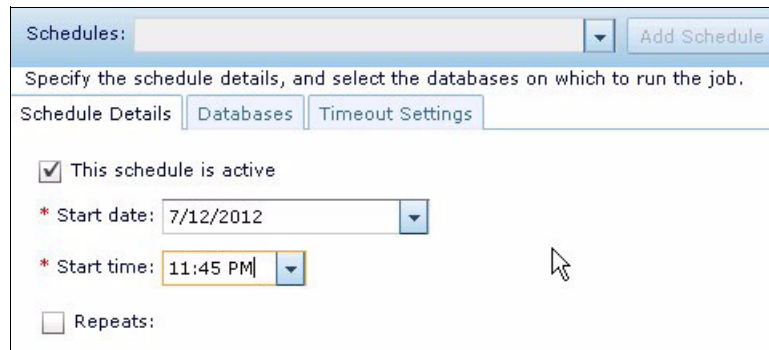
Storage Group dropdown menu:

- IBMSTOGROUP
- SG_COLD
- SG_HOT
- SG_WARM

Figure 12-32 Define the range for storage group

4. Set the data migration job schedule.

In the component configuration, select the **Schedule** tab. This tab enables you to define the job frequency, the databases on which to run the job, and action to take if the script execution duration exceeds a timeout period. Figure 12-33 shows the Schedule tab.



Schedules: Add Schedule

Specify the schedule details, and select the databases on which to run the job.

Schedule Details | Databases | Timeout Settings

☒ This schedule is active

* Start date:

* Start time:

☐ Repeats:

Figure 12-33 Schedule for data migration job

Be aware that a data migration job can take hours to finish. Therefore, select a day and time to perform the migration when you do not have mission-critical applications or users connected.

5. Apply, save, and run the job.

After running the job, you can check the job execution log to see whether there is any data migration activity. While still in the Job Manager panel, select the **History** tab. You can view the status detail (indicating whether the job failed, succeeded, or is still running), or select a job and open the log file to see detailed information.

In our example, after we selected our data migration log and clicked **View log** in the web browser, we saw that a table space aged and was moved to the Cold storage group. See Figure 12-34.

ID:	1342026509864
Name:	Multi Temperature Storage
Start Time:	2012-08-06 02:25:11
End Time:	2012-08-06 02:25:13
Output from executing:	<pre>Time Unit: DAYS Rule: FromAge: 0, ToAge: 30, MoveToStorageGroup: SG_HOT Alter statements: - No Qualified Table Spaces Rule: FromAge: 30, ToAge: 60, MoveToStorageGroup: SG_WARM Alter statements: - No Qualified Table Spaces Rule: FromAge: 60, ToAge: Unlimited, MoveToStorageGroup: SG_COLD Alter statements: - ALTER TABLESPACE "TSFACTDATA_2011Q1" USING STOGROUP "SG_COL Submission status of Alter statements: ALTER TABLESPACE "TSFACTDATA_2011Q1" USING STOGROUP "SG_COLD" ;</pre>
Result:	The job executed successfully. Execution Status code: 2

Figure 12-34 Optim Configuration Manager data migration job log

With the Optim Configuration Manager data migration job, DBAs automate a recurrent and time-consuming task. The job can also send notifications and can be configured to run before or after other jobs.

12.7 InfoSphere Warehouse high availability with Q replication and Optim Configuration Manager

Optim Configuration Manager can be used to redirect client applications from one system to another. When combined with a replication solution such as Q replication, it helps improve reliability, fault-tolerance, performance, and accessibility in a simplified manner.

Q Replication is a solution that uses WebSphere MQ message queues to transmit transactions between source and target databases. This type of replication offers several benefits:

- ▶ Minimum latency - Changes are sent as soon as they are committed at the source and read from the log.
- ▶ High volume throughput - WebSphere Q Capture and Q Apply programs use a multithreaded architecture to cope with rapid changes at the source.
- ▶ Minimum network traffic - Messages are sent by using a compact format and only necessary information is sent through the network.
- ▶ Asynchronous communication - In case of a system failure, messages remain in the queue and are processed when the system is able to handle new requests again. This is based on the fact that both Q Apply and Q Capture programs can operate independently in an asynchronous mode.

With Q Replication, the target or standby database is available for read and write access. This means that you can offload processes to the standby database, thus relieving the production environments.

Q Replication allows many different configurations. You can replicate between remote servers or within a single server. You can replicate changes in a single direction or in multiple directions. Replicating in multiple directions can be bidirectional (useful for managing standby or backup systems) or peer-to-peer (useful for synchronizing data on production systems).

Q Replication supports three types of replication:

- ▶ Unidirectional replication
- ▶ Bidirectional replication
- ▶ Peer-to-peer replication

In this section, we demonstrate how to set up a unidirectional replication of a database by using three products:

- ▶ WebSphere MQ - to create the queues
- ▶ IBM Replication Center - to create the replication rules
- ▶ Optim Configuration Manager - to redirect client applications

All of these products are shipped with InfoSphere Warehouse Advanced Edition.

Create queues with WebSphere MQ

Follow these steps to create WebSphere MQ objects:

1. Open WebSphere MQ.

Figure 12-35 shows WebSphere MQ initial panel.



Figure 12-35 WebSphere MQ initial panel

2. Create queue managers.

We have to create queue managers for both source and the target databases. Right-click Queue Managers and select **New** → **Queue Manager**.

Figure 12-36 shows two queue managers are created, SRC-QM for the source database and TGT_QM for the target database.



Figure 12-36 Queue managers created

Next we create queues for both the source and target databases.

3. Create the source queues.

Start the source queue manager by right-clicking **SRC_QM** and select **Start**. To add the source queues, right-click the source queue manager SRC_QM to display the option menu and then select **New**.

We create the following queues for the source queue manager:

- ASN.RESTARTQ - This is a local queue that holds a single message that tells the Q Capture program where to start reading in the DB2 recovery log after a restart.

Each Q Capture program must have its own restart queue.

- ASN.ADMINQ - The administration queue is a local queue that receives control messages from a Q Apply program or a user application to the Q Capture program.

Each administration queue is to be read by only one Q Capture program.

- ASN.SAMPLE_TO_TGT_SAMP.DATA - This is a remote send queue that directs data messages from a Q Capture program to a Q Apply program or a user application. In remote configurations, this is the local definition on the source system of the receive queue on the target system.

Each send queue is to be used by only one Q Capture program.

- TGT_QM: This is the transmission queue. This queue is local and it holds messages that are waiting to go across a channel. This queue can be named as the destination queue manager, which is a reminder about where its messages go.

Figure 13 - 18 shows the source queues that were created.

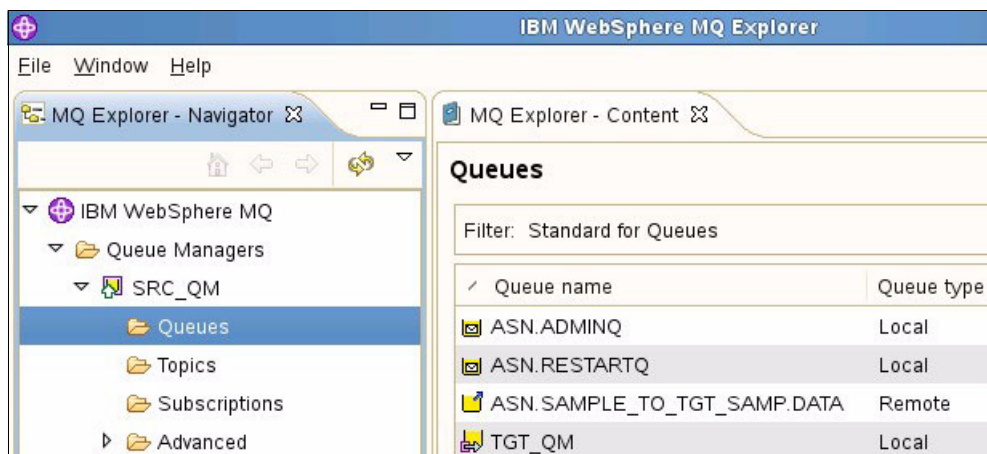


Figure 12-37 Source queues

4. Create the source channels.

We create two channels for the source queue manager:

- SRC_QM_TO_TGT_QM - The sender channel act as a communication tunnel between the source to the target. It uses the TGT_QM transmission queue.
- TGT_QM_TO_SRC_QM - The receiver channel act as a communication tunnel between the target to the source

Figure 12-38 shows the source channels that were created.

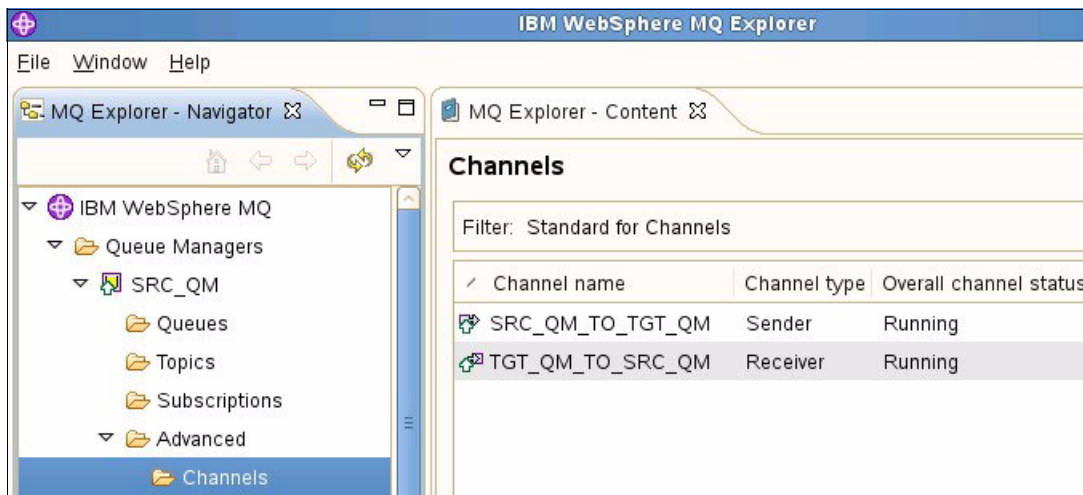


Figure 12-38 Source channels

5. Create the source TCP listener.

A TCP listener for the source queue manager is the listener that forwards messages to the queues using a specific port. We create REPL_LSTR as the source TCP listener; see Figure 12-39.

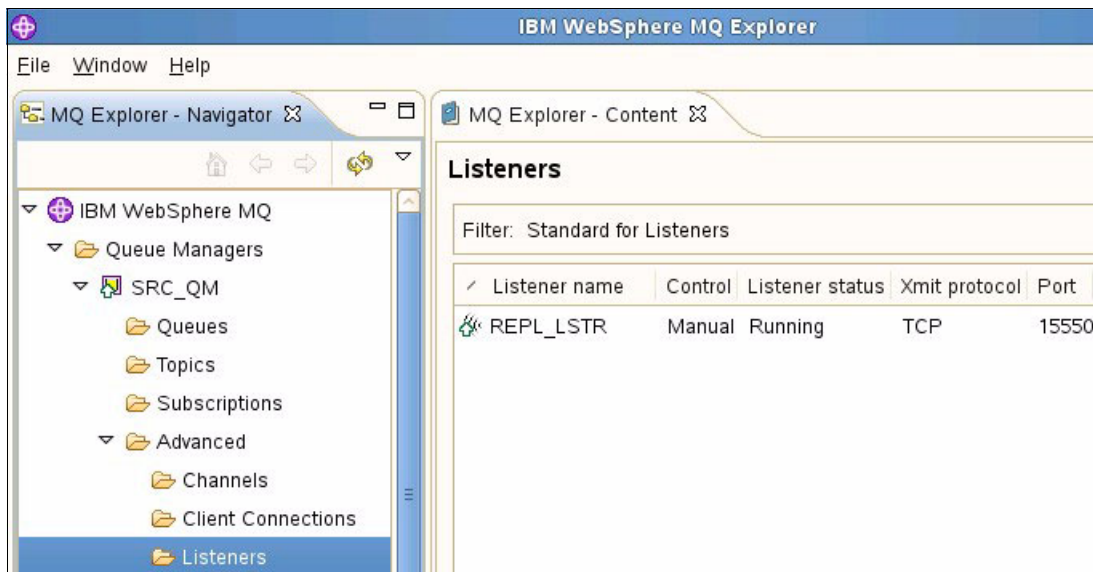


Figure 12-39 Source listener

6. Create the target queues, channel, and TCP listener.

We repeat the same steps to create the objects for the target queue manager.

The target queues are:

- ASN.SAMPLE_TO_TGT_SAMP.DATA - This is the receive queue for the target system.
- ASN.ADMINQ - This is used by the Q Apply program (on the target) to access the Q Capture administration queue (on the source).
- SRC_QM - This is the local transmission queue.
- IBMQREP.SPILL.MODELQ - This is the model queue that is defined on the target system to hold transaction messages from a Q Capture program while a target table is being loaded. The Q Apply program creates spill queues dynamically during the loading process based on the model queue definition, and then deletes them.

Figure 12-40 shows the target queues created under TGT_QM.

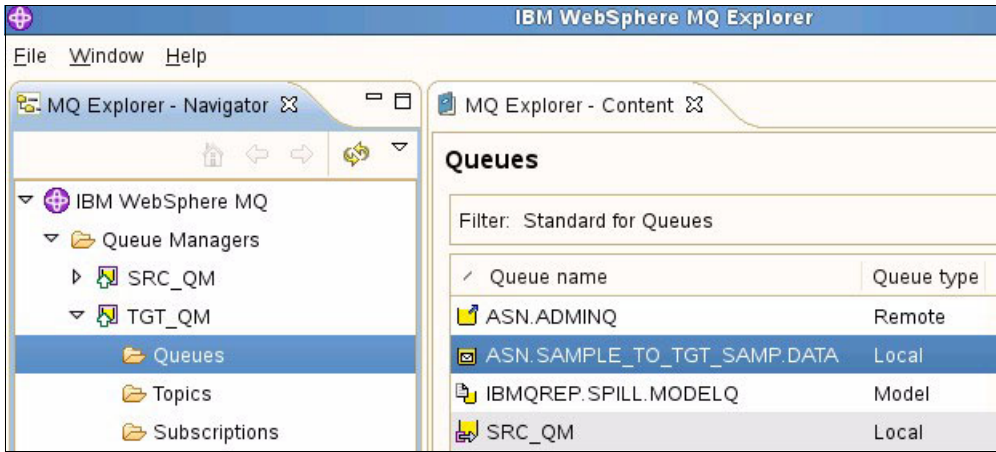


Figure 12-40 Target queues

The two channels for the target queue manager are:

- TGT_QM_TO_SRC_QM - The sender channel acts as a communication tunnel between the target to the source. It uses the SRC_QM transmission queue.
- SRC_QM_TO_TGT_QM - The receiver channel acts as a communication tunnel between the source to the target.

Figure 12-41 shows the target channels that were created.

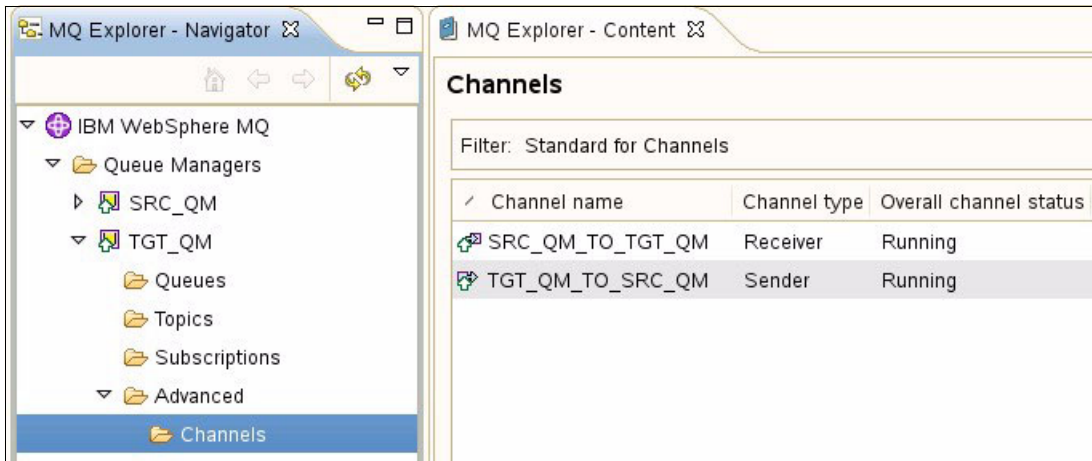


Figure 12-41 Target channels

The TCP listener created for the target queue manager is REPL_LSTR, as shown in Figure 12-42.



Figure 12-42 Target listener

After creating all the WebSphere MQ objects, make sure the channels and the listeners are running. Right-click each of them and select **Start**.

Create replication rules using IBM Replication Center

To create replication rules, you first must create the Q Replication control tables for both the source and the target databases. You then define the replication rules.

Creating source and target control tables

We create the control tables for source and target databases using the IBM Replication Center as follows:

1. Start IBM Replication Center.

To start IBM Replication Center, run the **db2rc** command from a Linux system. In a Windows machine, go to **Start** → **All Programs** → **IBM DB2** → **DB2 Copy Name** → **Replication Center**, where DB2 Copy Name indicates the DB2 copy name specified during the installation.

Figure 12-43 shows the IBM Replication Center.

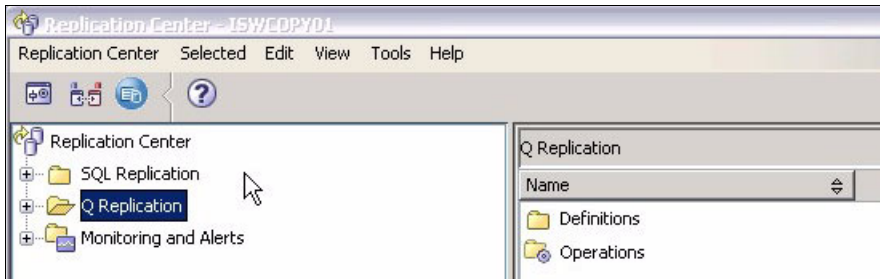


Figure 12-43 IBM Replication Center

2. Create the database objects for Q Replication source database.
- Expand **Q Replication** and then expand **Definitions**. Right-click **Q Capture Servers** and select **Create Q Capture Control Tables**. See Figure 13 - 25.

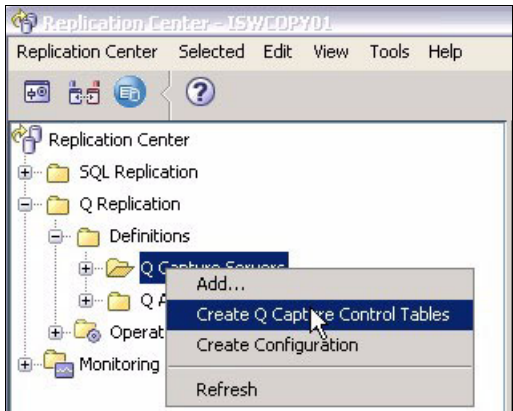


Figure 12-44 Create Q Replication objects for the source

The wizard guides you through the process to create Q Replication objects. For the control tables, we used the Replication Center recommendations, as shown in Figure 12-45. You can also define your own settings.



Figure 12-45 Q Replication source start configuration

3. Select the source database.

In the Specify a Q Capture server panel, select the source database where the control tables will be created. Our source database is CSTINDB, as shown in Figure 12-46.

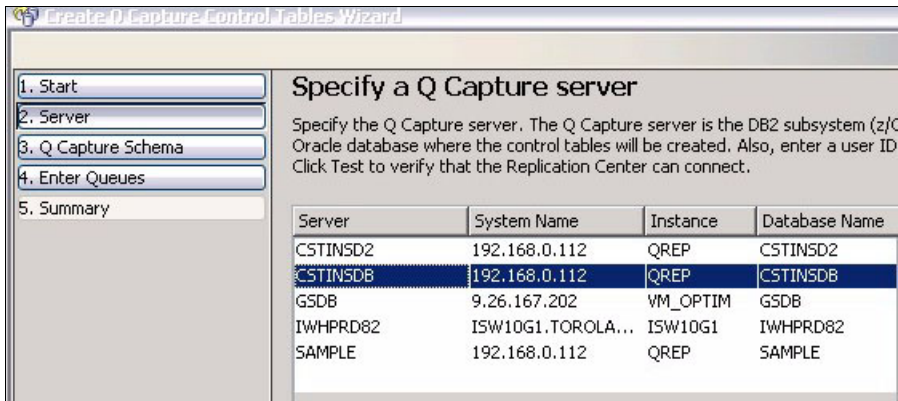


Figure 12-46 Q Replication source server configuration

4. Define a schema.

Define the database schema that will hold the control table; see Figure 12-47.

The screenshot shows the 'Create Q Capture Control Tables Wizard' dialog box. On the left is a vertical list of steps: 1. Start, 2. Server, 3. Q Capture Schema (highlighted), 4. Enter Queues, and 5. Summary. The main area is titled 'Specify a Q Capture schema'. It contains the text: 'Specify a schema to identify the Q Capture control tables that will be created on this server and that server. z/OS: Specify the DB2 for z/OS subsystem and database where the control tables will be created.' Below this is a text field labeled 'Q Capture schema' with the value 'ASN' entered. At the bottom, there is an unchecked checkbox labeled 'Create both Q Capture and Q Apply control tables on this server' and a link that says 'When do you want to choose this option?'.

Figure 12-47 Q Replication source schema configuration

5. Select the WebSphere MQ objects.

We select the queue manager SRC_QM that we created in the beginning of this section; see Figure 12-48.

The screenshot shows the 'Create Q Capture Control Tables Wizard' dialog box at Step 4, 'Enter Queues'. The left sidebar shows steps 1 through 5, with '4. Enter Queues' highlighted. The main area is titled 'WebSphere MQ objects' and contains the text: 'Specify the name of a WebSphere MQ queue manager on the system where the Q Capture program manager handles queues and messages for the Q Capture program.' There are three input fields, each with a dropdown arrow: 'Queue manager' with 'SRC_QM' selected, 'Administration queue' with 'ASN.ADMINQ' selected, and 'Restart queue' with 'ASN.RESTARTQ' selected. At the bottom, there is an information icon and a tip: 'Tip: You can view and select WebSphere MQ objects. [More details](#)'.

Figure 12-48 Q Replication source queues configuration

6. Review the summary.

Review the summary of the source control tables settings and make changes if required; see Figure 12-49.

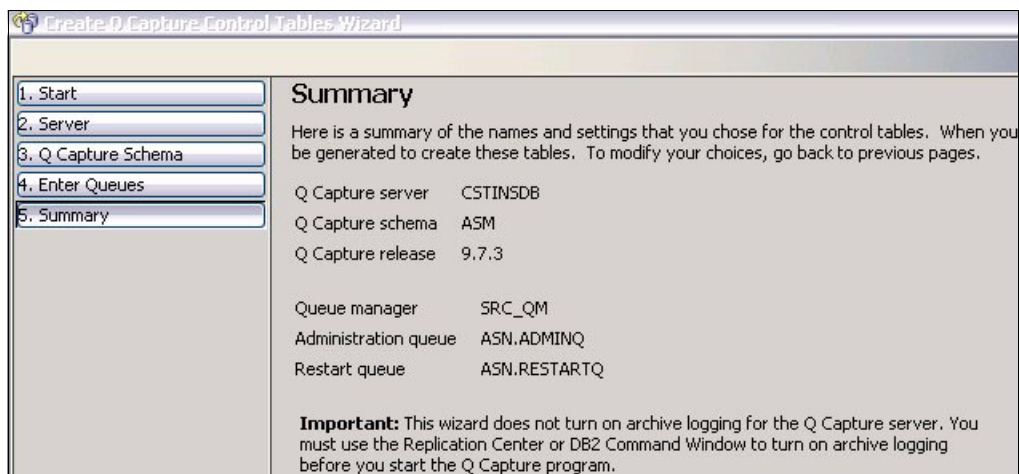


Figure 12-49 Q Replication source summary page

7. Run SQL.

The Q Replication wizard generate SQL statements for creating the source control tables. You can run the SQL statements now or save the file to run it later. We select **Run now** to create the control tables; see Figure 12-50.

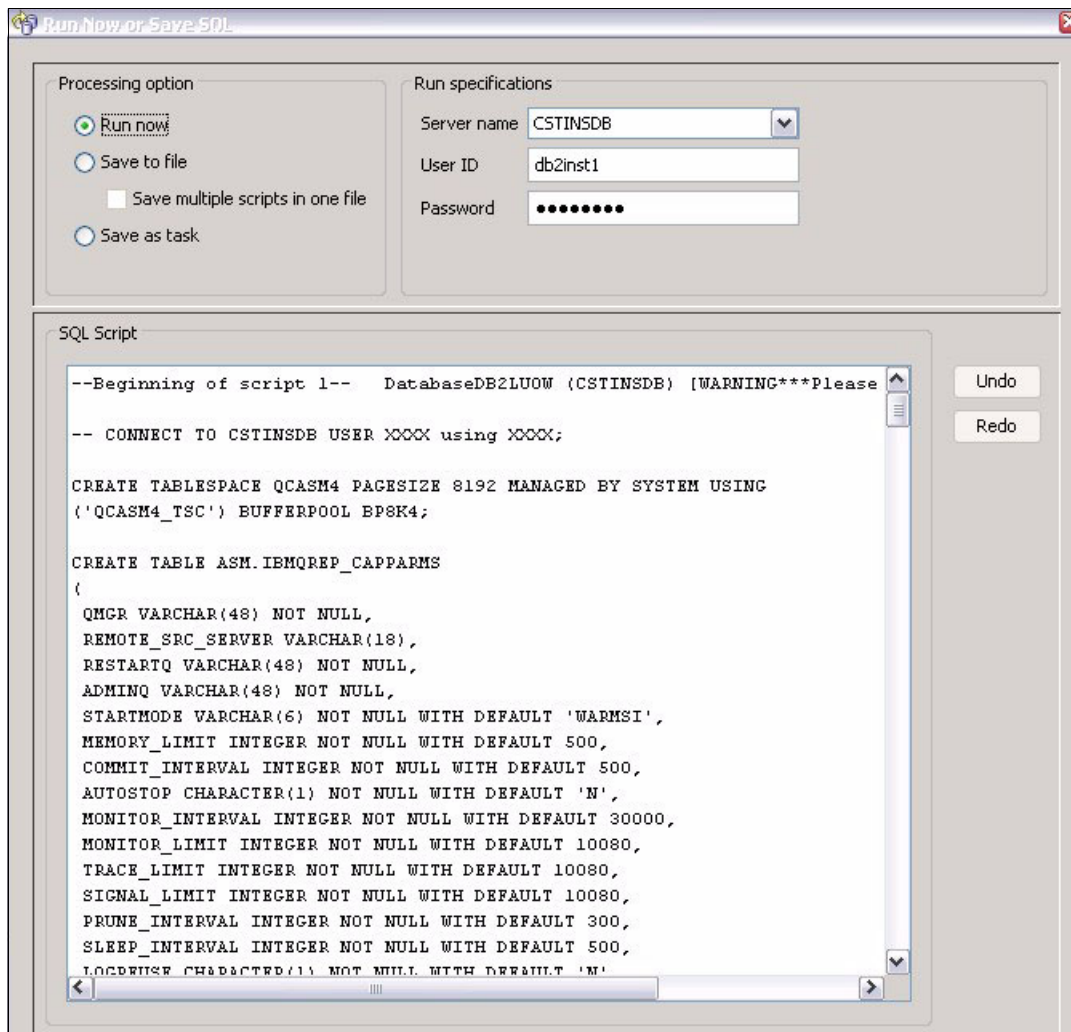


Figure 12-50 Q Replication source SQL

The source database that we configured is now shown under the Q Capture Servers directory; see Figure 12-51.

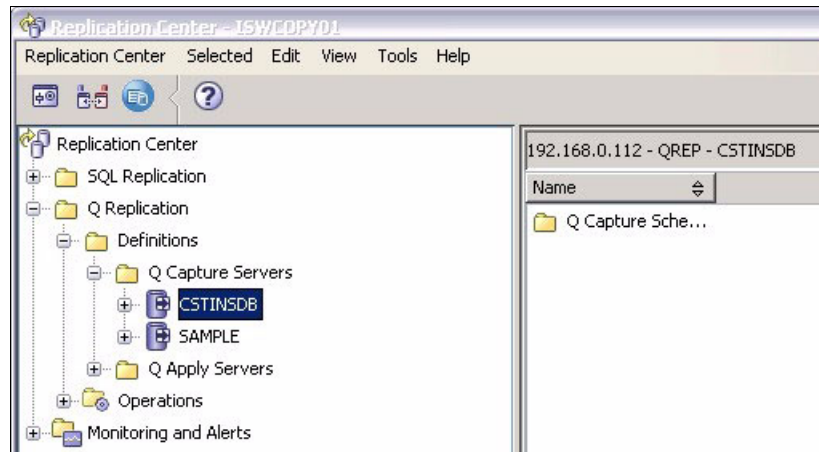


Figure 12-51 Source database

8. Create the database objects for the Q Replication target database.

Expand **Q Replication** and then expand **Definitions**. Right-click **Q Apply Servers** and select **Create Q Apply Control Tables**; see Figure 12-52.



Figure 12-52 Create Q Replication objects for the target

9. The Q Replication target starts configuration.

The wizard guide you through the process to create the control tables for the target database. The steps are similar to those for creating the source control

tables. Again, we use the Replication Center recommendations to define the control tables; see Figure 13-34.



Figure 12-53 Q Replication target start configuration

For the Q Apply schema, we use ASN1. The WebSphere MQ target queues manager is TGT_QM.

We run the SQL statements immediately to create the control tables for the target database. Now we have both source and target control tables created.

Creating a replication queue map

Define a replication queue map to specify which WebSphere MQ queues are going to send the data from the source database server to the target database server. Each replication queue map identifies the following WebSphere MQ queues and parameters:

- ▶ Send queue - This is the WebSphere MQ queue where the Q Capture program sends source transactions and informational messages. When you define a replication queue map, you must select a send queue that is configured to transport compact messages.
- ▶ Receive queue - This is the WebSphere MQ queue from which the Q Apply program receives source transactions and informational messages.
- ▶ Administration queue - This queue is used by the Q Apply program to send control messages to the Q Capture program. The messages that the Q Apply program sends on this queue have several purposes, including telling a Q Capture program to start sending its messages, initiating a full refresh of a target table, or telling the Q Capture program to stop replicating certain Q subscriptions.

To define the replication queue maps, follow these steps:

1. Within the Q Replication Center, expand the Q Replication directory until you see Replication Queue Maps. Right-click **Replication Queue Maps** and select **Create**; see Figure 12-54.

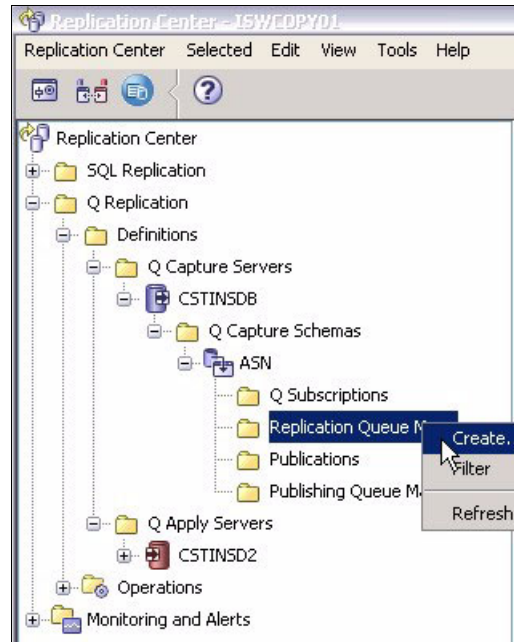


Figure 12-54 Add a Q Replication queue map

2. In the next panel, specify the properties for the new replication queue map that include the source and target databases, the queue information, and the replication queue map name; see Figure 12-55 on page 460.

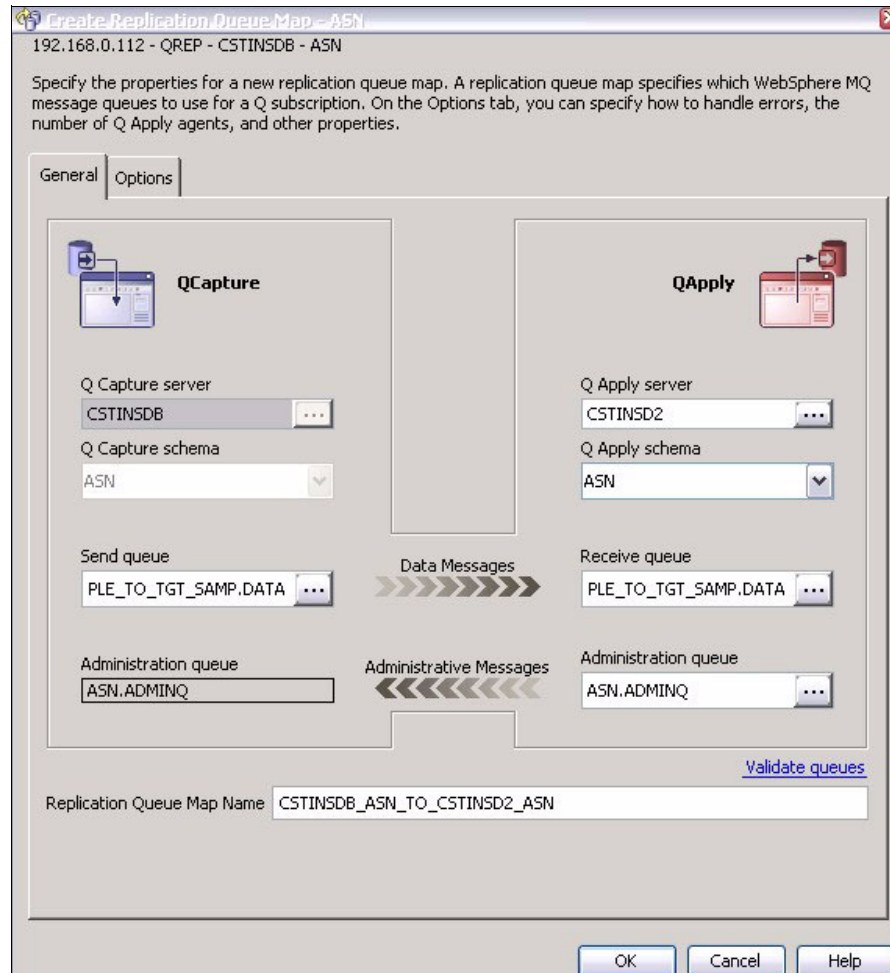


Figure 12-55 Configuration of Q Replication queue map

3. Validate.

The wizard tests the queue definitions as shown in Figure 12-56.



Figure 12-56 Validation of Q Replication queue map

4. Create a Q subscription.

A Q subscription is used to define how data from a single source table is replicated to a single target table or is passed to parameters in a stored procedure for data manipulation.

The Q subscription also tells the replication programs which changes to capture from the source table, which queues to use for sending and receiving change messages, and how to process the messages.

Right-click **Q Subscription** under the Q Capture Schemas and select **Create**; see Figure 12-57.

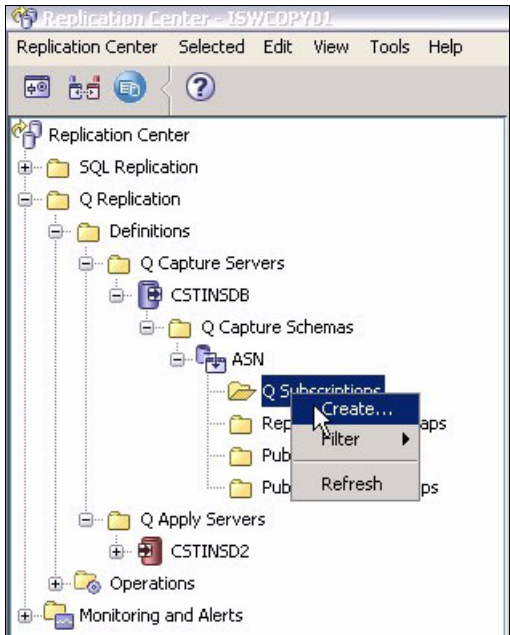


Figure 12-57 Creating a Q Replication subscription

Figure 12-58 shows the initial panel of a Q subscription setup.

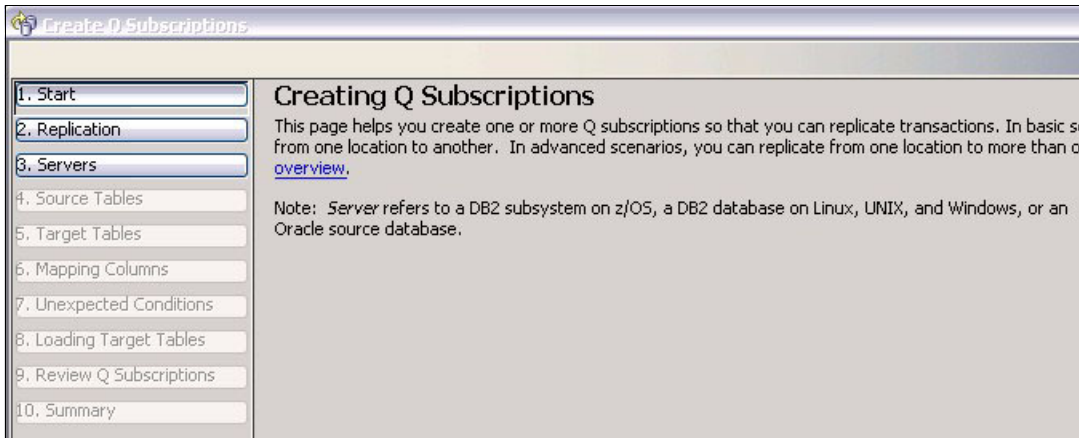


Figure 12-58 Q Replication subscription start configuration

5. Select the replication type.

In the “Which type of replication panel” select the type of replication. In this example, we select **Unidirectional**; see Figure 12-59.

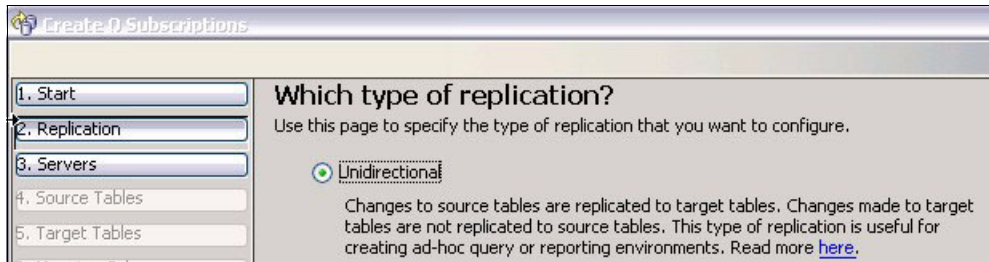


Figure 12-59 Selecting replication type

6. Specify target and source servers.

Specify source and target servers and the replication queue map that Q Capture and Q Apply use to communicate; see Figure 12-60.

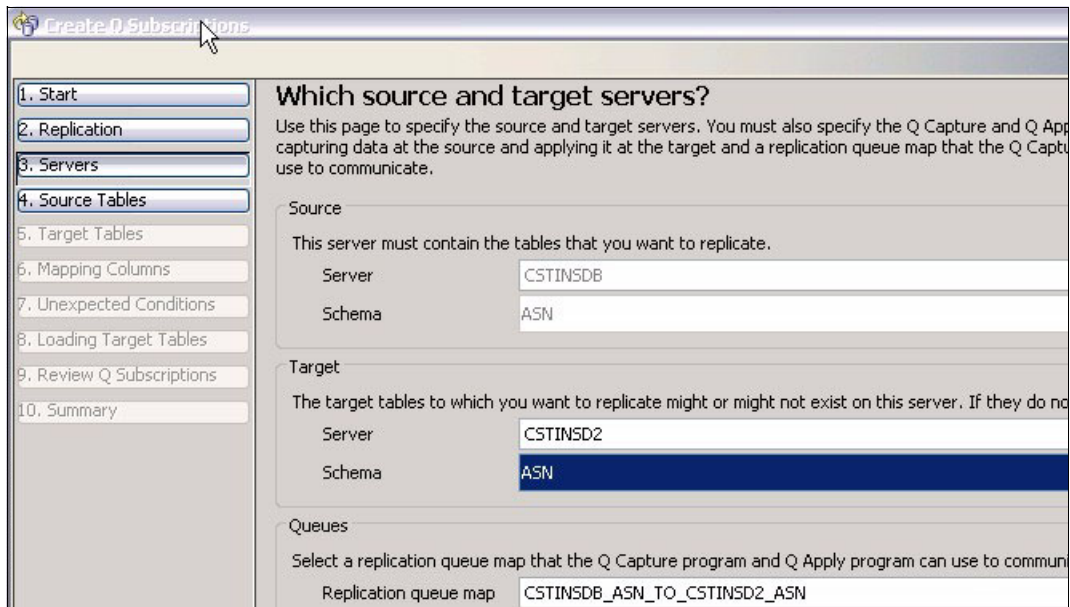


Figure 12-60 Q Replication source and target system

7. Select tables for replication.

You can filter the database to have a list of candidate tables, and then add the table to be replicated to the subscription; see Figure 12-61.

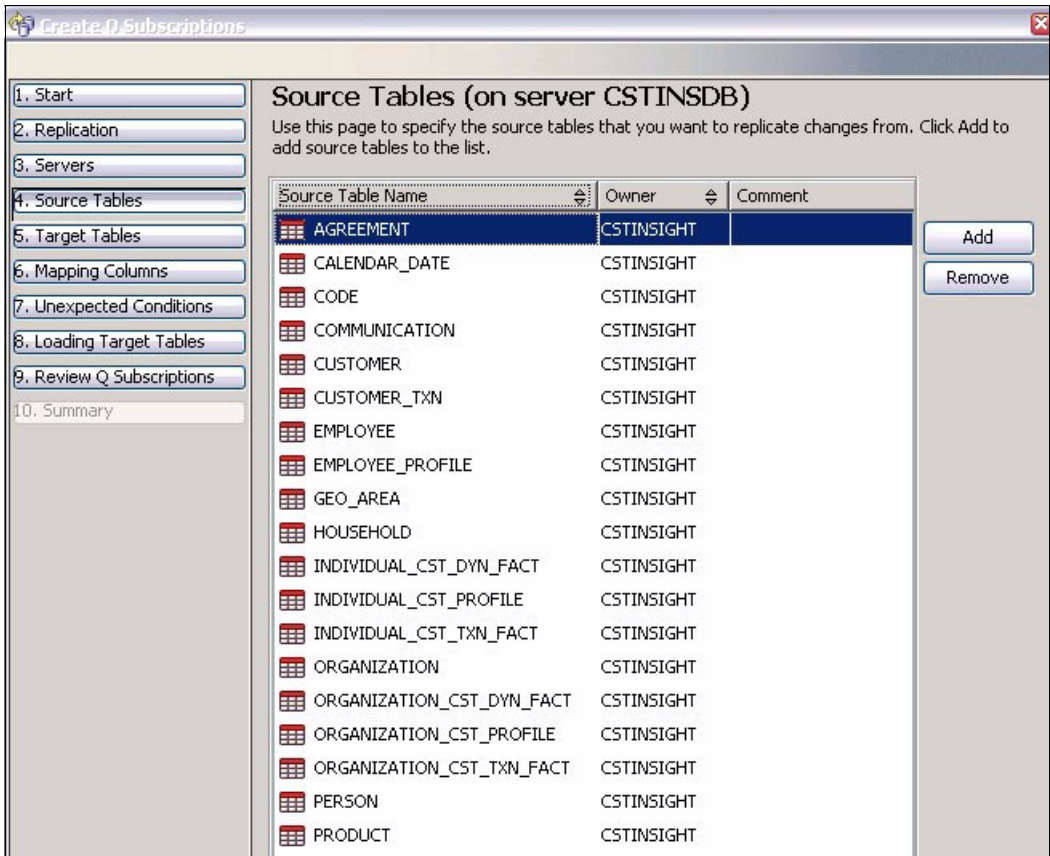


Figure 12-61 Selected tables for replication

8. Select a profile.

Select the profile for the target source; see Figure 12-62.

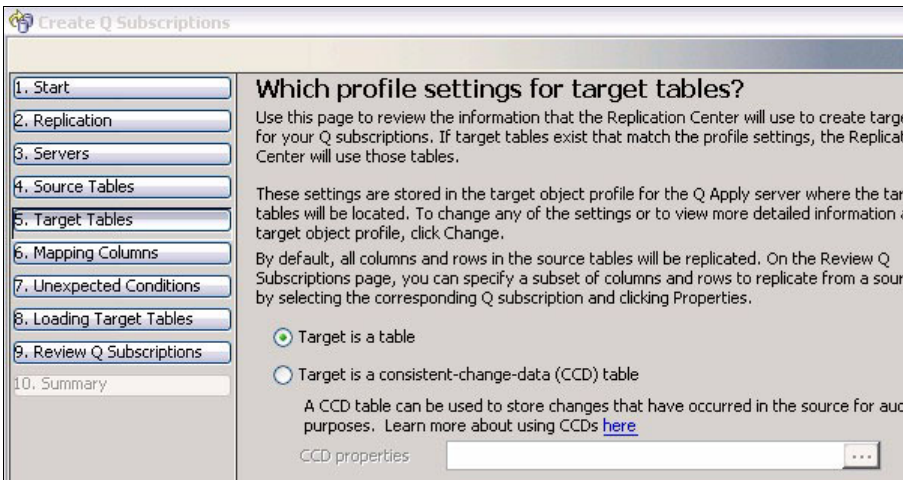


Figure 12-62 Q Replication configuration for the target tables

9. Table configuration.

You can configure the name convention for the source tables, specify a schema, table space, index, truncation rules, and the data type mapping; see Figure 12-63.

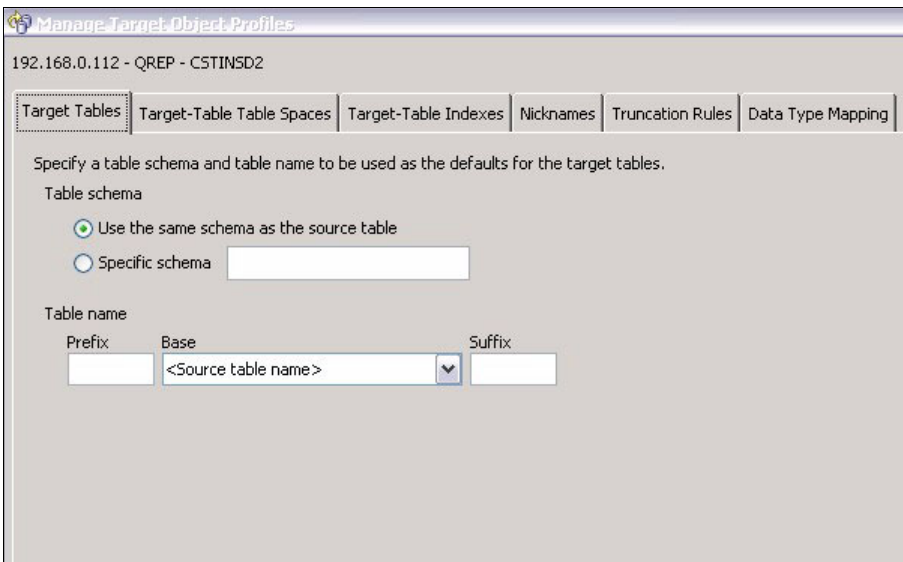


Figure 12-63 Q Replication advanced configuration for the target tables

10. Define the columns map.

Define the columns map from source to target, which can be by position or by name and data type; see Figure 12-64.

Create Q Subscriptions

1. Start
2. Replication
3. Servers
4. Source Tables
5. Target Tables
6. Mapping Columns
7. Unexpected Conditions
8. Loading Target Tables
9. Review Q Subscriptions
10. Summary

Which source columns map to which target columns?
If the target table for a Q subscription already exists, the Replication Center automatically attempts to map source columns to target columns, though you can change this mapping later. Use this page to specify the method that the Replication Center should use when mapping columns.

How should the Replication Center map source columns to target columns?

☒ Match column names and data types
Map together columns with identical names and compatible data types.

☐ Map by column position
Map together columns that are in the same position in the source tables and that have compatible data types.

Figure 12-64 Q Replication configuration for mapping source to target

11. Configure Q Apply for unexpected conditions.

Configure how Q Apply is to respond to unexpected conditions; see Figure 12-65.

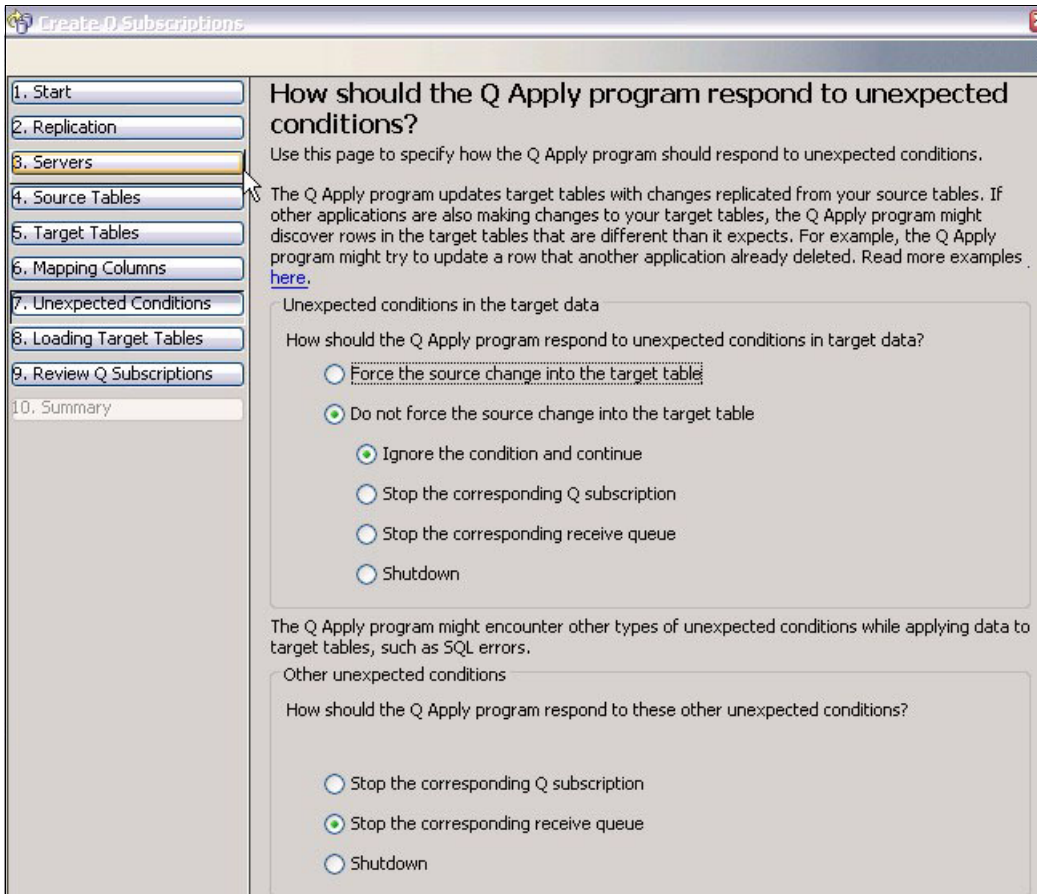


Figure 12-65 Q Replication configuration for unexpected conditions

12.Specify how to load target tables.

Specify how the target tables are to be loaded. In our case we allowed Q subscription to load the tables; see Figure 12-66.

1. Start

2. Replication

3. Servers

4. Source Tables

5. Target Tables

6. Mapping Columns

7. Unexpected Conditions

8. Loading Target Tables

9. Review Q Subscriptions

10. Summary

How should target tables be loaded?

Use this page to specify if and how your target tables will be loaded. When a Q subscription starts, the Q Apply program can initiate a load of the target tables with the utility or utilities that you specify.

Loading the target tables

How will the target table be loaded when a Q subscription starts?

☒ Automatic: The Q Apply program performs the load.

Choose appropriate load option: Best available: Let the Q Apply program choose a load me

What should Q Apply program do if target already has existing data? Replace existing data

☐ Manual: You perform the load manually and inform the Q Apply program when the load is corr

☐ None: The target table is not loaded

☐ Reload the target table if the source table is reloaded

Nicknames

How would you like to specify your nicknames?

☐ Let the Replication Center create the nicknames (Recommended)

Server definition

☐ Specify existing nicknames

If you choose this option, you must edit each Q subscription that you want to have loaded f
specify nicknames for them. Click Properties on the Review Q Subscriptions page to edit Q su

☒ Do not use a nickname

☒ Start all Q subscriptions automatically

Selecting this option causes the Q Capture program to start your new Q subscriptions automatically.
Q Capture program starts or is reinitialized.

Figure 12-66 Q Replication configuration for loading target

13.Review the Q subscription status.

Review the Q subscriptions and, if necessary, make corrections; see Figure 12-67.

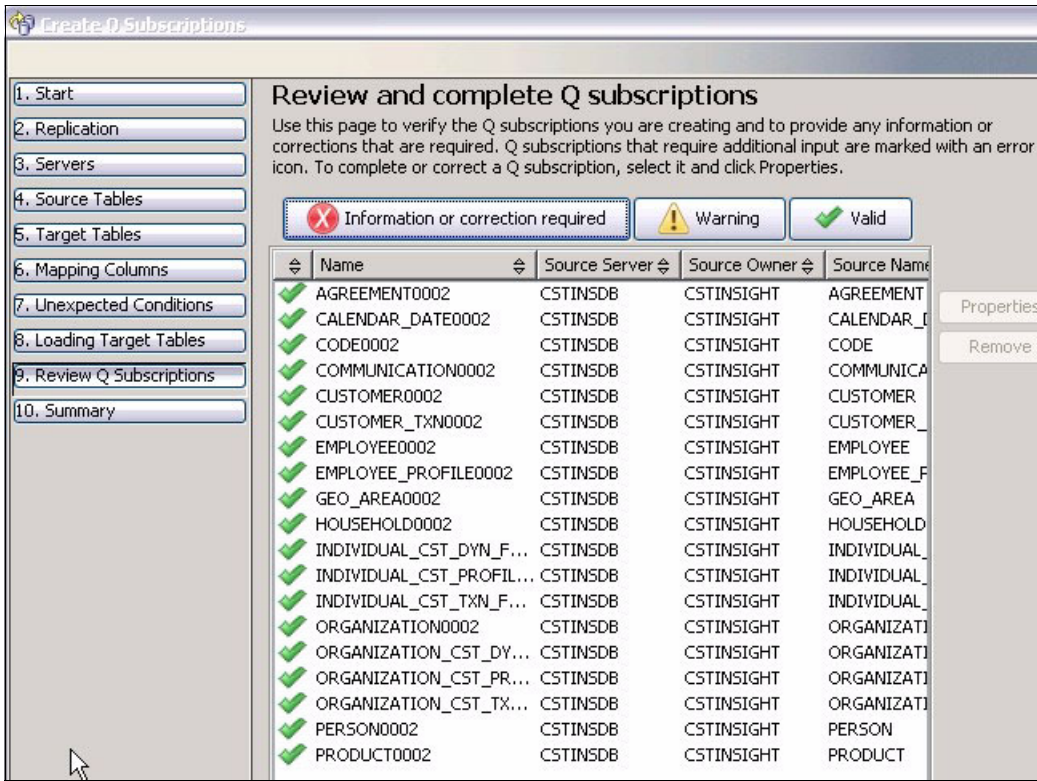


Figure 12-67 Q Replication subscription status page

14. Review the summary.

Review the summary and finish; see Figure 12-68.

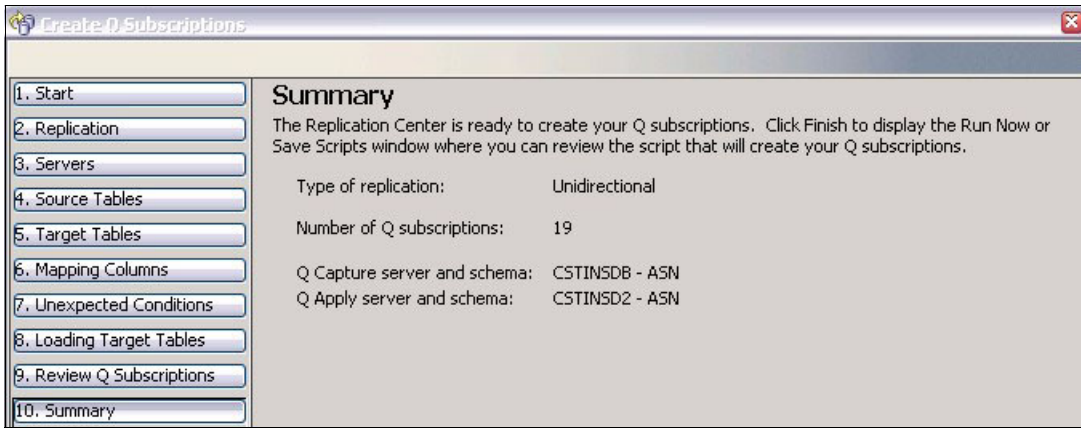


Figure 12-68 Q Replication subscription summary page

15.Enable the database for replication.

Finally, we enable the database for replication; see Figure 12-69.

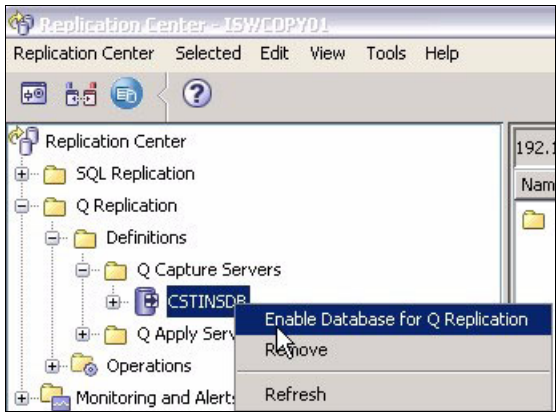


Figure 12-69 Enable a database for replication

You might be asked to perform a backup and to configure archive logging in the database. This is essential for Q Replication because it requires sending logs over to the target server; see Figure 12-70.



Figure 12-70 Q Replication advisory to configure logging and perform a backup

Redirect database connections using Optim Configuration Manager

Using Q Replication to replicate the transactions tables or tables that are important to the business and constantly being updated can be part of a recovery strategy. A benefit of this backup strategy is that the target server is available for access while the table records are replicating.

Optim Configuration Manager keeps track of the configuration changes in the monitored databases. Any change made to the source server is identified by Optim Configuration Manager and can be later applied to the target server. This allows DBAs to keep both the source and target database server configuration synchronized easily.

You can set up the Optim Configuration Manager to redirect users or applications on the source server to the target server in case the source target is not

available. Follow these steps to create and enable rules to redirect users or applications automatically:

1. Select the database to redirect.

Open the Optim Configuration Manager web interface. Click **Open** → **Control** → **Redirect Database Connection**. Select the database you want to redirect, as shown in Figure 12-71.

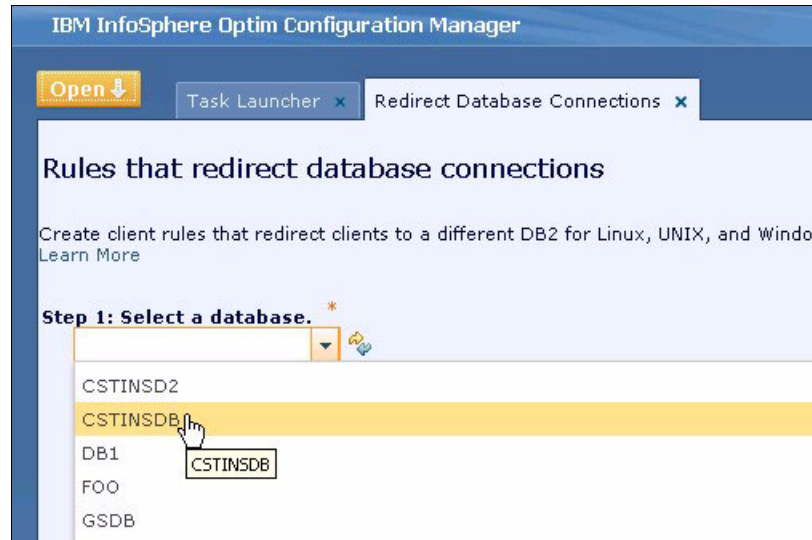


Figure 12-71 Select a database to create a redirect rule

2. Add a new rule.

Click **New** to create a new rule; see Figure 12-72.



Figure 12-72 Create a redirect rule

3. Specify the redirecting rule.

The Optim Configuration Manager editor allows you to configure the conditions by which your rule is triggered. For example, a rule can be triggered by a client IP address, by a specific WebSphere Application Server, or by a data source information.

This is also where you define the destination database, which is the database to which the connection will be redirected; see Figure 12-73.

Edit Client Rule 25 - Redirect database connections

Create a client rule that redirects clients to a different DB2 for Linux, UNIX, and Windows database or

Comment:

▼ Rule conditions for selecting the clients that will be redirected

Specify the clients that this rule affects by selecting attributes to filter for the clients.

Specify condition: AND Field: clientUUID is

Review the conditions that specify the clients that are affected by the rule. The first condition cannot be removed.

WHEN serverName IS 9.26.167.203 AND portNumber IS 50010 AND clientIP IS CSTINSD2 AND clientUUID IS

▼ Rule action that redirects the clients

Define a different DB2 for Linux, UNIX, and Windows database or a different DB2 for z/OS subsystem

Hostname/IP * 9.26.167.203

Port * 50010

Database Name * CSTINSD2

Figure 12-73 Properties of the redirect rule

- 4. Enable the rule.

After the rules are defined, enable the rules; see Figure 12-74.

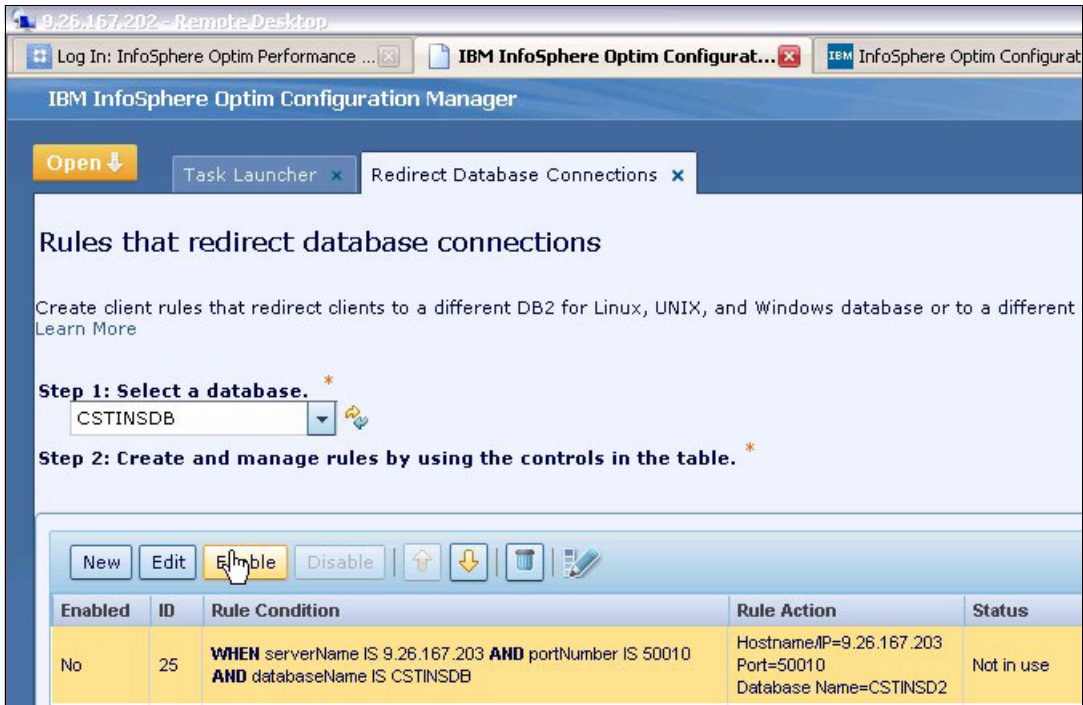


Figure 12-74 Enable a redirect rule

To use Optim Configuration Manager to explore a database client, the Optim Data Tools Runtime Client must be installed on each client machine. After it is installed and configured, you can use the sample application to test a redirect rule.

When we connect to the monitored database, Optim Configuration Manager identifies the IP address. Based on the rule created for this database and this IP address, it redirects the connection to another database.

Figure 12-75 on page 476 shows that our application connection has been redirected.

```

C:\>HOL\OCM\ZOS\module2\application3>RunApplication3.bat
===== application information =====
ip address: 9.26.167.202
===== data source information =====
name      : RedirectExampleDataSource
=====

Connection attempt #1
Database specified in the application: CSTINSDB, Actual database connected to: C
STINSDB
*** Database connection redirect occurred ***

Connection attempt #2
Database specified in the application: CSTINSDB, Actual database connected to: C
STINSDB
*** Database connection redirect occurred ***

```

Figure 12-75 Sample application to redirect a connection

In the Optim Configuration Manager, the status is Downloaded after a rule is in use; see Figure 12-76.

Rule Condition	Rule Action	Status
WHEN serverName IS 9.26.167.203 AND portNumber IS 50010 AND databaseName IS CSTINSDB AND clientIP IS 9.26.167.202	Hostname/IP=9.26.167.203 Port=50010 Database Name=CSTINSDB	Downloaded

Figure 12-76 Status of a rule changed when in use

InfoSphere Warehouse Editions comparison

This appendix details the various editions of InfoSphere Warehouse that are available to be purchased from IBM. Table A-1 lists both the features and the products included with each edition.

Note: This table is for informational use only and is not a definitive list. For more current and detailed information, see the individual product announcement letters or speak to an IBM representative.

Table A-1 InfoSphere Warehouse Editions and features summary

	Advanced Enterprise	Enterprise ^a	Enterprise Base	Advanced Departmental	Departmental	Developer
DB2 10 Enterprise Server Edition with Database Partitioning	✓	✓	✓	✓	✓	✓
Continuous Data Ingest	✓	✓	n/a	✓	n/a	✓

Multi-Temperature Data Management	✓	✓	n/a	✓	n/a	✓
Storage Optimization Feature (with Adaptive Compression)	✓	✓	n/a	✓	✓	✓
Label and Row Based Access Control	✓	✓	n/a	✓	✓	✓
Cubing Services	✓	✓	✓	✓	✓	✓
SQL Warehousing Tool (SQW)	✓	✓	✓	✓	✓	✓
InfoSphere Federation Server	✓	✓	✓	✓	✓	✓
Design Studio	✓	✓	✓	✓	✓	✓
InfoSphere Warehouse Administration Console	✓	✓	✓	✓	✓	✓
Cognos 10 BI	5 users	5 users	n/a	5 users	5 users	1 user
InfoSphere Replication Server*	✓	✓	✓	✓	✓	✓
DB2 Workload Management	✓	✓	n/a	✓	✓	✓
Optim Performance Manager	Extended Edition included	Extended Edition included	n/a	✓	✓	Extended Edition included
IBM Intelligent Miner®	✓	✓	n/a	✓	✓	✓

Text Analytics	✓	✓	n/a	✓	✓	✓
InfoSphere Data Architect	10 users	n/a	n/a	n/a	n/a	n/a
Optim Query Workload Tuner	✓	n/a	n/a	✓	n/a	n/a
Optim High Performance Unload	✓	n/a	n/a	n/a	n/a	n/a
DB2 Merge Backup	✓	n/a	n/a	n/a	n/a	n/a
DB2 Recovery Expert	✓	n/a	n/a	n/a	n/a	n/a
Optim Database Administrator	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio
Optim Configuration Manager	✓	n/a	n/a	n/a	n/a	n/a
Optim Development Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio	Now included in Data Studio
Data Studio	Free download	Free download	Free download	Free download	Free download	Free download
InfoSphere Warehouse Pack Customer Insight	✓	Optional Extra	Optional Extra	✓	Optional Extra	Optional Extra
InfoSphere Warehouse Pack Market and Campaign Insight	✓	Optional Extra	Optional Extra	✓	Optional Extra	Optional Extra
InfoSphere Warehouse Pack Supply Chain Insight	✓	Optional Extra	Optional Extra	✓	Optional Extra	Optional Extra

a. Certain usage restrictions apply to these components. For details, refer to the program's license information document.

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

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Dimensional Modeling: In a Business Intelligence Environment*, SG24-7138
- ▶ *InfoSphere Warehouse: A Robust Infrastructure for Business Intelligence*, SG24-7813
- ▶ *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912
- ▶ *Dynamic Warehousing: Data Mining Made Easy*, SG24-7418

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ “Predictive analytics on SAP with SPSS and InfoSphere Warehouse: Using cutting edge analytics technology to learn more about your customers”, Stefanie Schezinger, Thomas Schwarz, Michael Wurst, IBM Corp., July, 2010
- ▶ IBM white paper “Best practices: Temporal data management with DB2”, Mattias Nicola, IBM Corp., April, 2012
- ▶ “Change the game with in-database analytics: how to run SAS scoring models inside DB2” presentation by Eileen Lin, Mattias Nicola, and Subho Chatterjee, 2011 IBM Information on Demand® conference

- ▶ Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, and Bob Becker, *The Data Warehouse Lifecycle Toolkit: The Complete Guide to Dimensional Modeling*, (2nd ed), John Wiley and Sons, Inc., 2008

Online resources

These websites are also relevant as further information sources:

- ▶ IBM InfoSphere Warehouse Information Center
http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.dwe.navigate.doc%2Fwelcome_warehouse.html
- ▶ Cognos Business Intelligence Information Center
http://publib.boulder.ibm.com/infocenter/cbi/v10r1m0/index.jsp?topic=%2Fcom.ibm.swg.im.cognos.ug_rptstd_fin.10.1.0.doc%2Fug_rptstd_fin_id80business_insight_advanced.html
- ▶ Cognos documentation
http://www.ibm.com/support/entry/portal/documentation/software/cognos/cognos_business_intelligence_and_financial_performance_management
- ▶ 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/>
- ▶ eBook: *Getting Started with IBM Data Studio for DB2*
<http://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+IBM+Data+Studio+for+DB2>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

Solving Operational Business Intelligence with InfoSphere Warehouse Advanced Edition

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Solving Operational Business Intelligence with InfoSphere Warehouse Advanced Edition

Comprehensive solution for operational warehouse challenges

Learning-integrated business intelligence framework

Advanced analytics and integrated tool examples

IBM InfoSphere Warehouse is the IBM flagship data warehouse platform for departmental data marts and enterprise data warehouses. It offers leading architecture, performance, backup, and recovery tools that help improve efficiency and reduce time to market through increased understanding of current data assets, while simplifying the daily operations of managing complex warehouse deployments.

InfoSphere Warehouse Advanced Enterprise Edition delivers an enhanced set of database performance, management, and design tools. These tools assist companies in maintaining and increasing value from their warehouses, while helping to reduce the total cost of maintaining these complex environments.

In this IBM Redbooks publication we explain how you can build a business intelligence system with InfoSphere Warehouse Advanced Enterprise to manage and support daily business operations for an enterprise, to generate more income with lower cost. We describe the foundation of the business analytics, the Data Warehouse features and functions, and the solutions that can deliver immediate analytics solutions and help you drive better business outcomes.

We show you how to use the advanced analytics of InfoSphere Warehouse Advanced Enterprise Edition and integrated tools for data modeling, mining, text analytics, and identifying and meeting the data latency requirements. We describe how the performance and storage optimization features can make building and managing a large data warehouse more affordable, and how they can help significantly reduce the cost of ownership. We also cover data lifecycle management and the key features of IBM Cognos Business Intelligence.

This book is intended for data warehouse professionals who are interested in gaining in-depth knowledge about the operational business intelligence solution for a data warehouse that the IBM InfoSphere Warehouse Advanced Enterprise Edition offers.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks