IBM

Data Mart Consolidation: Getting Control of Your Enterprise Information

- Managing your information assets and minimizing operational costs
- Enabling a single view of your business environment
- Minimizing or eliminating those data silos

Chuck Ballard
Amit Gupta
Vijaya Krishnan
Nelson Pessoa
Olaf Stephan

Archived

Redbooks

ibm.com/redbooks

Archived

International Technical Support Organization

Data Mart Consolidation: Getting Control of Your Enterprise Information

July 2005

Archived

SG24-6653-00

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

Archived

First Edition (July 2005)

This edition applies to DB2 UDB V8.2, DB2 Migration ToolKit V1.3, WebSphere Information Integrator V8.2, Oracle Database 9i, and Microsoft SQL Server 2000.

© Copyright International Business Machines Corporation 2005. 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 that wrote this redbook xii
Become a published author xiv
Comments welcome xiv

Chapter 1. Introduction 1
1.1 Managing the enterprise data 4
1.1.1 Consolidating the data warehouse environment 4
1.2 Management summary 5
1.2.1 Contents abstract 8

Chapter 2. Data warehousing: A review 11
2.1 Data warehousing 12
2.1.1 Information environment 13
2.1.2 Real-time business intelligence 14
2.1.3 An architecture 16
2.1.4 Data warehousing implementations 18
2.2 Advent of the data mart 20
2.2.1 Types of data marts 21
2.3 Other analytic structures 23
2.3.1 Summary tables, MQTs, and MDC 23
2.3.2 Online analytical processing 27
2.3.3 Cube Views 28
2.3.4 Spreadsheets 29
2.4 Data warehousing techniques 30
2.4.1 Operational data stores 30
2.4.2 Data federation and integration 33
2.4.3 Federated access to real-time data 37
2.4.4 Federated access to multiple data warehouses 38
2.4.5 When to use data federation 39
2.4.6 Data replication 41
2.5 Data models 42
2.5.1 Star schema 43
2.5.2 Snowflake schema 44
2.5.3 Normalization 46

© Copyright IBM Corp. 2005. All rights reserved.

Chapter 3. Data marts: Reassessing the requirement 49
3.1 The data mart phenomenon 51
3.1.1 Data mart proliferation. 52
3.2 A business case for consolidation 54
3.2.1 High cost of data marts 54
3.2.2 Sources of higher cost 56
3.2.3 Cost reduction by consolidation 57
3.2.4 Metadata: consolidation and standardization 60
3.2.5 Platform considerations. 62
3.2.6 Data mart cost analysis sheet 62
3.2.7 Resolving the issues 64
3.3 Summary 65

Chapter 4. Consolidation: A look at the approaches 67
4.1 What are good candidates for consolidation? 68
4.1.1 Data mart consolidation lifecycle. 69
4.2 Approaches to consolidation 71
4.2.1 Simple migration 72
4.2.2 Centralized consolidation 76
4.2.3 Distributed consolidation 82
4.2.4 Summary of consolidation approaches 84
4.3 Combining data schemas 88
4.3.1 Simple migration approach 88
4.3.2 Centralized consolidation approach 89
4.3.3 Distributed consolidation approach 91
4.4 Consolidating the other analytic structures 93
4.5 Other consolidation opportunities 96
4.5.1 Reporting environments 96
4.5.2 BI tools 100
4.5.3 ETL processes 101
4.6 Tools for consolidation 103
4.6.1 DB2 Universal Database. 104
4.6.2 DB2 Data Warehouse Edition 104
4.6.3 WebSphere Information Integrator 106
4.6.4 DB2 Migration ToolKit 108
4.6.5 DB2 Alphablox 108
4.6.6 DB2 Entity Analytics 110
4.6.7 DB2 Relationship Resolution 111
4.6.8 Others... 111
4.7 Issues with consolidation. 113
4.7.1 When would you not consider consolidation? 114
4.8 Benefits of consolidation 115

Archived

Chapter 5. Spreadsheet data marts . . . 117
5.1 Spreadsheet usage in enterprises . . . 117
5.1.1 Developing standards for spreadsheets . . . 118
5.2 Consolidating spreadsheet data . . . 121
5.2.1 Using XML for consolidation . . . 122
5.2.2 Transferring spreadsheet data to DB2 with no conversion . . . 129
5.2.3 Consolidating spreadsheet data using DB2 OLAP Server . . . 132
5.3 Spreadsheets and WebSphere Information Integrator . . . 133
5.3.1 Adding spreadsheet data to a federated server . . . 133
5.3.2 Sample consolidation scenario using WebSphere II . . . 137
5.4 Data transfer example with DB2 Warehouse Manager . . . 139
5.4.1 Preparing the source spreadsheet file . . . 139
5.4.2 Setting up connectivity to the source file . . . 139
5.4.3 Setting up connectivity to the target DB2 database . . . 140
5.4.4 Sample scenario . . . 140

Chapter 6. Data mart consolidation lifecycle . . . 149
6.1 The structure and phases . . . 150
6.2 Assessment . . . 151
6.2.1 Analytic structures . . . 151
6.2.2 Data quality and consistency . . . 154
6.2.3 Data redundancy . . . 160
6.2.4 Source systems . . . 161
6.2.5 Business and technical metadata . . . 162
6.2.6 Reporting tools and environment . . . 163
6.2.7 Other BI tools . . . 166
6.2.8 Hardware/software and other inventory . . . 167
6.3 DMC Assessment Findings Report . . . 168
6.4 Planning . . . 178
6.4.1 Identify a sponsor . . . 179
6.4.2 Identify analytical structures to be consolidated . . . 179
6.4.3 Select the consolidation approach . . . 179
6.4.4 Other consolidation areas . . . 180
6.4.5 Prepare the DMC project plan . . . 181
6.4.6 Identify the team . . . 181
6.5 Implementation recommendation report . . . 182
6.6 Design . . . 183
6.6.1 Target EDW schema design . . . 183
6.6.2 Standardize business definitions and rules . . . 185
6.6.3 Metadata standardization . . . 186
6.6.4 Identify dimensions and facts to be conformed . . . 187
6.6.5 Source to target mapping . . . 191
6.6.6 ETL design . . . 191

Archived

6.6.7 User reports requirements . . . 194
6.7 Implementation . . . 195
6.8 Testing. . . . 196
6.9 Deployment . . . 196
6.10 Continuing the consolidation process . . . 197

Chapter 7. Consolidating the data . . . 199
7.1 Converting the data . . . 200
7.1.1 Data conversion process. . . . 200
7.1.2 Time planning . . . 201
7.1.3 DB2 Migration ToolKit . . . 202
7.1.4 Alternatives for data movement . . . 204
7.1.5 DDL conversion using data modeling tools . . . 207
7.2 Load/unload. . . . 208
7.3 Converting Oracle data . . . 208
7.4 Converting SQL Server . . . 211
7.5 Application conversion . . . 214
7.5.1 Converting other Java applications to DB2 UDB . . . 216
7.5.2 Converting applications to use DB2 CLI/ODBC . . . 218
7.5.3 Converting ODBC applications . . . 220
7.6 General data conversion steps . . . 220

Chapter 8. Performance and consolidation . . . 227
8.1 Performance techniques . . . 229
8.1.1 Buffer pools . . . 229
8.1.2 DB2 RUNSTATS utility . . . 230
8.1.3 Indexing. . . . 232
8.1.4 Efficient SQL . . . 235
8.1.5 Multidimensional clustering tables . . . 236
8.1.6 MQT . . . 240
8.1.7 Database partitioning . . . 241
8.2 Data refresh considerations . . . 244
8.2.1 Data refresh types. . . . 244
8.2.2 Impact analysis . . . 245
8.3 Data load and unload . . . 245
8.3.1 DB2 Export and Import utility . . . 246
8.3.2 The db2batch utility . . . 249
8.3.3 DB2 Load utility . . . 250
8.3.4 The db2move utility . . . 253
8.3.5 The DB2 High Performance Unload utility . . . 253

Chapter 9. Data mart consolidation: A project example . . . 255
9.1 Using the data mart consolidation lifecycle . . . 256
9.2 Project environment . . . 257

Archived

9.2.1 Overview of the architecture 257
9.2.2 Issues with the present scenario 260
9.2.3 Configuration objectives and proposed architecture 262
9.2.4 Hardware configuration 264
9.2.5 Software configuration 265
9.3 Data schemas 266
9.3.1 Star schemas for the data marts 266
9.3.2 EDW data model 272
9.4 The consolidation process 274
9.4.1 Choose the consolidation approach 274
9.4.2 Assess independent data marts 275
9.4.3 Understand the data mart metadata definitions 277
9.4.4 Study existing EDW 278
9.4.5 Set up the environment needed for consolidation 280
9.4.6 Identify dimensions and facts to conform 280
9.4.7 Design target EDW schema 282
9.4.8 Perform source/target mapping 283
9.4.9 ETL design to load the EDW from data marts 283
9.4.10 Metadata standardization and management 291
9.4.11 Consolidating the reporting environment 293
9.4.12 Testing the populated EDW data with reports 294
9.5 Reaping the benefits of consolidation 298

Appendix A. Consolidation project example: Table descriptions 301
Data schemas on the EDW 302
Data schemas on the ORACLE data mart 308
Data schemas on the SQL Server 2000 data mart 310

Appendix B. Data consolidation examples 315
DB2 Migration ToolKit 316
Consolidating with the MTK 318
Example: Oracle 9i to DB2 UDB 324
Example: SQL Server 2000 to DB2 UDB 335
Consolidating with WebSphere II 344
Example - Oracle 9i to DB2 UDB 344
Example - SQL Server to DB2 UDB 353

Appendix C. Data mapping matrix and code for EDW 365
Source to target data mapping matrix 366
SQL ETL Code to populate the EDW 376

Appendix D. Additional material 381
Locating the Web material 381
Using the Web material 382

Archived

How to use the Web material 382

Abbreviations and acronyms 383

Glossary 387

Related publications 393
IBM Redbooks 393
Other publications 393
How to get IBM Redbooks 394
Help from IBM 394

Index 395

Archived

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 give 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

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 illustrates 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.

© Copyright IBM Corp. 2005. All rights reserved.

Trademarks

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

AIX®
Approach®
Architecture™
AS/400®
Cube Views™
Database 2™
Distributed Relational Database
DB2®
DB2 Connect™
DB2 Extenders™
DB2 OLAP Server™
DB2 Universal Database™
DRDA®
@server®
Informix®
Intelligent Miner™
iSeries™
IBM®
IMS™
Lotus®
OS/390®
Rational®
Rational Rose®
Redbooks™
Redbooks (logo) ™
Red Brick™
WebSphere®
Workplace™
z/OS®

The following terms are trademarks of other companies:

Solaris, J2SE, J2EE, JVM, JDK, JDBC, JavaBeans, Java, EJB, and Enterprise JavaBeans are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

Archived

Preface

This IBM Redbook is primarily intended for use by IBM® Clients and IBM Business Partners involved with data mart consolidation.

A key direction in the business intelligence marketplace is towards data mart consolidation. Originally data marts were built for many good reasons, such as departmental or organizational control, faster query response times, easier and faster to design and build, and fast application payback.

However, data marts did not always provide the best solution when it came to viewing the business enterprise as an entity. And consistency between the data marts was, and is, a continuing source of frustration with business management. They to provide benefits to the department or organization to whom they belong, but typically do not give management the information they need to efficiently and effectively run the business. This has become a real concern with the current emphasis on, and dramatic benefits gained from, business performance management.

In many cases data marts have led to the creation of departmental or organizational data silos. That is, information is available to a specific department or organization, but not integrated across all the departments or organizations. Worse yet, many of these silos were built without concern for the others. This led to inconsistent definitions of the data, inconsistent collection of data, inconsistent currency of the data across the organization, and so on. The result is an inconsistent picture of the business for management, and an inability to achieve good business performance management. The solution is to consolidate those data silos to provide management a consistent and complete set of information for the business needs.

In this redbook we provide details on the data warehousing environment, and best practices for consolidating and integrating your environment to produce the information you need to best manage your business.

We are certain you will find this redbook informative and helpful, and of great benefit as you develop your data mart consolidation strategies.

© Copyright IBM Corp. 2005. All rights reserved.

The team that wrote this redbook

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

Some team members worked locally at the International Technical Support Organization - San Jose Center, while others worked from remote locations. The team members are depicted below, along with a short biographical sketch of each:

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

Amit Gupta is a Data Warehousing Consultant in IBM, India. He is a Microsoft® Certified Trainer, MCDBA, and a Certified OLAP Specialist. He has 6 years of experience in the areas of databases, data management, data warehousing, and business intelligence. He teaches extensively on dimensional modeling, data warehousing, and BI courses in IBM India. His areas of expertise include dimensional modeling, data warehousing, and metadata management. He holds a degree in Electronics and Communications from Delhi Institute of Technology, Delhi University, New Delhi, India.

Vijaya Krishnan is a Database Administrator in IBM Global Services, Bangalore, India in the Siebel Technology center department. He has over 8 years of experience in application development, DB2® UDB database administration and design, business Intelligence, and data warehouse development. He is an IBM certified Business Intelligence solutions designer and an IBM certified DB2 Database Administrator. He holds a Bachelors degree in engineering from the University of Madras in India.

Nelson Pessoa is a Database Administrator at IBM Brazil where he has worked for 6 years. He holds a Bachelors degree in Computer Science from the Centro Universitário Nove de Julho, São Paulo, Brazil, and currently works as a Systems Specialist working with customers around the country and in internal projects. He has also worked with other IRM applications and ITIL disciplines. His areas of expertise include Data WareHouse, DB2, Data Integration, Data Modeling, Programming ETLs, Reporting Applications.

Olaf Stephan is a Data Integration Specialist at the E&TS, Engineering & Technology Services organization in Mainz, Germany. He has 6 years of experience in DB2 UDB, data management, data warehousing, business intelligence, and data integration. He holds a Masters degree in Electrical Engineering, specializing in Communications Technology, from the University of Applied Sciences, Koblenz, Germany.

Special acknowledgement

Henry Cook, Manager, BI Sector, Competitive Team, EMEA, UK. Henry is an expert in data warehousing and data mart consolidation. He provided guidance when forming the structure of the book, contributed significant content, and offered valuable feedback during the technical review process

Other contributors

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

From IBM locations worldwide
Garrett Hall - DB2 Information Management Skills, Austin, Texas
Barry Devlin - Software Group, Lotus® and IBM Workplace™, Dublin, Ireland
Bill O'Connell - Senior Technical Staff Member, Chief BI Architect, DB2 UDB Development, Markham, ON Canada
Stephen Addison - SWG Services for Data Management, UK
Keith Brown - Business Intelligence Practice Leader, UK
Paul Gittins - Software Sales Consultant, UK
Tim Newman - DB2 Alphablox Technical Sales, Bedfont, UK
Paul Hennessey - IBM Global Services, CRM Marketing and Analytics, UK
Karen Van Evans - Application Innovation Services - Business Intelligence, Markham, ON Canada
John Kling - IGS Consulting and Services, Cincinnati, Ohio
David Marcotte - Retail Industry Software Sales, Waltham, Massachusetts

Archived

Bruce Johnson - Consultant and Data Architect, IGS, Minneapolis, Minnesota
Aviva Phillips - Data Architect, Southfield, Michigan
Koen Berton - Consultant, IGS, Belgium

From the International Technical Support Organization, San Jose Center
Mary Comianos - Operations and Communications
Yvonne Lyon - Technical Editor
Deanna Polm - Residency Administration
Emma Jacobs - Graphics

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  `ibm.com/redbooks`

- Send your comments in an email to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

Archived

1

Introduction

In this redbook, we discuss the topic of data mart consolidation. That includes the issues involved and approaches for resolving them, as well as the requirements for, and benefits of, data mart consolidation.

But why consolidate data marts? Are they not providing good information and value to the enterprise? The answers to these and similar questions are discussed in detail throughout this book. In general, businesses are consolidating their data marts for three basic reasons:

1. **Cost savings:** There is a significant cost associated with data marts in the form of such things as:
   a. Additional servers.
   b. Additional software licenses, such as database management systems and operating systems.
   c. Operating and maintenance costs for activities such as software updates, backup/recovery, data capture and synchronization, data transformations, and problem resolution.
   d. Additional resources for support, including the cost of their training and ongoing skills maintenance — particularly in a heterogeneous environment.
   e. Additional networks for their connectivity and operations.

© Copyright IBM Corp. 2005. All rights reserved.

f. Additional application development costs, and associated tools cost, for servicing the multiple data marts — particularly in a heterogeneous environment.

2. **Improved productivity of developers and users:** Consolidating the data marts enables improved hardware, software, and resource standardization, resulting in minimizing the heterogeneous environment. This means fewer resource requirements, less training and skills maintenance, fewer development tasks in the minimized development environments, reuse of existing applications, and enhanced standardization.
3. **Improved data quality and integrity:** This is a significant advantage that can restore or enhance user confidence in the data. Implementation of common and consistent data definitions, as well as managed data update cycles, can result in query and reporting results that are consistent across the business functional areas of the enterprise.

This is not to say that you should not have any data marts. Data marts can satisfy many objectives, such as departmental or organizational control, faster query response times, easier and faster to design and build, and fast payback. However, this is not true in all situations. As you will discover in reading this redbook, whether the data mart is dependent or independent is a major consideration in determining its value.

And, data marts may not always provide the best solution when it comes to viewing the business enterprise as a whole. They may provide benefits to the department or organization to whom they belong, but may not give management the information required to efficiently and effectively manage the performance of the enterprise.

For example, in many cases the data marts led to the creation of departmental or organizational, independent, data silos. That is, data was available to the specific department or organization, but was not integrated across all the departments or organizations. Worse yet, many were built without concern for the other business areas. They may also have resulted from activities such as mergers and acquisitions.

Typically these multiple data marts were even built on different technologies and with hardware and software from multiple vendors. This led to inconsistent definitions of the data, inconsistent collection of data, inconsistent collection times for the data, difficult sharing and integration, and so on. The result is an inconsistent picture of the business for management, and an inability to do good business performance management. The solution is to consolidate those data silos to provide management with the information they need.

If you choose to consolidate your data marts and grow your enterprise data warehousing environment, IBM DB2 is an ideal platform on which to do so. IBM DB2 Universal Database™ (DB2 UDB) is the leading database management system in the world today. It provides the robust power and capability to consolidate your data marts into the data warehouse you need to meet your enterprise information requirements. DB2 can provide scalability for continued growth and the performance and response times to satisfy user requirements, along with outstanding value for your money for a high return on your investment. This makes DB2 an excellent strategic choice for your data warehousing environment.

Creating a robust enterprise data warehousing environment is more critical than ever today. This is because the business environment is changing at an ever-increasing rate, with speed and flexibility as key requirements to meet business goals, and perhaps even to remain viable as a business entity. Businesses must be flexible enough to change quickly to meet customer demands and shareholder expectations. It is the only way to enable growth and maintain business leadership. These changes can include such things as changing business processes, engaging new markets, developing and purchasing new business software, and maintaining a flexible and dynamic support infrastructure. And to remain competitive, the speed of developing, changing, and reporting on these activities is of the essence.

The companies that can meet these objectives of speed and flexibility will be the market leaders. The term used, when referring to this capability, is business performance management. And the key requirement for it is the availability of current information — from an enterprise-wide perspective. To get that information requires the integration of consistent data from the departments and organizations that comprise the enterprise. It means having data that represents a single consistent view of the enterprise, rather than a view only of a department or organization, for making the decisions required to manage the performance of the enterprise. Too often, the data is contained in multiple data marts around the organization. Getting a single consistent view of the enterprise from these data marts is not easy, and often not even possible.

Note: We use the term "existing systems" in this redbook to refer to systems that have been implemented with non-current technology and/or heterogeneous technologies. These types of systems are sometimes referred to as legacy systems. They may still satisfy the purpose for which they were designed, but are difficult and expensive to migrate to, or integrate with, systems based on newer technology with enhanced functionality.

1.1 Managing the enterprise data

The highly competitive industry environment is demanding change to enable not only business success and profitability, but also to provide tools and capabilities that can enable business survival! One such capability, previously discussed, is business performance management. It is a proactive ability to enable business managers to manage!

Management must be able to clearly understand their current business status in order to make the decisions required to enable them to meet their performance measurements. The primary key requirement for this is high quality, consistent data. It is this requirement that is fueling the current drive for data mart consolidation.

Data marts, and other decision support databases and analytic data structures, have been used as the base for many solutions in companies around the world. But, although they have helped satisfy some of the business needs of individual business areas, they have created an even bigger set of business issues for the enterprise. Here are some of the issues involved:

- Data in these data marts is frequently incomplete and inconsistent with other data sources in the enterprise.
- The results and conclusions derived from these data marts are potentially inaccurate and misleading. Reasons for this are discussed throughout this redbook.
- Resources to develop and maintain these data marts are being diverted from the many other projects that could better benefit the enterprise. In particular, this means a data warehousing environment that could provide a high quality and consistent set of data for use across the enterprise.

Since data marts can enable benefits to individual departments or business areas, the result has been their uncontrolled proliferation. And companies are realizing that the cost now outweighs those benefits, particularly from an enterprise point of view. The solution is seen to be the consolidation of many of these data marts into a well structured and managed enterprise data warehouse.

1.1.1 Consolidating the data warehouse environment

Businesses have learned that if they are to manage and meet their business performance goals, having high quality, consistent data for decision-making is a must. There is an aggressive movement to get control of the data, and manage it from an enterprise perspective. Managing this valuable business information asset is a requirement for meeting business measurements and shareholder expectations.

Consolidating the enterprise data is a major step in getting control. And, having it managed from an enterprise perspective is the key to meeting the enterprise goals. It is the only way to give management that much sought after goal of a “single view of the enterprise, or single version of the truth”, that is so desired, and required.

There are many benefits in data mart consolidation, both tangible and intangible. Typically they center around the need and desire to save money, cut costs, and enhance data quality, consistency, and availability. Many organizations have adopted a practice of creating a new data mart, or database, every time a new data analysis requirement arises. This exacerbates the problems of data proliferation, data inconsistency, non-current data sources, and increasing data maintenance costs. It is this practice that must be stopped.

So, what are the benefits? Here are a few of the specific benefits to consider as we begin our discussion:

- Realize significant tangible cost savings by eliminating redundant IT hardware, software, data, and systems, and the associated development and maintenance expenses.
- Eliminate uncertainty and improve decision-making by establishing a high-quality, managed, and consistent source of analytic data.
- Improve productivity and effectiveness through the application of best practices in business intelligence.
- Establish a data management environment that can enable you to have confidence in both decision making, and regulatory reporting (particularly important for compliance with regulations such as Basel II, Sarbanes Oxley, IAS, and IFRS).
- Enhance the agility and responsiveness of an enterprise to new requirements and opportunities.

1.2 Management summary

In this redbook we discuss the topic of data mart consolidation. We also demonstrate the results of a sample consolidation project developed in our test environment. These test results add to the publicly available documentation purporting the benefits of data mart consolidation. Included also are a number of best practices to help achieve your consolidation goals.

Data mart consolidation comprises a number techniques and technologies relative to data management. Some of these are very similar, and in fact are inter-related with consolidation. Prime examples are data integration, data federation, and data consolidation. Here we give a brief description of each:

Archived

- **Integration:** Data from multiple, often heterogeneous, sources are accessed and transformed to a standard data definition and data type, combined, and then typically stored in a common or consistent format, on a common or consistent platform for on-going use.
- **Federation:** Data from multiple, typically heterogeneous, sources are accessed in-place, transformed to a common data definition and data type, and combined. This is typically to satisfy the immediate requirements of a query or reporting application, rather than being stored for on-going use.
- **Consolidation:** This is basically a specific form of integration. It may require modifying the data model of the target consolidated database and perhaps the source database. And, you may either physically move the source data to the target database, or perhaps just conform the dimensions of the source and target databases. Depending on the consolidation approach, the transactions populating the source may be modified to directly populate the target consolidated database.

You can see that they are similar in some respects, but different in others. For a better understanding, we have provided a summary of their characteristics, listed a number of their attributes, and then described how they relate. This technology summary is presented in Table 1-1.

Table 1-1 Technology attributes

Attribute	Integration	Federation	Consolidation
General characteristics	Logically combining and inter-relating (joining) data from multiple sources such that the result conforms to a single desired data model. Results are typically stored in physical data targets.	Joining of data from multiple distributed heterogeneous data sources, that can then be viewed as if it were local. The data is typically not permanently stored in new data targets.	Integrating data from various analytical structures into a single desired data model, and stored in physical data targets. The sources, in most situations, will then be eliminated. The various approaches for data consolidation are detailed in Chapter 4.
Combines data from disparate data sources into a common platform	Yes	Typically, no. Existing analytical structures remain in place, but linked through shared keys, columns and global metadata.	May or may not, depending upon the consolidation approach selected. Refer to Chapter 4 for more details.

Attribute	Integration	Federation	Consolidation
Performance	Typically improved because fewer join operations, and data is then retrieved from the same integrated data source.	Depends on the data sources. Can be an issue with multiple heterogeneous and distributed data sources. Particularly true if operational sources are involved. But this can be offset by the significant improvement in functionality.	Depends on the consolidation approach used. Queries on centralized servers are faster than those running on distributed servers.
Can support a real-time environment	Yes	Yes	Yes
Includes data from multiple environments	Yes	Yes	Yes
A collection of technologies used	Yes	Yes	Yes
Includes data transformation	Yes, but only while creating the integrated data target.	Yes, and can be ongoing with every query.	Yes, but only while creating the integrated data target
Results in data consistency	Yes	Dependent on state of data sources.	Yes
Manages data concurrency	Yes	No	Yes
Data is stored on one logical server	Yes	Typically no	Yes
Metadata integrated	Yes	May or may not be.	Metadata is standardized using the centralized consolidation approach, but not with simple migration. Some level of achieved with distributed consolidation, with the implementation of conformed dimensions. For more details, refer to Chapter 4.

Archived

Attribute	Integration	Federation	Consolidation
When to use a particular approach	When there is permission to copy and use data from the multiple data sources.	Copying data may not be allowed. There is always the issue of concurrency and latency with this approach. Also issue of performance when accessing operational data sources.	Depends on the approach used. For more details, refer to Chapter 4.

Data warehousing

To understand data mart consolidation, we must have a common understanding of data warehousing and some of the terminology. Unfortunately, many of the terms used in data warehousing are not really standardized. There are even differences among the well known thought leaders in data warehousing. This has lead to the proliferation of many meanings for the same or similar terms. And, it has introduced a good deal of misunderstanding.

We will discuss data warehousing terminology, not to develop definitions, but to enable a more common understanding as you read this redbook. These discussions are primarily contained in Chapter 2.

1.2.1 Contents abstract

In this section we have included a brief description of the topics presented in the redbook, and we describe how they are organized. The information presented includes some high level product overviews, but is primarily oriented to detailed technical discussions of how you can consolidate your data marts into a robust DB2 UDB data warehouse. Depending on your interest, level of detail, and job responsibility, you may want to be selective in the sections where you focus your attention. We have organized this redbook to enable that selectivity.

Our focus is on providing information to help as you develop a plan, and then execute it to consolidate your data marts on DB2 UDB.

Let us get started by looking at a brief overview of the chapter contents:

- **Chapter 1** introduces the objectives of this redbook, as well as a brief management summary. It also provides a brief abstract of the chapter contents to help guide you in your reading selections.

- **Chapter 2** is a brief review of data warehousing concepts and the various types of implementations, such as centralized, hub and spoke, distributed, and federated. We discuss data warehouse architectures and components, and position the introduction of data marts. Included is a discussion of the different types of analytic structures, such as spreadsheets, that can be considered to be data marts, and their high cost of development and maintenance.
- **Chapter 3** introduces the prime topic of the redbook, data mart consolidation. We describe the phenomenon of data mart proliferation and associated high cost. In this chapter, we discuss issues associated with data marts and the business case for consolidation. This includes the topic of the high costs associated with data marts, and other business issues that surround their use. We include approaches for identifying and determining both tangible and intangible costs associated with data mart proliferation.
- **Chapter 4** continues with the consolidation story, which can entail a good deal more than might be expected. We discuss the different approaches for consolidating, based on your particular environment and desires. We delve into associated activities, such as report consolidation, data conversion and migration, tools to help, and some guidelines for determining the best approach for you. Included is a discussion on the various risks and issues involved with the consolidation process and the circumstances under which it might not be appropriate. We also introduce what we call the data mart consolidation lifecycle, which can help guide you, and we overview a number of tools that can provide the capabilities you need.
- **Chapter 5** focuses on a specific type of data mart-like analytic structure that is common to every enterprise — that is, the spreadsheet. The objective is to enable you to get the valuable information from those spreadsheets into a form that makes the data more easily available to others in the enterprise.
- **Chapter 6** gives some good planning and implementation planning information, tools, and techniques. We have developed this information with a data mart consolidation lifecycle approach. This will help in the assessment, planning, design, implementation, and testing of your consolidation processes. It will help you understand where you are going before you start the trip.
- **Chapter 7** gets you into some of the technical topics related to heterogeneous data consolidation, such as data conversion, migration, and loading. We provide information on tools and techniques, and specific data type issues encountered in our own test environment — that is, data modeling, data model consolidation, and data type conversion. We show examples from our consolidation of Oracle and SQL Server into our DB2 UDB enterprise data warehouse.

Archived

- **Chapter 8** gives us the opportunity to deal with the issue of performance. There may be some concerns that users might have about performance, since this is typically one of the primary reasons for having created data marts. We discuss a number of approaches, techniques, and DB2 capabilities that can remove performance and ongoing maintenance concerns from the consolidation decision.
- **Chapter 9** is where we bring things together and describe an example consolidation project we completed. We consolidated two independent data marts that existed on Oracle and SQL Server into our DB2 enterprise data warehouse. We describe the approach and process we selected, and the tasks we completed.

Appendix A provides some of the technical details from our sample consolidation project for those who would like to better understand the details. In particular, it lists the elements of the Oracle, SQL Server, and DB2 data schemas that were used.

Appendix B also provides more details on our sample consolidation project. In particular, we give an overview of the DB2 Migration ToolKit and the WebSphere® Information Integrator. These products can play an important role in converting and migrating data in your consolidation project. In particular, we detail the migration tasks we performed to get out data from Oracle and SQL Server to DB2.

Appendix C finishes out the technical details from our consolidation example. We show you the data mapping matrix used in our sample project, as well as the ETL code that moved the data from Oracle and SQL Server to DB2.

Well, that is a brief look at the chapter contents. We believe you should read the entire redbook, but we also understand that you have specific areas of interest. This overview should help you select and prioritize your reading.

Archived

2

Data warehousing: A review

In this chapter we provide an introduction to data warehousing, describe some of the data warehousing techniques, and position it within the larger framework of Business Intelligence. The primary topics discussed are:

- Data warehouse architecture
- Data mart architecture
- Other data mart-like structures
- Data models
- The high cost of data marts

© Copyright IBM Corp. 2005. All rights reserved.

2.1 Data warehousing

A data warehouse, in concept, is an area where data is collected and stored for the purpose of being analyzed. The defining characteristic of a data warehouse is its purpose. Most of the data collected comes from the operational systems developed to support the on-going day-to-day business operations of an enterprise. This type of data is typically referred to as *operational data*. Those systems used to collect the operational data are transaction oriented. That is, they were built to process the business transactions that occur in the enterprise. Typically being online systems, they then provide online transaction processing (OLTP).

OLTP systems are significantly different from the data warehousing systems. The data is formatted and organized to enable fast response for the transactions and the subsequent storage of their data. The defining characteristic of OLTP is the speed of processing the transactions, which are necessarily of short duration and access small volumes of data. The analysis transactions performed in data warehousing are typically of a long duration, as they are required to access and analyze huge volumes of data. Thus, if complex end-user analysis queries are allowed to run against the operational business transaction systems, they would no doubt impact the response-time requirements of those systems. Thus the need to separate the two data environments.

Also, to analyze these huge volumes of data efficiently requires that the data be organized much differently than the OLTP data. Thus they are two separate and distinct sets of data used for their separate and distinct purposes. The data in a data warehouse is typically referred to as *informational data*. The systems used to perform the analytical processing of the informational data are also typically online, so are referred to as online analytical processing systems (OLAP).

The original business rationale for data warehousing is well known. It is to provide a clean, stable, consistent, usable, and understandable source of business data that is organized for analysis. That is, the operational data from the business processes needed to be transformed to a format and structure that could yield useful business information. To satisfy that need, requires an architecture based solution.

Although there are variants in the architectures used in business, most are quite similar and typically designed with multiple layers. This enables the separation of the operational and the informational data, and provides the mechanisms clean, transform, and enhance the data as it moves across the layers from the operational to the informational environment. Having this informational environment as a source of clean, high-quality of data is invaluable for the enterprise decision makers. And, it did support the enterprise. Thus we refer to it as an Enterprise Data Warehouse (EDW).

2.1.1 Information environment

The enterprise information environments are experiencing significant change, which is leading to a number of new trends. For example, everyone today wants more current information and wants it faster. Weekly and daily reports rarely meet the requirement. Users are demanding current, up-to-the-minute results as the evolution towards the goal of real-time business intelligence continues.

This type of requirement can seldom be met in an environment with independent data marts, thus another reason for the movement towards data mart consolidation. With that movement comes the requirement for a more dynamic, fluid, information environment. This is depicted in Figure 2-1, and we refer to it as the information pyramid.

Figure 2-1 Information pyramid

IT has traditionally seen these as separate layers, requiring data to be copied from one level to another. However, they should be seen as different views of the same information, with different characteristics, required by users needing to do a specific job. To emphasize that, we have labeled them as floors, rather than layers of information.

The previously mentioned approach, to move and copy the data between the floors (and typically from the lower to the higher floors) is no longer the only approach possible. There are a number of approaches that enable integration of the data in the enterprise, and there are tools that enable those approaches. At IBM, information integration implies the result, which is integrated information, not the approach.

We have stated that the data on each floor has different characteristics, such as volume, structure, and access method. Now we can choose how best to physically instantiate the floors. For example, we can decide if the best technology solution is to build the floors separately or to build the floors together in a single environment. An exception is floor zero, which, for some time, will remain separate. For example, an OLTP system may reside in another enterprise, or another country. Though separate, we still can have access to the data and can move it into our data warehouse environment.

Floors one to five of the information pyramid can be mapped to the layers in an existing data warehouse architecture. However, this should only be used to supplement understanding and subsequent migration of the data. The preferred view is one of an integrated enterprise source of data for decision making — and, a view that is current, or real-time.

2.1.2 Real-time business intelligence

To provide the information that management requires demands shorter data update cycles. Taken to one end of the update spectrum, we see the requirement of continuous real-time updates. However, this brings with it a number of concerns and considerations to be addressed.

One concern is that supplying a data warehouse with fresh real-time data can be very expensive. Also, some data cannot, or need not, be kept in the data warehouse, even though it may be of critical value. This may be due to its size, structure, or overall enterprise usage.

However, there are additional benefits to be gained from supplying real-time data to the data warehouse. For example, it enables you to spread that update workload throughout the day and avoid a peak workload by processing it overnight. This may, in fact, actually result in less overall server resource requirements which could reduce costs and make it easier to meet service level agreements.

To address the business requirement for real-time data, enterprises need additional methods of integrating data and delivering information without necessarily requiring all data to be stored in the data warehouse. Current information integration approaches must, therefore, be extended to provide a

common infrastructure that not only supports centralized and local access to information using data warehousing, but also distributed access to other types of remote data from within the same infrastructure. Such an infrastructure should make data location and format transparent to the user or to the application. This new approach to information integration is a natural and logical extension to the current approaches to data warehousing.

For example, until you can achieve real-time updates to the data warehouse, there will be some period of latency encountered. To satisfy real-time requirements during the interim, you may want to consider data federation. This can enable real-time dynamic access to the data warehouse as well as other real-time data sources — with no latency. This is depicted in Figure 2-2.

Figure 2-2 Approaches to real-time access

In general, doing almost anything in real-time can be relatively expensive. Thus the search for a solution that is *close enough to real-time* to satisfy requirements. We call this *near real-time business intelligence.* That term is a bit more flexible, and able to cover a broad spectrum of requirements. And yet it conveys many of the same notions. For example, it still implies very current information but leaves flexibility in the time frame. For example, the first implementation may instantiate the data in 2 hours. Perhaps later we can reduce that time to 30 minutes.

To summarize, near real-time business intelligence is about having access to information from business actions as soon after the fact as is justifiable, based on the requirements. This will enable access to the data for analysis and input to the management business decision-making process soon enough to satisfy the requirement.

Archived

The implementation of near real-time BI involves the integration of a number of activities. These activities are required in any data warehousing or BI implementation, but now we have elevated the importance of the element of time. The traditional activity categories of *getting data in,* and *getting data out,* of the data warehouse are still valid. But, now they are ongoing continuous processes rather than a set of steps performed in a particular sequence.

For more information on this subject, please refer to the IBM Redbook, *Preparing for DB2 Near-Realtime Business Intelligence*, SG24-6071.

2.1.3 An architecture

A data warehouse is a specific construct developed to enable data analysis. As such, it requires an architecture so that an implementation would provide the required capabilities. We show a high level view of this type of architecture in Figure 2-3.

Figure 2-3 Data warehouse - three layer architecture

This data warehouse architecture can be expanded to multiple layers to enable multiple business views to be built on the consistent information base provided by the data warehouse. This statement requires a little explanation.

Archived

Operational business transaction systems have different views of the world, defined at different points in time and for varying purposes. The definition of *customer* in one system, for example, may be different from that in another. The data in operational systems may overlap and may be inconsistent. To provide a consistent overall view of the business, the first step is to reconcile the basic operational systems data. This reconciled data and its history is typically stored in the data warehouse in a largely normalized form. Although this data may then be consistent, it may still not be in a form required by the business or in a form that can deliver the best performance. One approach to address these requirements is to use another layer in the data architecture, the data mart layer. Here, the reconciled data is further transformed into information that supports specific end-user needs for different views of the business that can be easily and speedily queried - and highly available.

One trade-off in this multi-layer architecture is the introduction of a degree of latency between data arriving in the operational systems and its appearance in the data marts. In the past, this was less important to most companies. In fact, many organizations have been happy to achieve a one-day latency that this architecture can easily provide, as opposed to the multi-week reconciliation time frames they often faced in the past. However, in the fast-moving and highly competitive environment of today, even that is often no longer sufficient.

The prime requirement today is driven by such initiatives as business performance management (BPM). To enable such initiatives, requires more, and more current, information for decision-making. Thus the trend is now towards real-time data warehousing. This, in turn, has fostered the emergence of an expanded area of focus on data analysis, which is called real-time business intelligence (BI).

The key infrastructure of BI is, of course, the data warehouse environment, and the establishment of an architected solution. IBM has such an architecture, and it is called the BI Reference Architecture, as depicted in Figure 2-4. Such an architecture enables you to build a robust and powerful environment to support the informational needs of the enterprise.

Figure 2-4 BI Reference Architecture

2.1.4 Data warehousing implementations

In this section we describe some of the differing types of data warehousing implementations. They each have can satisfy the basic requirements of data warehousing, and thus provide flexibility to handle the requirements in the differing types of enterprise. We provide a high-level description of the types of data warehousing implementations, as depicted in Figure 2-5.

- **Centralized:** This type of data warehouse is characterized as having all the data in a central environment, under central management. However, centralization does not necessarily imply that all the data is in one location or in one common systems environment. That is, it is centralized, but *logically* centralized rather than physically centralized. When this is the case, by design, it then may be referred to as a hub and spoke implementation. The key point is that the environment is managed as a single integrated entity.
- **Hub and Spoke data warehouse:** This typically represents one type of distributed implementation. It implies a central data warehouse, which is the hub, and distributed implementations of data marts, which are the spokes. Here again, the key is the environment is managed as a single integrated entity.

- Distributed data warehouse: in this implementation the data warehouse itself is distributed, whether with or without data marts. That can imply two different implementations. As examples:
  - The data warehouse can reside in multiple hardware and software environments. The key is that the multiple instances conform to the same data model, and are managed as a single entity.
  - The data warehouse can reside in multiple hardware and software environments, but as separate and independent entities. In this case they will typically not conform to a single data model and may be managed independently.
- **Virtual data warehouse:** This type of implementation is at one end of the spectrum of data warehousing definitions. It exists when the desire is not to move, integrate, or consolidate the enterprise data. Data from multiple, even heterogeneous, data sources is accessed for analysis — but not stored as a physical data source. Consideration must be given to how any transformation requirements are addressed. There will also be issues with data concurrency and the repeatability of any analysis because the data is not time variant and not stored. Since this all happens in real-time, there must also be careful considerations regarding performance expectations.

As you can see, there are a number of choices in how to implement a data warehousing environment. There are costs and considerations with each. Your focus should be on the creation of a source on valid, integrated, consistent, stable, and managed source of data for analysis. It is only in this way that you will receive the value and real benefits that are inherent with data warehousing.

Archived

Figure 2-5 Types of data warehouse environments

Performance and scalability requirements

Performance and scalability are important attributes of an enterprise data warehouse. Therefore it is very important that the EDW achieves all performance requirements. Typically the requirements are presented in a service level agreement (SLA) that must be met to satisfy the enterprise.

Scalability is equally as important to satisfy the growth of the data warehouse. This implies growth in the number of users as well as the growth in volumes of data to be collected, stored, managed, and analyzed. Although this can somewhat be managed by addition of hardware capacity, there is also a requirement that the architecture be flexible and scalable to handle these growing volumes as well.

2.2 Advent of the data mart

A data mart is a construct that evolved from the concepts of data warehousing. The implementation of a data warehousing environment can be a significant undertaking, and is typically developed over a period of time. Many departments and business areas were anxious to get the benefits of data warehousing, and reluctant to wait for the natural evolution.

To satisfy the needs of such departments, along came the concept of a data mart — or, simplistically put, a small data warehouse built to satisfy the needs of a particular department or business area, rather than the entire enterprise. Often the data mart was developed by resources external to IT, and paid for by the implementing department or business area, to enable a faster implementation.

The data mart typically contains a subset of corporate data that is of value to a specific business unit, department, or set of users. This subset consists of historical, summarized, and possibly detailed data captured from transaction processing systems, or from an enterprise data warehouse. It is important to realize that a data mart is defined by the functional scope of its users, and not by the size of the data mart database. Most data marts today involve less than 100 GB of data; some are larger, however, and it is expected that as data mart usage increases, they will continue to increase in size.

The primary purpose of a data mart can be summarized as follows:

- Provides fast access to information for specific analytical needs
- Controls end user access to the information
- Represents the end user view and data interface to the data warehouse
- Creates a multi-dimensional view of data for enhanced analysis
- Offers multiple slice-and-dice capabilities for detailed data analysis
- Stores pre-aggregated information for faster response times

2.2.1 Types of data marts

Basically, there are two types of data marts:

- **Dependent:** These data marts contain data that has been directly extracted from the data warehouse. Therefore, the data is integrated, and is consistent with the data in the data warehouse.
- **Independent:** These data marts are stand-alone, and are populated with data from outside the data warehouse. Therefore, the data is not integrated, and is not consistent with the data warehouse. Often the data is extracted from either an application, an OLTP database, or perhaps from an operational data store (ODS).

Many implementations have a combination of both types of data marts. In the topic of data mart consolidation, we are particularly interested in consolidating the data in the independent data marts into the enterprise data warehouse. Then, of course, hopefully eliminating the independent data mart, along with all the costs and resource requirements for supporting it. Figure 2-6 shows a high level overview of a data warehousing architecture with data marts.

Figure 2-6 Data warehousing architecture - with data marts

As you can see in Figure 2-6, there are a number of options for architecting a data mart. As examples:

- Data can come directly from one or more of the databases in the operational systems, with few or no changes to the data in format or structure. As such, this limits the types and scope of analysis that can be performed. For example, you can see that in this option, there may be no interaction with the metadata. This can result in data consistency issues.
- Data can be extracted from the operational systems and transformed to provide a cleansed and enhanced set of data to be loaded into the data mart by passing through an ETL process. Although the data is enhanced, it will not be consistent with, or in sync with, data from the data warehouse.
- Bypassing the data warehouse leads to the creation of an independent data mart. That is, it is not consistent, at any level, with the data in the data warehouse. This is another issue impacting the credibility of reporting.
- Cleansed and transformed operational data flows into the data warehouse. From there, dependent data marts can be created, or updated. It is key that updates to the data marts be made during the update cycle of the data warehouse to maintain consistency between them. This is also a major consideration and design point, as you move to a real-time environment. At that time it would be good to revisit the requirements for the data mart, to see if they are still valid.

2.3 Other analytic structures

However, there are many other data structures that are used for data analysis, and use differing implementation techniques. These fall in a category we are simply calling analytic structures. However, based on their purpose, they could be thought of as data marts.

2.3.1 Summary tables, MQTs, and MDC

Summary tables, MQTs, and MDC are approaches that can be used to improve query response time performance. Summary tables contain sets of data that have been summarized from a detailed data source. This saves storage space and enables fast query response when summary data is sufficient. However, many systems want to use both summary and detailed data. For example, many still want to look at the detail data, and those who may not, still do not want to impact their response time by recalculating from the detail on all their queries.

A materialized query table (MQT) is a table whose definition is based on the result of a query, and whose data is in the form of precomputed results that are taken from one or more tables on which the materialized query table definition is based. Multidimensional Clustering (MDC) is a method for clustering data in tables along multiple dimensions. That is, the data can be clustered along more than one key.

The summary tables can be created to hold the results of simple queries, or a collection of joins involving multiple tables. These are powerful ways to improve performance and minimize response times for complex queries, particularly those associated with analyzing large amounts of data. They are also most applicable to those queries that are executed on a very frequent basis.

For those applications or queries that do not need the detailed level of data, performance is significantly improved by having a summary table already created. Then each query or application does not have to spend time generating the results table each time that information is required. In addition, the summary table can hold other derived results, that again keep the queries and applications from calculating the derivations every time the query or application is executed.

As an example, aggregates or summaries of data in a set of base tables can be created in advance and stored in the database as Materialized Query Tables (MQTs), as depicted in Figure 2-7. Then, when queries are executed, the DB2 optimizer recognizes that the query requires an aggregation. It then looks to see if it has a relevant MQT available to use rather than developing the query result. If the MQT does exist, then the optimizer can rewrite the query to use the MQT rather than the base data.

As the MQT is a precomputed summary and/or filtered subset of the base data, it tends to be much smaller in size than the base tables from which it was derived. As such, significant performance gains can be made from using the MQT. If the resulting joins or aggregates can be generated once and used many times, then this clearly saves processing time and therefore improves overall throughput and cost performance. As we know, joining tables can be even more costly than aggregating rows within a single table.

Figure 2-7 Materialized view

One of the things to consider, when using a summary table or MQT, is the update frequency requirement. There are two approaches to refresh MQTs, deferred or immediate:

- In the deferred refresh approach, the contents of the MQT are not kept in sync automatically with the base tables when they are updated. In such cases, there may be a latency between the contents of the MQT and the contents in the base tables. **`REFRESH DEFERRED`** tables can be updated via the **`REFRESH`** table command with either a full refresh (**`NOT INCREMENTAL`**) option, or an incremental (**`INCREMENTAL`**) option. An overview of the deferred refresh mechanism is depicted in Figure 2-8.
- In the immediate refresh approach, the contents of the MQT are always kept in sync with the base tables. An update to an underlying base table is immediately reflected in the MQT as part of the update processing. Other users can see these changes after the unit of work has completed on a commit. There is no latency between the contents of the MQT and the contents in the base tables.

Archived

Figure 2-8 Deferred refresh mechanism

DB2 supports MQTs that are either maintained by the system, which is the default, or maintained by the user, as follows:

- **MAINTAINED BY SYSTEM** (default):

  In this case, DB2 ensures that the MQTs are updated when the base tables on which they are created get updated. Such MQTs may be defined as either **REFRESH IMMEDIATE**, or **REFRESH DEFERRED**. If the **REFRESH DEFERRED** option is chosen, then either the **INCREMENTAL** or **NON INCREMENTAL** refresh option can be chosen during refresh.

- **MAINTAINED BY USER:**

  In this case, it is up to the user to maintain the MQTs whenever changes occur to the base tables. Such MQTs *must be* defined with the **REFRESH DEFERRED** option. Even though the **REFRESH DEFERRED** option is required, unlike **MAINTAINED BY SYSTEM**, the **INCREMENTAL** or **NON INCREMENTAL** option does not apply to such MQTs, since DB2 does not maintain such MQTs. Two possible scenarios where such MQTs could be defined are as follows:

  – For efficiency reasons, when users are convinced that they can implement MQTs maintenance far more efficiently than the mechanisms used by DB2 (for example, the user has high performance tools for rapid extraction of data from base tables, and loading the extracted data into the MQTs).

  – For leveraging existing user maintained summaries, where the user wants DB2 to automatically consider them for optimization for new ad hoc queries being executed against the base tables.

Archived

Multidimensional Clustering (MDC) provides an elegant method for clustering data in tables along multiple dimensions in a flexible, continuous, and automatic way. MDC can significantly improve query performance, in addition to significantly reducing the overhead of data maintenance operations, such as reorganization, and index maintenance operations during insert, update, and delete operations. MDC is primarily intended for data warehousing and large database environments, and it can also be used in online transaction processing (OLTP) environments.

Regular tables have indexes that are record-based. Any clustering of the indexes is restricted to a single dimension. The clustering of the index is not guaranteed; the clustering degrades once the page free space is used up. A periodic reorganization of the index is required.

MDC tables have indexes that are block-based. The blocks are defined clustering dimensions. Data is clustered across multiple dimensions. The clustering of the index is guaranteed. Clustering is automatically and dynamically maintained over time. No periodic reorganization is necessary to re-cluster the index. Queries against clustering dimensions only carry out input and output actions that are necessary to access the selected data. The performance of queries is improved.

MDC allows data to be independently clustered along more than one key. This is unlike regular tables, which can have their data clustered only according to a single key. Therefore, scans of an MDC table via any of the dimension indexes are equally efficient, unlike a regular table where only a scan of the data via the clustering index is likely to be efficient. MDC has wide applicability in fact tables in star schema implementations, and it is therefore quite common to see the word dimensions used rather than keys.

So how does this impact the data mart environment? Sometimes data marts have been created to separate out the data so that it can be organized to allow more efficient access, and thus faster query response times. Using MDC within the consolidated data warehouse may provide those benefits without having to physically separate the data onto a data mart.

Figure 2-9 highlights the differences between clustering in a regular table and multidimensional clustering in an MDC table.

Figure 2-9 Traditional RID clustering and Multidimensional clustering

For more details on summary tables, MQTs, and MDC, please refer to the IBM Redbook, *DB2 UDB ESE V8 non-DPF Performance Guide for High Performance OLTP and BI*, SG24-6432.

2.3.2 Online analytical processing

Online analytical processing (OLAP) is a key technology in data warehousing. The OLAP functionality is characterized by dynamic multi-dimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities. In simpler terms, this means that OLAP is used because it provides an easy to use and intuitive interface for business users and can process the data very efficiently.

The following list shows some of the functional capabilities delivered with OLAP:

- Calculations and modeling applied across dimensions, through hierarchies, and/or across members
- Trend analysis over sequential time periods
- Slicing subsets for on-screen viewing
- Drill-down to deeper levels of consolidation

- Reach-through to underlying detail data
- Rotation to new dimensional comparisons in the viewing area

OLAP is implemented in a multi-user client/server mode and offers consistently rapid responses to queries, regardless of database size and complexity. OLAP helps the user synthesize enterprise information through comparative, personalized viewing, as well as through analysis of historical and projected data in various “what-if” data model scenarios. This is achieved through use of the functionality previously listed.

The term OLAP is a general term that encompasses a number of different implementations of the technologies. And, there are several types of OLAP. Let us take a look at some of the most common implementations that are available:

- **MOLAP:** Is a term for Multidimensional OLAP. Here, the database is stored in a special, typically proprietary, structure that is optimized (through pre-calculation) for very fast query response time and multidimensional analysis. However, it can take significant time for the pre-calculation and load of the data, and significant space for storing the calculated values. This implementation also has limitations when it comes to scalability, and may not allow updating.
- **ROLAP:** Stands for Relational OLAP. Here, the database model is also multidimensional as with MOLAP. But, a standard relational database is used, and the data model can be a star schema or a snowflake schema. This implementation will still provide fast query response time, but that is largely governed by the complexity of the SQL used as well as the number and size of the tables that must be joined to satisfy the query. A primary benefit with this implementation is the significant scalability achieved as it is housed on a standard relational database.
- **HOLAP:** Enables a hybrid version of OLAP. As the name implies, it is a hybrid of ROLAP and MOLAP. A HOLAP database can be thought of as a virtual database whereby the higher levels of the database are implemented as MOLAP and the lower levels of the database as ROLAP.

This is not necessarily a case of deciding which is best, it is more about which will best satisfy your particular requirements for OLAP technology; perhaps a combination of all three.

2.3.3 Cube Views

Cube Views™ is a DB2 mechanism used to improve OLAP scalability performance, and to allow DB2 to work directly on OLAP data. With Cube Views, information about your relational data is stored in metadata objects that provide a new perspective from which to understand your data.

Note: DB2 Cube Views is included in the DB2 Data Warehouse Edition (DWE); otherwise, it must be purchased separately.

DB2 Cube Views makes the relational database an effective platform for OLAP by giving it an OLAP-ready structure and by using metadata to label and index the data it stores. This delivers more powerful and cost-effective analysis and reporting on demand, both because the database can perform many OLAP tasks on its own, and because it speeds up data sharing with OLAP tools. DB2 Cube Views makes the metadata about dimensions, hierarchies, attributes, and facts, available to all tools and applications working with the database. By handling OLAP data and metadata directly in one environment this makes development more efficient.

For more information about DB2 Cube Views, refer to the IBM Redbook, *DB2 Cube Views: A Primer,* SG24-7002.

2.3.4 Spreadsheets

One of the most widely used tools for analysis is the spreadsheet. It is a very flexible and powerful tool, and therefore can be found in almost every enterprise around the world. This is a good-news and bad-news situation: good because it empowers users to be more self-sufficient; bad because it can result in a multitude of independent (non-integrated and non-shared) data sources that exist in any enterprise.

Here are a few examples of what spreadsheets are used for:

- Finance reports, such as a price list or inventory
- Analytic and mathematical functions
- Statistic Process Control, which is often used in manufacturing to monitor and control quality

This proliferation of spreadsheets (data sources) exposes the enterprise to all the issues that surround data quality, data consistency, data currency, and even data validity. They face many of the same issues we have been discussing relative to data marts, such as these:

- The spreadsheet data are often not consistent with the operational data sources, the ODS, or the data warehouse.
- Very few people typically have knowledge about the content, or even the source or whereabouts, of the data.

- All this spreadsheet data resides on a multitude of hardware platforms, in numerous operating environments, and consists of a multitude of data definitions and data types.
- Reports of all types and configurations are developed from these spreadsheets and can be a challenge to understand. That is, there is typically no consistency or any reporting standards being observed.
- Since spreadsheet data can be amended so easily, it is difficult to ensure that the data has not been tampered with or distorted.

Organizations need to get control of their data so they can manage it and be confident that it is valid. There are a number of ways to approach this, and they are discussed in 4.2, “Approaches to consolidation” on page 71.

The proliferation of all of these types of data marts adds to operating costs in any enterprise. For more information on this topic, see 3.2.1, “High cost of data marts”.

2.4 Data warehousing techniques

To round out our discussion of data warehousing, in this section we describe a few examples of data warehousing techniques that are key to any implementation:

- Operational data stores for real-time operational analytics
- Data federation and integration
- Data replication

Each of these techniques provides value and capability that can be considered based on the requirements of the implementation.

2.4.1 Operational data stores

An operational data store (ODS) is an environment where data from the various operational databases is integrated and stored. It is much like a data warehouse, but is typically aimed at providing real-time analytics for a particular subject area. And it is usually concerned with a much shorter time horizon than the data warehouse.

The purpose is to provide the end user community with an integrated view of the operational data. It enables the user to address operational challenges that span over more than one business function or area. In addition, the data is cleansed and transformed to ensure that it is a good source for input to the data warehousing environment.

The principal differentiators are the update frequency and the direct update paths from applications, compared to the controlled update path of the data warehouse or data marts.

The following characteristics apply to an ODS:

- **Subject oriented:** The ODS may be designed not only to fulfill the requirements of the major subject areas of a corporation, but also for a specific function or application. For example, a risk management application may need to have a holistic view of a customer.
- **Integrated:** Existing systems push their detailed data through a process of transformation into the ODS. This leads to an integrated, corporate-wide understanding of data. This transformation process is quite similar to the transformation process of a data warehouse. When dealing with multiple existing systems, there are often data identification and conformance issues that must be addressed; for example, customer identification — that is, determining which codes are used to describe the gender of a person.
- **Near current data delivery:** The data in the ODS is continuously being updated. Changing data in this manner requires a high frequency update. Changes made to the underlying existing systems must reach the ODS quickly, to maintain a current view of the status of the operational area. Some data needs to be updated immediately, while other data need only be part of the planned periodic (perhaps hourly or daily) update. Thus, the ODS typically requires both high frequency and high velocity update.

Frequency and velocity:

Frequency describes how often the ODS is updated. Quite possibly the updates come from completely different existing systems that use distinct population processes. It also takes into account the volume of updates that are occurring to the base operational tables.

Velocity is the speed with which an update must take place. It is determined using the point in time a existing system change occurs and the point in time that the change must be reflected in the ODS.

- **Current data:** An ODS reflects the status of its underlying source data systems. The data is typically kept quite up-to-date. In this book we follow the architectural principle that the ODS should contain little or no history. Typically there is sufficient data to show the current position, and the context of the current position. In practice, 30 to 90 days of history would be typical; for example, a recent history of transactions to assist a call center operator. Of course, if there are overriding business requirements, this principle may be altered. If your ODS must contain history, you should ensure that you have a complete understanding of what history is required and why, and you must consider all the ramifications of keeping that data; for example, the sizings, archiving, and performance requirements.
- **Detailed:** The ODS is designed to serve the operational community, and therefore is kept at a detailed level. Of course, the definition of "detailed" depends on the business requirements for the ODS. For example, the granularity of data may or may not be the same as in the source operational system. That is, for an individual source system, the balance of every single account is important. But for the clerk working with the ODS, only the summarized balance of all accounts may be important.

Over time, the ODS may become the "master". Note that a particular challenge is that, while the ODS is the master for some future data and existing data, for others it is the existing systems. While this is going on, if data is updated in the ODS, those updates may need to be propagated back into the existing systems. The need to synchronize these updates made to the ODS and propagated into the existing systems is a major design consideration.

Once we have gotten through the transition and the ODS is the master used by all processes, this is no longer an issue. However, during the transition, this needs careful consideration. When we consolidate, we are by definition going through a process of transition. This is a period during which old and new data structures will co-exist, and needs to be considered in the planning.

To put this discussion in a better perspective, we have depicted a typical ODS architecture in Figure 2-10.

Figure 2-10 ODS architecture

2.4.2 Data federation and integration

The fast moving business climate demands increasingly fast analysis of large amounts of data from disparate sources. These are note only the traditional application sources such as relational databases, but also sources such as extensible markup language (XML) documents, text documents, scanned images, video clips, news feeds, Web content, e-mail, analytical cubes, and special-purpose data stores.

To consolidate this heterogeneous data, there are two primary alternative approaches that can be of help. As examples, consider data federation and data integration. These two approaches are similar and interrelated, but with some subtle differences. For example, integration more typically involves physical consolidation of the data. That is, the data may be transformed, converted, and/or enhanced to maintain the interrelationship. Federation typically implies a more temporary integration of the data. For example, a query is executed that requires data to be accessed, and perhaps joined, from multiple heterogeneous environments. The query completes, but the original data is still resident in the original source environments and the joined result may not be instantiated.

These alternatives involve the process of defining access to heterogeneous data sources. In addition to access, there are other capabilities that are very powerful. For example, the data in different source environments can be joined with a single SQL statement. Let us look at federation and integration.

Federation provides the facility to create a single-site image of disparate data by combining information from multiple heterogeneous databases located in multiple locations. The heterogeneous data sources may be located locally or remotely, and access is optimized using a middle ware query processor. A federated server is used, and is an abstraction layer that hides complexity and idiosyncrasies associated with the implementation of the heterogeneous data sources. The federated server works behind the scenes to provide access to this disparate data transparently and efficiently. Such work includes automatic data transformations, API conversions, functional compensation, and optimization of the data access operations. Federation also allows the presentation of client data through a unified user interface. The IBM WebSphere Information Integrator (WII), WebSphere Information Integrator for Content, and DB2 UDB, are the product offerings that can provide heterogeneous federation capabilities.

So, federation is the ability to access multiple heterogeneous data sources in multiple heterogeneous environments, *as if they were resident in your local environment*. For example, a federated database system allows you to query, join, and manipulate data located on multiple other servers. The data can be in multiple heterogeneous data management systems, such as Oracle, Sybase, Microsoft SQL Server, Informix®, and Teradata, or it can be in other types of data stores such as a spreadsheet, Web site, or files. You can refer to multiple database managers, or an individual database, in a single SQL statement. And, the data can be accessed directly or through database views. The IBM product that performs this type of functionality is the WebSphere Information Integrator (WebSphere II).

Integration more typically refers to the physical consolidation of data from multiple heterogeneous environments. That is, multiple heterogeneous sources of data are brought together to reside as a single data source. To accomplish this, the data types and elements must be standardized and consistent. These process actions are implemented and the result is an integrated source of data. For more details on the attributes of these technologies, refer to Table 1-1 on page 6.

An example

Figure 2-11 depicts how data from two different sources, Source 1 and Source 2, is accessed via the database server in order to present an integrated view of the data to the client. In this example, the database server would need to have an application that would enable the external data sources, Data Source 1 and Data Source 2, to be accessible from the database server. Then, an SQL query,

defined by the client, could access the data from Data Source 1 and Data Source 2, and it could even be joined with tables already residing on the database server. The queries executed from the client produces an integrated view of the data from the database server, Data Source 1, and Data Source 2.

Figure 2-11 Data source access

An example with WebSphere Information Integrator

IBM offers tools and technologies that can be used for data federation and integration of heterogeneous data sources, in a DB2 database environment. One of those tools is the WebSphere II. If access only to the DB2 family of databases or Informix is required, then WebSphere II is not required. WebSphere II is a product designed to provide access to heterogeneous data sources and consolidate it in a DB2 database. To access the heterogeneous data sources, WebSphere II uses *wrappers*. These wrappers are provided by WebSphere II enable the connectivity to heterogeneous data sources. A library of wrappers is provided to enable access to a number of data sources, such as Oracle, Sybase, SQL/Server, Teradata, generic ODBC, Flat files, XML documents, and MS-Excel files. It contains information about the remote data source characteristics, and it understands their capabilities. By way of federation and integration the client views a combined result set from all the source systems. This process is depicted in Figure 2-12.

Archived

Figure 2-12 Federation with WebSphere II

DB2 UDB is the foundation for WebSphere II, and federates data from DB2 UDB on any platform, and data from almost any heterogeneous data source. For example, it can also federate data from non-DB2 sources, such as Oracle, as well as non-relational sources such as XML. While WebSphere II and DB2 UDB have an SQL interface, WebSphere II for Content uses IBM Content Manager interfaces and object APIs as its federation interface. Being able to get a unified data view of the data throughout the enterprise can be accomplished using SQL and XML client access to federated databases. Information integration enables federated access to unstructured data (such as e-mail, scanned images, and audio/video files) that is stored in repositories, file systems, and applications. This facilitates efficient storage, versioning, check-in/check-out, and searching of content data. An additional benefit of WebSphere II for content management is the reduction in operational costs.

For more information about WebSphere II refer to the DB2 documentation or to the redbook *Getting Started on Integrating Your Information,* SG24-6892.

Note: The information integration product has been renamed from the DB2 Information Integrator (DB2II) to the WebSphere Information Integrator (WebSphere II). As the referenced redbook was written prior to this renaming, you may still find references to DB2II. Be aware that it is the same product.

2.4.3 Federated access to real-time data

In a traditional data warehousing environment, a query or report may require up-to-the-minute data as well consolidated historical and analytical data. To accomplish this, the real-time data may be fed continuously into the data warehouse from the operational systems, possibly through an operational data store (ODS). This is depicted in Figure 2-13. There are some things to consider with this approach. For one, not only must significant quantities of data be stored in the data warehouse, but also the ETL environment must be capable of supporting sustained continuous processing. Data federation can help meet this requirement by providing access to a combination of live operational business transaction data and the historical and analytical data already resident in a data warehouse.

Figure 2-13 Federated access to real-time data

With federated data access, when an end-user query is run, a request for specific operational data can be sent to the appropriate operational system and the result combined with the information retrieved from the data warehouse. It is important to note that the query sent to the operational system should be simple and have few result records. That is, it should be the type of query that the operational system was designed to handle efficiently. This way, any performance impact on the operational system and network is minimized.

In an IBM environment, enterprise information integration (EII) is supported using WebSphere II. Operational data sources that can be accessed using WebSphere II include those based on DB2 UDB, third-party relational DBMS, and non-relational databases, as well as IBM WebSphere MQ queues and Web services.

IBM WebSphere II federated queries are coded using standard SQL statements. The use of SQL allows the product to be used transparently with existing

business intelligence (BI) tools. This means that these tools can now be used to access not only local data warehouse information, but also remote relational and non-relational data. The use of SQL protects the business investment in existing tools, and leverages existing IT developer skills and SQL expertise.

2.4.4 Federated access to multiple data warehouses

Another area of data warehousing where a federated data approach can be of benefit is when multiple data warehouses and data marts exist in an organization. This is depicted in Figure 2-14.

Typically, a data warehousing environment would consist of a single (logical, not necessarily physical) enterprise data warehouse, possibly with multiple underlying *dependent* data marts. This would be a preferred approach. In reality, however, this is not the case in many companies. Mergers, acquisitions, uncoordinated investments by different business units, and the use of application packages, often lead to multiple data warehouses and stand-alone *independent* data marts.

In such an uncoordinated data warehousing environment, as multiple data warehouses or data marts are added, it becomes increasingly more difficult to manage the quality, redundancy, consistency, and currency of the data. Typically, a significant amount of data is duplicated across the environment. This is a costly environment, because of such things as the additional costs of creating and maintaining the redundant data, as well as the added complexity it adds to an already complex environment. A better approach is to consolidate and rationalize the multiple data warehouses. This can be expensive and time-consuming, but typically will result in a much better, more consistent, higher quality, and less expensive environment - and a worthwhile investment.

A federated data approach can be used to simplify an uncoordinated data warehousing environment through the use of business views that present only the data needed. This is depicted in Figure 2-14. While this approach may not fully resolve the differences between the data models of the various data warehouses, it does provides a lower-cost and more simplified data access.

This approach to accessing disparate information can evolve over time, and complement the data mart consolidation efforts. In this way, the inevitable inconsistencies in meaning, or content, that have arisen among different data warehouses and marts are incrementally removed. This will enable an easier consolidation effort.

It may be that some data on existing systems will need to be re-engineered to enable it to be joined to other data. But, this should be a much easier exercise than scrapping the system and replacing it completely.

Figure 2-14 Federated access to data marts and data warehouse

2.4.5 When to use data federation

It is important to re-emphasize that IBM does not recommend eliminating data warehouses and data marts, and changing BI query, reporting and analytical workloads in favor of data federation — also known as virtual data warehousing. Virtual data warehousing approaches have been tried many times before, and most have failed to deliver the value and capabilities that business users require. Data federation does not replace data warehousing — it complements and extends it.

Data federation is a powerful capability, but it is essential to understand the limitations and issues involved. One issue to consider is that federated queries may need access to remote data sources, such as an operational business transaction system. As previously discussed, there are potential impacts of complex query processing on the performance of operational applications. With the federated data approach, this impact can be reduced by sending only simple and specific queries to an operational system. In this way performance issues can be predicted and managed.

Another consideration is how to logically and correctly relate data warehousing information to the data in operational and remote systems. This is a similar issue that must be addressed when designing the ETL processes for building a data warehouse. The same detailed analysis and understanding of the data sources and their relationships to the targets is required. Sometimes, it will be clear that a data relationship is too complex, or the source data quality too poor to allow federated access. Data federation does not, in any way, reduce the need for detailed modeling and analysis. You still need to exercise rigor in the design process, because of the real-time and on-line nature of any required data transformation or data cleanup. When there is significant data cleansing or

complex transformations required, data warehousing may be the preferred solution.

We have discussed both EII and ETL in this section. Each have their different functionality and role to play in data warehousing and data mart consolidation. EII is primarily suited for extracting and integrating data from heterogeneous sources. ETL also extracts data, but is primarily suited for then transforming and cleansing the data prior to loading it into a target database.

The following list describes some of the circumstances when data federation would be an appropriate approach to consider:

- **There is real-time or near real-time access to rapidly changing data.** Making copies of rapidly changing data can be costly, and there will always be some latency in the process. Through federation, the original data is accessed directly. However, the performance, security, availability and privacy aspects of accessing the original data must be considered.
- **Direct immediate write access to the original data is possible.** Working on a data copy is generally not advisable when there is a need to modify the data, as data integrity issues between the original data and the copy can occur. Even if a two-way data consolidation tool is available, complex two-phase locking schemes are required.
- **It is technically difficult to use copies of the source data.** When users require access to widely heterogeneous data and content, it may be difficult to bring all the structured and unstructured data together in a single local copy. Also, when source data has a very specialized structure, or has dependencies on other data sources, it may not be possible to make and query a local copy of the data.
- **The cost of copying the data exceeds that of accessing it remotely.** The performance impact and network costs associated with querying remote data sets must be compared with the network, storage and maintenance costs of storing multiple copies of data. In some cases, there will be a clear case for a federated data approach when:
  - Data volumes in the data sources are too large to justify copying it.
  - A very small or unpredictable percentage of the data is ever used.
  - Data has to be accessed from many remote and distributed locations.
- **It is illegal or forbidden to make copies of the source data.** Creating a local copy of source data that is controlled by another organization or that resides on the Internet may be impractical, due to security, privacy or licensing restrictions.
- **The users' needs are not known in advance.** Allowing users immediate and *ad hoc* access to needed data is an obvious argument in favor of data federation. Caution is required here, however, because of the potential for

users to create queries that give poor response times and negatively impact both source system and network performance. In addition, because of semantic inconsistencies across data stores within organizations, there is a risk that such queries would return incorrect answers.

2.4.6 Data replication

In simple terms, replication is the copying of data from one place to another. Data can be extracted by programs, transported to some other location, and then loaded at the receiving location. A more efficient alternative is to extract only the changes since the last processing cycle and then transport and apply those to the receiving location. When required, data may be filtered and transformed during replication. There may be other requirements for replication, such as time constraints.

In most cases, replication must not interfere with production applications and have minimal impact on performance. IBM has addressed this need with the DB2 data replication facility. Data is extracted from logs (SQL replication), so as not to interfere with the production applications.

Replication also supports incremental update (replicating only the changed data) of the replicas to maximize efficiency and minimize any impact on the production environment.

Businesses use replication for many reasons. In general, the business requirements can be categorized as:

- Distribution of data to other locations
- Consolidation of data from multiple locations
- Bidirectional exchange of data with other data sources
- Some variation or combination of the above

The WebSphere II Replication Edition provides two different solutions to replicate data from and to relational databases:

- SQL replication
- Queue (Q) replication

SQL replication

For replication among databases from multiple vendors, WebSphere II uses SQL-based replication architecture that maximizes flexibility and efficiency in managing scheduling, transformation, and distribution topologies. In SQL replication, WebSphere II captures changes using either log-based or trigger-based mechanism and inserts them into a relational staging table. An apply process asynchronously handles the updates to the target systems. WebSphere II is used extensively for populating data warehouses and data

marts, maintaining data consistency between disparate applications, or efficiently managing distribution and consolidation scenarios among headquarter and branch or retail configurations.

In addition, you can replicate data between heterogeneous relational data sources. As examples:

- DB2, IBM Informix Dynamic Server, Microsoft SQL Server, Oracle, Sybase SQL Server, and Sybase Adaptive Server Enterprises are supported as replication sources and targets.
- IBM Informix Extended Parallel Server and Teradata are supported as replication targets.

Queue (Q) replication

The IBM queue-based replication architecture offers low latency and high throughput replication with managed conflict detection and resolution. Changes are captured from the log and placed on a WebSphere message queue. The apply process retrieves the changes from the queue and applies them — in parallel — to the target system. Q replication is designed to support business continuity, workload distribution, and application integration scenarios.

For more information about the WebSphere II Replication Edition, please refer to the IBM Web page:

http://www-306.ibm.com/software/data/integration/replication.html

2.5 Data models

In this section, we give you a brief overview of the typical types of data models that are used when designing databases, data marts, and data warehouses. The models are based on different technologies that are meant to provide the type of data access support, organization, and performance, desired in a particular situation. The most common are:

- Star schema
- Snowflake schema
- Normalized — most commonly third normal form (3NF)

When selecting which type of data model to use, there are a number of considerations. Of course, performance is typically the first mentioned and is indeed an important one. However, it is not the only one. And it may not always be the most important one. For example, ease of understanding and navigating the data model can be key considerations. These considerations not only impact IT, but users as well. This is particularly true in data warehousing. The purpose of

a data warehouse is to enable easy analysis of the data. Here the importance of understanding the data model and data relationships is a primary consideration.

2.5.1 Star schema

The star schema has become a common term used to connote a dimensional model. It has become very popular with data marts and data warehouses because it can typically provide better query performance than the normalized model historically associated with a relational database. There are other issues and considerations, but performance is typically the most dominant. For example, ease of understanding is a major benefit.

A star schema is a significant departure from a normalized model. It consists of a typically large table of facts (known as a fact table), with a number of other tables containing descriptive data surrounding it, called dimensions. When it is drawn, it resembles the shape of a star, hence the name. This is depicted in Figure 2-15.

Figure 2-15 Star schema

The basic elements in a star schema model are:

- Facts
- Dimensions
- Measures (variables)

Fact

A fact is a collection of related data items, consisting of measures and context data. Each fact typically represents a business item, a business transaction, or an event that can be used in analyzing the business or business processes. For example, the columns might contain measures such as sales for a given product, for a given store, for a given time period.

Dimension

A dimension is a collection of members or units that describe the fact data from a particular point of view. In a diagram, a dimension is usually represented by an axis. In a dimensional model every data point in the fact table is associated with only one member from each of the multiple dimensions. The dimensions determine the contextual background for the facts. What defines the dimension tables is that they have a parent primary key relationship to a child foreign key in the fact table. The star schema is a subset of the database schema.

Measure

A measure is a numeric attribute of a fact, representing the performance or behavior of the business relative to the dimensions. The actual members are called variables. For example, measures are such things as the sales in money, the sales volume, and the quantity supplied. A measure is determined by combination of the members of the dimensions, and is located on facts.

2.5.2 Snowflake schema

Further normalization and expansion of the dimension tables in a star schema results in the implementation of a snowflake design.In other words, a dimension is said to be snowflaked when the low-cardinality columns in the dimension have been removed to separate normalized tables that then link back into the original dimension table. This is depicted in Figure 2-16.

Archived

Figure 2-16 Snowflake schema

As an example, we have expanded (snowflaked) the Product dimension in Figure 2-16 by removing the low-cardinality elements pertaining to Family, and putting them in a separate Family table. That separate table is linked to the Product dimension by an index entry (Family_id) in both tables. From the Product dimension table, the Family attributes in the related subset of rows are extracted, in this example the Family Intro_date. In addition, the keys of the hierarchy (Family_Family_id) are also included in the table.

When do you snowflake?

Snowflaking should be generally avoided in dimensional modeling, because it can slow down user queries, and it makes the model more complex.The disk space savings gained by normalizing the dimension tables typically are less than two percent of the total disk space needed for the overall dimensional schema.

However, snowflaking (shown in Figure 2-17) could perhaps be beneficial under situations such as the following:

- When two entities have data at different grain. As shown in Figure 2-17, the two tables “Employee” and “Country_Demographics” have different grains. The “Employee” table stores the employee information, whereas the “Country_Demographics” table stores the country demographics information.

Note: Grain defines the level of detail of information stored in a table (fact or dimension).

- When the two entities are most likely supplied by a different source system. It is most likely that the two tables “Employee” and “Country_Demographics” are being fed by two separate source systems. The likely source system for the “Employee” table could be an human resources application whereas the source for the “Country_Demographics” table could be a world health organization database.

Figure 2-17 A case for acceptable snowflaking

2.5.3 Normalization

The objective of normalization is to minimize redundancy by not having the same data stored in multiple tables. As a result, normalization can minimize any integrity issues because SQL updates then only need to be applied to a single table. However, queries, particularly those involving very large tables, that include a join of the data stored in multiple normalized tables may require additional processing to maintain expected performance.

Although data in normalized tables is a very pure form of data and minimizes redundancy, it can be a challenge for users to navigate. For example, if a user must navigate a data model that requires a 15-way join it may likely be more difficult and less intuitive than a star schema with standard and independent dimensions.

In general, the data in relational databases is stored in normalized form. Normalization basically involves splitting large tables of data into smaller and smaller tables, until you end up with tables where no column is functionally dependent on any other column, each row consists of a single primary key, and a set of totally independent attributes of the object that are identified by the primary key. This type of structure is said to be in **third normal form (3NF)**.

Definition: Third normal form (3NF):

A table is in third normal form if each non-key column is independent of other non-key columns, and is dependent only on the key.

Another much-used shorthand way of defining third normal form is “The Key, the Whole Key, and Nothing but the Key”.

Third normal form is strongly recommended for OLTP applications since data integrity requirements are stringent, and joins involving large numbers of rows are minimal. A sample normalized schema is shown in Figure 2-18. Data warehousing applications, on the other hand, are predominantly read only, and therefore typically can benefit from denormalization, which involves duplicating data in one or more tables to minimize or eliminate joins. In such cases, adequate controls must be put in place to ensure that the duplicated data is always consistent in all tables to avoid data integrity issues.

Figure 2-18 Normalized schema

Archived

Archived

3

Data marts: Reassessing the requirement

Data warehousing, as a concept, has been around for quite a number of years now. It developed from the need for more and better information for making business decisions. Today, almost every business in the world has some type of data warehousing implementation. It is a proven concept, with substantial and validated business payback. Business users have seen the benefits, and are, in general, anxious to have such capability as is delivered with data warehousing.

But....and there seems always to be a “but” when it comes to these types of initiatives. The “but” comes from the wide range of definitions and implementation approaches for data warehousing that exist. These have surfaced over the years from many sources.

The original concept centered around an enterprise, or centralized, data warehouse being the place where a clean, valid, consistent, and time-variant source of historical data would be kept to support decision-making. That data would come from all the data sources, from 1 to n, around the enterprise. This is depicted in Figure 3-1.

As you might imagine, this brought with it a variety of issues, and decisions to be made regarding such things as:

- What data should be in the data warehouse?
- How long should it be kept?

© Copyright IBM Corp. 2005. All rights reserved.

- How should it be organized?
- When should it be put into the data warehouse?
- When should it be archived?
- When formats should be used?
- Who will use this data?
- How will we access it?
- How much will it cost to build and use?
- How will the costs be apportioned (who will pay for it)?
- How long will it take?
- Who will control it?
- Who will get access to it first?

Figure 3-1 Enterprise data warehouse

It was commonly agreed upon that, at the enterprise level, data warehousing could be a very large initiative and take significant time, money, and resources to implement. All this played a part in contributing to a slower than desired implementation schedule. But having a critical need for information and the benefits of data warehousing, the user community wanted it now! Thus came the search for a faster way to get their needs met.

Many departments and business areas within the enterprise went about finding ways to build a data warehouse to satisfy their own particular needs. Thus came the advent of the data mart. In simple terms, a data mart is defined to be a small data warehouse designed around the requirements of a specific department or business area. Since there was no interrelationship with a data warehouse, these were know as *independent* data marts. This means that the data mart was totally independent of any organizational data warehousing strategy or other data warehousing effort.

3.1 The data mart phenomenon

This direction then was to build data marts whenever and wherever needed, with little or no thought given to an enterprise data warehouse. This did result in many benefits for those specific organizations implementing the data marts. And because of those benefits, many others wanted a data mart too — thus the advent of data mart proliferation.

Although there were benefits for the individual departments or business areas, it was soon realized that along with those benefits. came many issues for the enterprise. One major issue is depicted in Figure 3-2. Soon many data marts began to appear, all taking time, money, and resources. The bigger issues centered around the quality and consistency of the data, and overall cost.

Figure 3-2 Data mart proliferation

The proliferation of data marts has resulted in issues such as:

- Increased hardware and software costs for the numerous data marts
- Increased resource requirements for support and maintenance
- Development of many extract, transform, and load (ETL) processes
- Many redundant and inconsistent implementations of the same data
- Lack of a common data model, and common data definitions, leading to inconsistent and inaccurate analyses and reports

- Time spent, and delays encountered, while deciding what data can be used, and for what purpose
- Concern and risk of making decisions based on data that may not be accurate, consistent, or current
- No data integration or consistency across the data marts
- Inconsistent reports due to the different levels of data currency stemming from differing update cycles; and worse yet, data from differing data sources
- Many heterogeneous hardware platforms and software environments that were implemented, because of cost, available applications, or personal preference, resulting in even more inconsistency and lack of integration

3.1.1 Data mart proliferation

Recognizing the growing issues surrounding data marts, there has begun a trend back to a more centralized data warehousing environment. However, there are still many of the same issues to be faced and decisions to be made, such as:

- Who controls the data?
- What is the priority for access to the data warehouse?
- How many users can be supported from a particular organization?
- Will I get acceptable response times on my queries?

Based on answers to these questions, there may still be a desire for data marts. However, realizing the issues with independent data marts, departments and business areas should take a different approach. Some have decided to still create data marts, but with the following considerations:

- Source data only from the enterprise data warehouse
- Implement independently to achieve a faster implementation
- Contract for services to build the data mart

These are still data marts, but are called *dependent* data marts. This is because they depend on the data warehouse for their data. Though this can resolve some of the issues, there are still many others. Consider consolidation, for example.

Consolidation

This data mart proliferation has brought many organizations to the point where often the costs are no longer acceptable from an enterprise perspective — thus, the advent of the data mart consolidation (DMC) movement. It is realized that to keep the benefits that had been so valuable, the enterprise must now get their data and information assets under control, use them in an architected manner, and manage them.

Advances in hardware, software, and networking capabilities now make an enterprise level data warehouse a reachable and preferred solution. What are the benefits of DMC to the enterprise? Here are a few:

- Reduce costs, by eliminating the redundant hardware and software, and the associated maintenance.
- Reduce costs by consolidating to a single modern platform based on current cost models. Price performance improvements over the past few years should enable the acquisition of replacement systems that are much less costly than the original systems.
- Improve data quality and consistency by standardizing data models and data definitions to:
  - Regain confidence in the organizations reports, and the underlying data on which they are based.
  - Achieve the much-discussed “single version of the truth” in the enterprise, which really means having a single integrated set of quality data that accurately describes the enterprise environment and status, and upon which decision makers can rely.
  - Improve productivity of developers in accessing and using the data sources, and users in locating and understanding the data.
  - Satisfy regulatory requirements, such as Basel II and Sarbanes Oxley.

Note: BASEL II is a committee of central banks, bank supervisors, and regulators from the major industrialized countries. For more information on Sarbanes Oxley, see:

http://www.sarbanes-oxley.com

- Enable the enterprise to grow, and evolve to next generation capabilities, such as real-time data warehousing.
- Integrate the enterprise to enable such capabilities as business performance management for a proactive approach to meeting corporate goals and measurements.

DMC does not mean we recommend that all enterprises eliminate all of their data marts. There are still valid and justifiable reasons for having a data mart, but we just need to minimize the number. When we say this, we are primarily referring to the independent data marts, but we should also seek to minimize the number of dependent data marts. Why? Well, although the data will be consistent, there are still issues. For example, the data may not always be as current (up to date) as we would like it to be — depending on the type and frequency of the data refresh process.

In general, although at a high level, this discussion demonstrates that it is worth giving serious thought to starting a data mart consolidation initiative.

3.2 A business case for consolidation

Now let us take a look at some additional reasons for considering a data mart consolidation initiative. In general, the reasons to do so are to:

- Save money by cutting costs.
- Enable you to be more responsive to new business opportunities.
- Enhance business insight through better data quality and consistency.
- Improve the productivity of your developers with new techniques and tools, and your users by easier access, standard reports, and ad hoc requests.

But, there are also many other benefits as we look a bit deeper. These are discussed further in the subsequent sections.

3.2.1 High cost of data marts

One of the primary points in this redbook is that there is a high cost associated with data mart proliferation. The cost is high because many organizational areas and resources are impacted. As examples, IT systems, development, and users will all realize increased costs because they all play a role in the creation and maintenance of a data mart. This is depicted in Figure 3-3.

Figure 3-3 Cost model for data mart creation

Many factors contribute to that cost, and we mention a few of them here:

- Departments and organizations want to have their own data marts so they can control and manage the contents and use. The result is many more data marts than are actually needed to satisfy the requirements of the business — thus, a much higher hardware, software, development, and maintenance cost for the enterprise.
- Heterogeneous IT environments abound. Many enterprises have multiple RDBMSs, reporting tools, and techniques, to create and analyze the data marts. This becomes an ever increasing expense for an enterprise in the form of the number of resources, additional training and skills development, maintenance costs, and additional transformations to integrate the data from the various data marts, since they typically use different data types and data definitions that must be resolved. This situation becomes further exacerbated with the increasing use of unstructured data, which requires more powerful and more expensive hardware and software — further increasing the costs of the data marts.
- Much of the data in an enterprise is used by multiple people, departments, or organizations. If each has a data mart, the data then exists in all those multiple locations. This means data redundancy, which means duplicate support costs. However, it is an even more expensive proposition because of the potential problems it can cause. For example, having the same data at multiple locations, but likely refreshed on a different periodic basis, can result in inconsistent and unreliable reports. This is a management nightmare.
- Management and maintenance of the data marts becomes more expensive because it must be performed on the many different systems that are installed, on behalf of the same data. For example, much of the same data will be extracted and delivered multiple times to populate multiple data marts.
- Having multiple data marts that are developed and maintained by multiple organizations will undoubtedly result in multiple definitions of the same or similar data elements because of non-standardization. This in turn will typically also cause inconsistent and unreliable reports. In addition, much time will be spent determining which data is needed, and from which sources, for each identified purpose.
- Application development is also a very important key factor, that can increase the cost of data marts. For example, as requirements change, the organization must pay for application changes in the multiple data mart environment — with such activities as customizing the ETL, the reporting processes, and sometimes even the data mart data model.

With the data mart proliferation and data redundancy, the total cost of ownership (TCO) is significantly increased.

3.2.2 Sources of higher cost

As we has seen in 3.2.1, “High cost of data marts” on page 54, one reason for consolidation is the elimination of data redundancy. Many departments or business areas in an enterprise believe that they need a local data mart that is under their own control and management. But, at what cost? Not only is there redundancy of data, but also of hardware, software, resources, and maintenance costs. And with redundancy, you automatically get all the issues of data currency — one of the biggest reasons for inaccurate and inconsistent reporting.

That is, many data marts contain data from some of the same sources. However, they are not updated in any consistent controlled manner, so the data in each of the data marts is inconsistent. It is current as of a specific time, but that time is different for each data mart. This is a prime reason for inconsistent reporting from the various departments in an enterprise.

We have depicted a number of the cost components inherent in independent data marts in Figure 3-4. And, their impact is realized on each of the multiple data marts in the enterprise. These are the areas to be examined for potential cost reduction in a data mart consolidation project.

Figure 3-4 Cost components of independent data marts

Another major source of the higher cost is the maintenance; that is, the time and resources required to get all those data marts loaded, updated, and refreshed. Even those activities are not without issues. For example, there is a need to keep the data marts in sync. What does mean? Let us take an example. Say that a department owns two data marts. One services the marketing organization and the other the sales organization. Both data marts are independent and are refreshed on separate schedules. The impact is that the logical relationship between the content of the data marts is not consistent — nor are the reports that are based on them. These are the issues that continue to confuse and irritate management, and all decision makers. They cannot trust the data, and it costs time, money, and resources to analyze and resolve the issue. Here is your opportunity to eliminate those issues.

Another big cost factor is the additional hardware and software required for the data marts. And, that cost actually expands because there is other related hardware and software to support the data mart. As examples, you need:

- ETL (extract, transform, and load) capability to get data from the source systems into the data mart, and update it on an on-going basis
- Services to keep the hardware and software working properly, and updated to supported levels
- Space to house the hardware and peripherals
- Specialized skills and training when heterogeneous, and incompatible, hardware and software systems are used

So how can we reduce these costs?

3.2.3 Cost reduction by consolidation

One consequence of data mart proliferation is that the most organizations have business applications that run on a complex heterogeneous IT infrastructure, with a variety of servers, storage devices, operating systems, architectures, and vendor products. The challenge then is to reduce costs for IT hardware and software, as well as the skilled resources required to manage and maintain them.

Here are a few examples of the types of costs that can be reduced:

- Multiple vendors for hardware and software
- Multiple licences for database management systems
- Support costs for implementation and maintenance
- Maintenance for hardware and software products
- Software development and maintenance
- User skills and training

Archived

Hardware capability

From a hardware perspective, consideration could be given to changing from a 32-bit environment to a 64-bit environment. Applications require more and more memory, which may not be able to be accommodated by the 32-bit technology. One of the big advantages of the 64-bit technology is that applications can use more than 4 GB of memory. This also translates to increased throughput and the associated decrease in the price for additional throughput. That is, it also gives you a price/performance advantage.

Development costs

Furthermore, having many different vendor applications can be a source of higher costs. For example, reporting systems or ETL products typically come with their own special environments. That is, they need special implementations such as their own repository databases. That will lead to additional expenses in software development, a high cost in any IT project. With a heterogeneous IT software infrastructure, these costs will be even higher.

Primary reasons for the increased development expenses are:

- Different APIs, which lead to additional porting effort
- Different SQL functions, which are typically incompatible among the vendors
- Incompatibility of version levels, even with the same software
- Unsupported functions and capabilities
- Developing of the same or similar applications multiple times for the multiple different system environments

The effort for user training, and classes to build skills, is significant. It is exacerbated by the constant release changes that come with the software packages: The fewer the packages used and the less heterogeneity, the lower the typical overall cost. For example, with multiple disparate tools, you lose negotiating leverage for such items as volume discounts because you are dealing with multiple vendors and lower volumes.

Software packaging

DB2 Universal Database (UDB) Data Warehouse Edition (DWE) can help in the consolidation effort because it includes software for the following items:

- DBMS
- OLAP
- Data marts
- Application development

There are several editions available, based on the capabilities desired. As an example, consider the Enterprise Edition.

DB2 Data Warehouse Enterprise Edition is a powerful business intelligence platform that includes DB2, federated data access, data partitioning, integrated online analytical processing (OLAP), advanced data mining, enhanced extract, transform, and load (ETL), workload management, and it provides spreadsheet integrated BI for the desktop. DWE works with and enhances the performance of advanced desktop OLAP tools such as DB2 OLAP Server™. The features are:

- DB2 Alphablox for rapid assembly and broad deployment of integrated analytics
- DB2 Universal Database Enterprise Server Edition
- DB2 Universal Database, Database Partitioning Feature (large clustered server support)
- DB2 Cube Views (OLAP acceleration)
- DB2 Intelligent Miner™ Modeling, Visualization, and Scoring (powerful data mining and integration of mining into OLTP applications)
- DB2 Office Connect Enterprise Web Edition (Spreadsheet integration for the desktop)
- DB2 Query Patroller (rule-based predictive query monitoring and control)
- DB2 Warehouse Manager Standard Edition (enhanced extract/transform/load services supporting multiple Agents)
- WebSphere Information Integrator Standard Edition (in conjunction with DB2 Warehouse Manager, provides native connectors for accessing Oracle databases, Teradata databases, Sybase databases, and Microsoft SQL server databases)

A recent addition to DB2 DWE is DB2 Alphablox, for Web analytics. With DB2 Alphablox, DB2 DWE rounds out the carefully selected set of IBM Business Intelligence (BI) products to provide the essential infrastructure needed to extend the enterprise data warehouse. DB2 Alphablox can also be deployed in operational applications to provide embedded analytics, extending the DWE value-add beyond the data warehouse environment.

DB2 Alphablox extends DWE with an industry-leading platform for the rapid assembly and broad deployment of integrated analytics and report visualization embedded within applications. It has an open, extensible architecture based on Java™ 2 Platform, Enterprise Edition (J2EE) standards, an industry standard for developing Web-based enterprise applications. DB2 Alphablox simplifies and speeds deployment of analytical applications by automatically handling many details of application behavior without the need for complex programming.

Archived

As delivered in both Enterprise and Standard DWE editions, DB2 Alphablox includes the DB2 Cube Views metadata bridge. The synergistic combination of these technologies in DWE means DB2 Alphablox applications, using the relational cubing engine, can connect directly to the DB2 data warehouse and still enjoy a significant range of multidimensional analytics and navigation, along with the performance acceleration of DB2 Cube Views. DB2 Alphablox in DWE also includes the standard relational reporting component, adding value in environments where SQL is important for reporting applications.

Because DB2 Alphablox is intended to access data solely through the DB2 data warehouse (including remote heterogeneous data sources via optional WebSphere II federation), this version of DB2 Alphablox in DWE does not include multidimensional connectors for accessing non-relational MOLAP-style cube servers, nor does it include relational connectors for other IBM or non-IBM databases. These connectors are available via separate licensing.

For more information about Data Warehouse Edition and DB2 Alphablox, please refer to the IBM Web page:

http://www-306.ibm.com/software/data/db2/alphablox/

3.2.4 Metadata: consolidation and standardization

Metadata is very important, and forms the base for your data environment. It is commonly referred to as “data about data”. That is, it constitutes the definition of data with such elements as:

- Format
- Coding/values
- Meaning
- Ownership

Having standardized or consistent metadata is key to establishing and maintaining high data quality. Thus management and control of the use of metadata in a data mart environment is critical in maintaining reliable and accurate data.

High data quality and consistency is the result of effective and efficient business processes and applications, and contributes to the overall profitability of an enterprise. It is required for successful data warehousing and BI implementation, and for enabling successful business decision-making.

Poor quality data costs time and money, and it will lead to misunderstanding of the data and erroneous decision-making. To achieve the desired quality and consistency requires standardization of the data definitions. And, if there are independent data marts, the obvious conclusion is that there will be no common or standardized metadata. Therefore, with independent data marts you cannot guarantee that you have data that has high quality and consistency.

The data definitions are created and maintained in the *metadata*. This metadata must be managed and maintained, and standardized across the enterprise, if we want high quality and reliable data. Managing the metadata is in itself a major task, and impacts every area of the business.

For example, assume that there is a data element called *inventory*. It has metadata that consists of the definition of inventory. If the proper usage of that data element is not managed, we know we will have inconsistent and inaccurate reporting.

In this particular example, we could have the following choices for a definition for inventory:

1. The quantity of material found in the enterprise storage areas
2. The quantity of material found in the enterprise storage areas, plus the material in the production areas waiting to be processed.
3. The quantity of material found in the enterprise storage areas, plus the material in the production areas waiting to be processed, plus the material that has been shipped, but not yet paid for (assuming we relieve inventory when a purchase has been completed).

When we have choices, it can be difficult to maintain control and manage those choices. For example:

- Which of the definitions for inventory is correct?
- Which are used in the independent data marts?
- Do the production departments understand which definition is correct?

This metadata management and control is a critical success factor when implementing a data warehousing environment — and indeed, in any DMC effort.

Archived

3.2.5 Platform considerations

As we have discussed in “Cost reduction by consolidation” on page 57, there exist opportunities to reduce the costs by changing the platform. This is in itself a consolidation project.

> **Note:** The term *platform* refers to the hardware and operating system software environment.

Consideration must be given not only to the power of the hardware, but the operating system as well. And, care must be taken to ensure that all the required software products operate together on the selected platform.

For example, when considering a software change, here are some things you should consider:

- Is the same functionality available with the new software platform? For example, does the DBMS comply with the SQL ANSI92 standard? And, does it have powerful SQL features, scalability, reliability, and availability to support the desired environment? Does it support capabilities such as stored procedures, user defined function, triggers, and user defined data types?
- Is there a good application development environment that is flexible and easy to use — one that makes the new Web-based applications easy and fast to develop?
- What is the existing reporting environment, and what, and how many, changes will be required?
- How may ETL programs and processes have to changed and customized? Will the number be able to be reduced, or be less complex?

3.2.6 Data mart cost analysis sheet

In this section we look at costs associated with data marts, from two aspects:

1. **Data mart cost analysis:** To help you get started in determining the costs associated with your data mart environment, we have developed a Cost Analysis worksheet template. It is depicted in Figure 3-5.

   This data mart cost analysis can help identify costs such as hardware, software, networking, third party tools, ETL tools, and storage. In addition to the cost to purchase and implement particular hardware and software, it also includes costs for maintaining the data marts.

Archived

Cost Category	Description	$$ Amount
RDBMS data mart		
RDBMS repository		
Other software (ETL, migration, etc.)		
Hardware		
Consultants and contractors		
Employees		
Maintenance (platform, development, usage)		
Networking		
Third party tools		
Reporting tools		
ETL development costs		
Business user training		
IT developer training		
Storage cost		
Operations		
Administration		
and more		

Figure 3-5 Data mart cost analysis sheet

2. **Data mart business loss analysis:** In addition to the cost of implementing and maintaining data marts, you must consider the ability to eliminate or minimize losses for your enterprise. These losses could result from such things as missed opportunities and poor decisions that were based on poor quality, inconsistent, or otherwise inadequate data.

   To help you get started in making such an analysis, we have provided a basic structure for a Loss Analysis template worksheet, which is depicted in Figure 3-6. In this example, it shows a cost sheet to help analyze the loss associated with inadequate data analysis or missed opportunities due to data mart proliferation, and data quality issues relating to silo data marts.

Archived

Cost Category	Description	$$ Amount
Business loss – disintegrated data results in faulty analysis and missed opportunities.		
Excessive operational costs – printing and mailing from inaccurate listings		
Customer loss – attrition from ineffective marketing due to analysis of inconsistent data from a data mart.		
Customer loss – attrition due to poor service from analysis of inconsistent data from an independent data mart.		
Business loss – sales loss from out of stock condition due to inaccurate and inconsistent inventory data.		
Business loss – lost opportunity due to inconsistent and non-current data.		
and more		

Figure 3-6 Data mart business loss analysis sheet

Having a structure approach and some tools will go a long way towards helping you determine the real costs associated with data mart proliferation.

It may also be possible to connect the lack of integrated data with constraints on the business results. When this is possible, you can typically produce further business justification. For example, consider your development backlog. With the improved productivity possible with consolidation, you may be able to accelerate the delivery of the development requirements; which itself can then lead to earlier realization of the benefits.

3.2.7 Resolving the issues

When ready for a DMC project, you must develop a plan. Key to that plan is the identification and prioritization of the data marts to be consolidated. Although the easy answer would be to consolidate all the data marts, that is seldom the optimal result.

In most enterprises, there will be a justifiable need for one or more data marts. For example, consider the use of spreadsheets and other PC-based databases. As we have learned, these can also be considered as types of data marts. But it does not make sense to just eliminate them all.

So, what are good candidates for DMC? Here are a few:

- Data marts on multiple hardware and software platforms
- Independent data marts
- Spreadsheet and PC based data marts (such as Microsoft Access)
- Data marts implemented with multiple query tools

Consideration must be given to many other factors, such as:

- Availability of hardware on which to consolidate, because most often we will consolidate onto a new platform
- Decisions on consolidation of operating environments
- Application conversion effort
- Resource availability
- Skills availability
- Variety of data sources involved
- Volumes of data involved
- Selection of an ETL capability
- Volume of ETL used in a particular data mart

Such a project is much like any other IT development project. Careful consideration must be given to all impacted areas, and a project plan developed accordingly.

3.3 Summary

In this section, the focus has been on the impact of consolidation. There are many benefits, both tangible and intangible, and we have presented some of them for your consideration. We have shown that data mart consolidation can enable us to:

- Simplify the IT infrastructure and reduce complexity
- Eliminate redundant:
  - Information
  - Hardware
  - Software
- Reduce the maintenance effort for hardware and software
- Reduce the costs for software licenses
- Develop higher quality data
- Standardize metadata to enable consistent and high quality data

All this will enable you to create an environment to help you reach the goal of a *"single version of the truth"*.

Archived

Archived

4

Consolidation: A look at the approaches

Companies worldwide are moving towards consolidation of their analytical data marts. The need to remove redundant processes, to reduce software/hardware and staff costs, and to develop their “single version of the truth” have become key requirements for managing business performance. The need for data consolidation to produce accurate, current, and consistent information is changing from a “nice to have” to a “must-have” mandatory requirement for most enterprises.

In this chapter, we discuss the following topics:

- What are good candidates for consolidation?
- Data mart consolidation lifecycle
- Approaches to consolidation
- Consolidating data schemas
- Consolidating other analytic structures
- Other consolidation opportunities
- Tools for consolidation
- Issues faced in the consolidation process

© Copyright IBM Corp. 2005. All rights reserved.

4.1 What are good candidates for consolidation?

Enterprises around the world have implemented many different types of analytic structures, such as data warehouses, dependent data marts, operational data stores (ODS), spreadsheets, independent data marts, denormalized databases for reporting, replicated databases, and OLAP servers.

Here we list some of the important analytical structures that are candidates for consolidation:

- **Data warehouses:**

  In the case of external events such as mergers and acquisitions, there may exist two, or more, data warehouses. Typically in such scenarios, the data warehouses are also merged. It may be that there is a data warehouse that has been expanded over time, without a specific strategy or plan, or one that has drifted away from using best practices.

- **Dependent data marts (hub/spoke architecture):**

  An enterprise may choose to consolidate dependent data marts into the EDW for achieving hardware/software, resources, operational, and maintenance related savings.

- **Independent data marts:**

  Independent data marts are the best candidates for consolidation. Some of the benefits of consolidating independent data marts result from hardware/software savings, cleaner integrated data, standardized metadata, operational and maintenance savings, elimination of redundant data, and elimination of redundant ETL processes.

- **Spreadsheet data marts:**

  Spreadsheet data marts are silo analytical structures of information that have been created by many different users. Such marts have helped individuals, but may have been detrimental to the organization from an enterprise integrity and consistency perspective. These, and other PC databases, have been used for independent data analysis because of their low cost. But then it becomes apparent that development and maintenance of those structures becomes very expensive.

- **Others:**

  Other analytical structures which may be candidates for consolidation are flat files, denormalized databases, or any system which is becoming obsolete.

Archived

In 4.2, “Approaches to consolidation” on page 71, we discuss the various techniques for consolidation such as simple migration, centralized and distributed approaches.

In 4.4, “Consolidating the other analytic structures” on page 93, we discuss how the consolidation approaches can be used to consolidate the various analytical structures.

4.1.1 Data mart consolidation lifecycle

In this section we provide you with a brief overview of the lifecycle to use as an introduction, and to consider as you read this redbook. It is an important guide, and, as such, we have dedicated an entire chapter to it. We explain it in much more detail in Chapter 6, “Data mart consolidation lifecycle” on page 149.

Data mart consolidation may sound simple at first, but there are many things to consider. You will need a strategy, and a phased implementation plan. To address this, we have developed a data mart consolidation lifecycle that can be used as a guide.

A critical requirement, as with almost any project, is executive sponsorship. This is because you will be changing many existing systems that people have come to rely on, even though some may be inadequate or outmoded. To do this will require serious support from senior management. They will be able to focus on the bigger picture and bottom-line benefits, and exercise the authority that will enable changes to be made.

In addition to executive sponsorship, a consolidation project requires support from the management of the multiple business functional areas that will be involved. They are the ones that best understand the business requirements and impact of consolidation. The activities required will depend on the consolidation approach selected. We discuss those in 4.2, “Approaches to consolidation” on page 71.

The data mart consolidation lifecycle guides the consolidation project. For example, the activities involved in consolidating data from various heterogeneous sources into the EDW are depicted in Figure 4-1.

Archived

Figure 4-1 Data mart consolidation lifecycle

The data mart consolidation lifecycle consists of the following activities:

- **Assessment:** During this phase we assess the following topics:
  - Existing analytical structures
  - Data quality and consistency
  - Data redundancy
  - Source systems involved
  - Business and technical metadata
  - Existing reporting needs
  - Reporting tools and environment
  - Other BI tools
  - Hardware/software and other inventory

> **Note:** Based on the assessment phase, the “DMC Assessment Findings” report is created.

- **Planning:** Some of the key activities in the planning phase include:
  - Identifying business sponsor
  - Identifying analytical structures to be consolidated
  - Selecting the consolidation approach
  - Defining the DMC project purpose and objectives
  - Defining the scope
  - Identifying risks, constraints, and concerns

  - In the planning phase above, based on the DMC Assessment Findings report, the Implementation Recommendation report is created.
- **Design:** Some of the key activities involved in this phase are:
  - Target EDW schema design
  - Standardization of business rules and definitions
  - Metadata standardization
  - Identify dimensions and facts to be conformed
  - Source to target mapping
  - ETL design
  - User reports
- **Implementation:** The implementation phase includes the following activities:
  - Target schema construction
  - ETL process development
  - Modifying or adding end user reports
  - Standardizing reporting environment
  - Standardizing some other BI tools
- **Testing:** This may include running in parallel with production.
- **Deployment:** This will include user acceptance testing.
- **Loopback:** Continuing the consolidation process, which loops you back to start through some, or all, of the process again.

4.2 Approaches to consolidation

From a strategic perspective, there are a number of projects and activities you should consider to architect and organize your IT environment. These activities will result in improved productivity, lower operating costs, and an easier evolution to enhanced products and platforms as they become available. Basically they all fall under a general category of “getting your IT house in good order”.

We are certainly not advocating that all these activities be done before anything else. They will no doubt take some time, so they should be done in parallel with your “normal” activities. The following list describes some of the activities; you may also have others:

- Cleanse data sources to improve data quality.
- Minimize/eliminate redundant sources of data.
- Standardize metadata for common data definitions across the enterprise.
- Minimize and standardize hardware and software platforms.
- Minimize and standardize reporting tools.
- Minimize/eliminate redundant ETL jobs.
- Consolidate your data mart environment.

Archived

Of course, our focus in this redbook is on consolidating the data mart environment. And, as you will no doubt recognize, all the other activities on the list will help make that task easier. But, you should not wait on them to be completed — work in parallel.

So how do you get started? There are a number of approaches that can be used for consolidating data marts into an integrated enterprise warehouse. Each of these approaches, or a mix of them, may be used depending on the size of the enterprise, speed with which you need consolidate, and potential cost savings.

There are three approaches we will consider in this redbook:

- Simple migration (platform change, with same data model)
- Centralized consolidation (platform change, with new data model or changes to existing data model)
- Distributed consolidation (no platform change, with dimensions being conformed across existing data marts to achieve data consistency)

The following sections get into more detail on these approaches.

4.2.1 Simple migration

In the simple migration approach, certain existing data marts or analytical silos can be moved onto a single database platform. We believe that platform should be DB2. This could be considered a step in the evolution of a consolidation.

The only consolidation that occurs during this approach is that all data from independent data marts now exists on a single platform, but there is still disintegrated and redundant information in the consolidated platform. This is a quicker approach to implement, but does not integrate information in a manner to provide a single version of the truth.

The key features of the simple migration approach are as follows:

- All objects, such as tables, triggers, and stored procedures, are migrated from independent data marts to the centralized platform. Some changes may be required when there are platform changes.
- The users see no change in terms reporting. The reports continue to work the same way. Only the connection strings change from previous independent data marts and point to the new consolidated platform.

- The ETL code to extract data from source to target consolidation platform may be affected in the following manner:
  - For handwritten ETL code, changes need to be made to the SQL stored procedures when they are migrated from one database, for example, SQL Server 2000, to DB2. It may be that the stored procedure for ETL written in SQL Server 2000 uses functions available only in SQL Server 2000. In such a scenario, some adjustments may be need to be made to the stored procedure that will be used in the DB2 database.
  - If using a modern ETL tool, some minor modifications may be necessary in the ETL processes. But they will typically be straightforward, particularly with a tool such as WebSphere DataStage — for example, re-targeting the data flows and regenerating them for the DB2 enterprise data warehouse.
- Metadata and business definitions of common terms:

  From a conceptual standpoint, the metadata remains the same. There is no integration of metadata in simple migration approach. For example, let us assume there are two independent data marts named “sales” and “inventory”. Each of these data marts define the entity called “product” but in a *different* manner. The definition of such metadata remains the same even if these two data marts are migrated into the consolidated DB2 platform. Metadata associated with tools, data model, ETL processes, and business intelligence tools will also remain the same.

Advantages of simple migration approach

The advantages of the simple migration consolidation approach (as shown in Figure 4-2) are:

- Cost reduction in the form of:
  - Fewer resources required to maintain multiple independent data marts and technologies.
  - Hardware/software
- Secured and unified access to the data
- Standardization of tasks such as backups, security, and recovery.

Archived

Figure 4-2 Simple migration

Issues with simple migration approach

Issues with the simple migration consolidation approach (as shown in Figure 4-2) are as follows:

- The quality of data in the consolidated platform is the same as the quality of data which was present in the independent data marts before consolidation.
- No new functionality is enabled. For example, we do not introduce any new surrogate keys to maintain versioning of dimension data or to maintain history. We only migrate the existing data and code (ETL) to the new platform.
- There is no integration of data.
- Duplicate and inconsistent data will still exist.
- Multiple ETL processes are still required to feed the consolidated platform.
- Technical and business metadata are still not integrated. Figure 4-3 shows that independent data marts have their metadata repository, which is non-standardized and disintegrated. Using the simple migration consolidation approach, only the data is transferred to a central platform. And, there is no metadata integration or standardization. The metadata remains the same.

Archived

Figure 4-3 Metadata management (Simple Migration)

- In short, using this approach, the enterprise does not achieve a single version of the truth.

The simple migration consolidation follows a conventional migration strategy, such as would be used to migrate a databases from one platform to another. During the migration of data from one or more data marts to a consolidated platform, you need to understand the following elements:

- **Data sources and target:** Understand the number of objects involved in the transfer and all their inter-relationships.
- **Data transformations:** Data types between sources and target databases maybe incompatible. In such cases, data needs to be transformed before being loaded into the target platform. For an example of how this is done, please refer to Chapter 7, “Consolidating the data” on page 199 for data conversion from Microsoft SQL Server 2000 and Oracle 9i database to DB2.
- **Data volumes:** When several data marts are being consolidated on a single platform, it is important that any scalability issues be clearly understood. For example, it must have sufficient processing and I/O capability.
- **Storage:** Understand the space requirements of the consolidated database on a single platform — including needs for future growth.

Note: With this approach, the independent data marts cease to exist after the data has been migrated to the new consolidated platform.

When to use the simple migration approach?

An enterprise may decide to use this approach when:

- The primary goal is to get a quick cost reduction in software/hardware without incurring the cost of data integration.
- There are operational issues with tasks such as data feeds, backup, and recovery, that need to be resolved.
- The presence of obsolete software or hardware may require moving to a new technology.
- It is a first step in a larger consolidation strategy.
- A large volume and variety of data sources and data marts are involved, which will require a longer time and detailed project definition.

4.2.2 Centralized consolidation

In the centralized consolidation approach, the data can be consolidated in two ways, as shown in Figure 4-4:

- Centralized consolidation using *redesign*: In this approach we redesign the EDW. The architect of the new EDW may use the independent data marts to gain understanding of the business, however the EDW has its own new schema. The centralized consolidation approach can require significant time and effort.
- Centralized consolidation-merge with primary: In this approach, we identify one *primary* data mart. This primary data mart is then chosen to be the first to be migrated into the EDW environment. All other independent data marts migrated later are then conformed according to the primary data mart that now exists in the EDW. Basically, in this technique, one data mart is chosen to be primary and all others are later merged into it. This is why we also call it the “merge with primary” technique.

Archived

Figure 4-4 Two techniques for the centralized consolidation approach

The basic idea in the centralized consolidation approach as shown in Figure 4-5 is that the information from various independent data marts is consolidated in an *integrated and conformed* way. By integration, we mean that we identify the common information present in the different independent and disintegrated data marts. The common information needs to be conformed, or made consistent, across various independent data marts.

Once conformed dimensions and facts are identified, the design for the EDW takes place. While designing the schema for the EDW, you could completely redesign it if you find serious data quality issues. Also, we could use one of the two independent data marts as the primary mart and use it as the base schema of the EDW. All other independent data marts migrated later are conformed according to the *primary* data mart which exists in the EDW.

Note: A conformed dimension means the same thing to each fact table to which it can be joined. A more precise definition of a conformed dimension is: Two dimensions are said to be conformed if they share one, more, or all attributes that are drawn from the same domain. In other words, one dimension may be conformed even if it contains a subset of attributes from the primary dimension.

Fact conformation means that if two facts exist in two separate location in the EDW, then they must be the same to be called the same. As an example, revenue and profit are facts that must be conformed. By conforming a fact we mean that all business processes must agree on a common definition for the “revenue” and “profit” measures so that separate revenue/profit from separate fact tables can be combined mathematically.

Figure 4-5 Centralized Consolidation

The EDW schema in centralized consolidation is designed in a way that eliminates redundant loading of similar information. This is depicted in Figure 4-6 which shows two data marts being populated independently by separate OLTP systems, called Sales and Marketing. Both the independent data marts have common information, such as customer, but this common information from an organizational standpoint is not integrated at the data mart levels.

Figure 4-6 Independent data marts showing disintegrated customer information

In Figure 4-7 we see that in the case of centralized consolidation, the common information across the independent data marts is identified and the EDW is designed keeping conformity of customer information in mind. This process of

conformance is repeated for any new data marts added in the EDW. Not only do we conform dimensions but we can also conform facts.

Figure 4-7 Centralized EDW with standardized & conformed customer information

However, once conformity has been achieved, the design of the EDW using the centralized consolidation approach can follow either of the two techniques as shown in Figure 4-4.

The key features of centralized consolidation are as follows:

- The primary focus is to integrate data and achieve data consistency across the enterprise.
- There is a platform change for all independent data marts that move to a consolidated EDW environment.
- Data is cleansed and quality checked, and sustainable data quality processes are put in place.
- Data redundancy is eliminated.
- Surrogate keys are introduced for maintaining history and versioning.
- Redundant ETL processes are eliminated. The new ETL process generally involves the following:
  - ETL logic is needed to transfer data from the old data marts to EDW, in the form of a one-time activity to load existing historical data.
  - ETL logic is needed to feed the EDW from the source OLTP systems.
  - ETL logic originally used to feed the old data marts is abandoned.

Archived

- Reports being generated from independent data marts are affected in the following way after consolidation:
  - Reporting environments may change completely if the organization decides to rationalize several old reporting tools and choose a new reporting tool as the corporate standard for reporting.
  - Reports will change even if the reporting tool remains the same. This is because the back-end schema has changed to the consolidated EDW schema, so reports need to be re-implemented using the new data model, and re-tested.
  - The entire portfolio of reports are examined and reduced to a standard set, with many of the old redundant reports being completely eliminated. It is not unusual for 50% to 70% of existing reports to be found to be obsolete.
- Metadata:

  Metadata is standardized in this approach. As shown in Figure 4-8, in the case of independent data marts, there is no standardization across various elements of metadata — for data mart “1”, “2”, “3”, ...“n”, each has its own metadata environment. Basically, what this means is that the greater the number of independent data marts, the more inconsistency there is in the data and metadata.

  Each data mart defines metadata and creates repositories for various metadata elements in its own way. On the other hand, using the centralized consolidation approach, the metadata is standardized for the enterprise.

Figure 4-8 Managing the metadata

- These are some benefits of having a common standardized metadata management system:
  - It helps to maintain consistent business terminology across the enterprise.
  - It reduces the dependency of business users on IT, for activities such as running reports.
  - It assists in helping users understand the content of the data warehouse.
  - It speeds development of new reports and maintenance of existing reports since data definitions are easy to discover and less time is spent trying to determine what the data should be.
- Schema:

  The impact of having a different schema depends on the approach:
  - Centralized consolidation - Using Redesign: This means that a new EDW schema is designed.
  - Centralized consolidation - Merge with Primary: This means that an existing data mart or data warehouse is chosen as the primary schema and other independent data marts are merged into it. The schema for the primary data mart or data warehouse undergoes minor, or no, changes.

Note: In centralized consolidation, the independent data marts cease to exist after the EDW has created. However, until the time the EDW is under construction, the independent data marts continue to exist to produce reports. In fact it may be important to run the old and new systems in parallel during user acceptance testing and to enable users to prove to themselves that the new system satisfies their needs.

Advantages of the centralized consolidation approach

The advantages of the f centralized consolidation approach are as follows:

- It provides quality assured, consistent data.
- It cuts down the cost and provides a better ROI over a period of time.
- Consolidating data from independent data marts helps enterprises to meet government regulations such as Sarbanes Oxley, because the quality, accuracy, and consistency of data improves.
- It standardizes the enterprise business and technical metadata.
- Independent data marts typically do not maintain the history and versions of their dimensions. Using the centralized consolidation approach, we maintain the proper history and versioning. An example of this is shown in our sample exercise in Chapter 9, “Data mart consolidation: A project example” on page 255. We use a dimension mapping table to maintain history and

Archived

versioning. By using this approach we can also maintain the structure changes that happen to hierarchies over time.

- It provides a more secure environment for the EDW, as it is managed centrally.
- It is a starting point for standardizing several of enterprises tools and processes such as these:
  - The reporting environment for the entire enterprise may be standardized.
  - Several tools are involved with any data mart. Some examples are tools used for configuration management, data modeling, documentation, OLAP, project management, Web servers, and third party tools. After consolidation, many of these tools could also be rationalized to a more manageable subset.

Note: We are not suggesting that you necessarily standardize on one query tool. It may be that two or three are still required, but that may be better than the typical six to eight often found in use in many organizations.

Issues with centralized approach

The disadvantages of centralized consolidation approach are:

- It requires considerable time, expertise, effort and investment.

When to use the centralized consolidation approach?

An enterprise may decide to use this approach:

- When enterprises want to have the ability to look at trends across several business/functional units. We show a practical example of consolidating such independent silo data marts in Chapter 9, “Data mart consolidation: A project example” on page 255.
- If the enterprise wants to standardize its business and technical metadata.
- After an acquisition. Both environments should be examined to determine which data mart or data warehouse will be the primary and which will be merged.
- When a enterprise has two or more data warehouses/marts that need to be consolidated.

4.2.3 Distributed consolidation

In distributed consolidation approach, the information across various independent data marts is consolidated without *physically* integrating the data marts. This is done by restructuring the dimensions in each data mart so that they *conform* with each other. This is shown in Figure 4-9.

The advantage of this approach is that an enterprise does not need to start afresh but can leverage the power of existing independent data marts. The disadvantage of this approach is that redesign of the data marts leads to changes in the front end applications. The redesign process can also get complex if there are multiple data marts that need to be conformed.

Typically, the distributed consolidation approach is used as a short-term solution until the enterprise is able to achieve a centralized enterprise data warehouse. That is, it may be the precursor to full integration.

Figure 4-9 Distributed consolidation approach

The key features of distributed consolidation are as follows:

- There is minor change in the dimensional structures of independent data marts being conformed.
- Some form of staging area is needed which is used to create and populate conformed dimensions into the independent data marts.
- There is minimal or no change in the transformation code for loading the independent data marts (which become dependent after conforming dimensions). Minimal change generally occurs when a column name changes after we change a non-conformed dimension in an independent data mart to a conformed dimension.
- Metadata for conformed dimensions is standardized, but the rest of the metadata remains the same.
- Existing reports will typically undergo change. This is because the dimensions in data marts are redesigned based on conformed dimensions.

Note: Unlike the simple migration and centralized consolidation approaches where the data marts were eliminated after successful consolidation, the data marts in the distributed approach continue to exist.

Archived

Advantages of the distributed consolidation approach

The main advantages of the distributed consolidation approach are:

- It can be implemented in much less time as compared to centralized consolidation.
- The organization can solve data access issues across multiple data marts by using conformed dimensions.
- Organizations can leverage the capability of existing data marts and do not need to redesign the solution.
- It provides some metadata integration at the conformed dimensions level.

Issues with distributed consolidation approach

The disadvantages of the distributed consolidation approach are:

- There is no cost reduction at the hardware/software level as the various independent data marts will continue to exist. The only difference is that their dimensions have been conformed to a standardized dimension.
- Multiple data marts still need to be managed.
- Multiple ETL processes still exist to populate data marts, although some level of redundant ETL load for conformed dimensions has been eliminated.
- Data security is distributed and lies with the administrator of each data mart. Also because there are several data marts across several hardware/software platforms, security maintenance needs more effort.

When to use distributed consolidation approach?

This approach should be used in the following circumstances:

- When the organization is not in a position to immediately eliminate the independent data marts, but needs to implement requirements for accessing data from multiple data marts. In such a scenario, the best way to introduce consistency in data is by standardizing these independent data marts to use conformed dimensions. The control of these data marts still remains with the business department or organization, but consistency is achieved.
- When it makes sense to use this approach as a starting point for moving towards a broader centralized consolidation approach.

4.2.4 Summary of consolidation approaches

There is no best approach. Each approach can help in some element of the broader consolidation effort in the enterprise.

Archived

Depending upon the needs of the enterprise, each approach, or a mix of these approaches, will help in the following ways:

- Reduce hardware/software/maintenance costs
- Integrate data
- Reduce redundancy
- Standardize metadata
- Standardize BI and reporting tools
- Improve developer and user productivity
- Improve quality of, and confidence in, the data being used

For easy reference, we have summarized some of the characteristics of the approaches in Table 4-1. This should be of assistance as you go through the decision process for your consolidation project.

Table 4-1 Consolidation approach summary

Characteristic	**Simple Migration**	**Centralized**		**Distributed**
		Redesign	**Merge with Primary**	
Hardware savings	Yes	Yes	Yes	None
Software savings	Yes	Yes	Yes	None
Data quality, conformity and integrity	None (Data quality is directly proportional to the quality of data in the independent data marts	High	High	Medium to high, depending on how much ETL change is done. ETL can impact data quality.
Security of data	High	High	High	Medium
Resource reduction	High	High	High	Some (development can now be more productive)
Complexity to design	Low	High	High	Medium to high, depending on existing data quality
Status of old independent data marts	Cease to exist	Cease to exist (once the new system is in production)	Cease to exist (once the new system is in production)	Old independent data marts are conformed and exist

Archived

Characteristic	Simple Migration	Centralized		Distributed
		Redesign	Merge with Primary	
Conformed dimensions	Do not exist	Redesign focus is on conforming information as much as possible	Conformity initially based on the *primary* data mart and other marts are folded into it.	Existing independent data marts are consolidated by introducing conformed dimensions.
Structure of EDW	Same as old independent data marts (without any integration except all earlier data marts are on same platform)	Normalized (mostly) and denormalized	Denormalized (for *primary* assigned data mart) and normalized	Denormalized (mostly star design) for individual data marts
Reports	No change to existing reports. Only connections of existing reports point to the central consolidated platform.	Existing reports change, and eliminate redundant reports.	Existing reports change, and eliminate redundant reports.	Existing reports change mostly partially due to certain dimensions being conformed and column names change
Reporting tools	Existing reports remain same (though organization may decide to standardize reporting tools)	Existing reports change (organization should standardize reporting tools)	Existing reports change (organization should standardize reporting tools)	Reporting tools remain the same
Complexity of ETL process	Data only extracted and placed in consolidated platform	High level of transformation and cleansing done	High level of transformation and cleansing done	Medium transformation done for conformed tables
ETL tools	ETL Tools maybe consolidated and to a standardized tool	ETL Tools may be consolidated and to a standardized tool	ETL Tools may be consolidated and to a standardized tool	ETL tools mostly remain same

Archived

Characteristic	Simple Migration	Centralized		Distributed
		Redesign	Merge with Primary	
Metadata repository	Repository stored on common platform as EDW but is not integrated as data marts are conceptually independent even after consolidation	Has its own integrated metadata repository from start	Metadata repository is created for *primary* data mart and other marts fold under this repository. Other marts repositories exist till final consolidation	Common metadata repository is created for all marts for conformed dimensions however individual repositories do exist.
Metadata integration savings	High	High	Moderate to high	Moderate
Speed of deployment	Very fast	Slow	Medium	Medium
Level of expertise required	Low	High	High	Medium to high
Maintenance costs	Reduced to single platform	Reduced to single platform, and eliminate redundant data.	Reduced to single platform, and eliminate redundant data	Maintenance costs remain same due to multiple platforms
Security	Centralized (EDW maintains the security)	Centralized (EDW maintains the security)	Centralized (EDW maintains the security)	Distributed (Each data mart responsible for its security)
Resource savings	High	High	High	None (Multiple servers exist and need resources for maintenance)
Operations	High	High	High	Low
DBA	Medium	High	High	Low
Development (modern tools, improved metadata, getter data quality)	Low	High	High	Medium

Archived

Characteristic	Simple Migration	Centralized		Distributed
		Redesign	Merge with Primary	
Users (modern tools, streamlined report set, easier to find/use info)	Low	High	High	Medium
ROI	Low to medium, with immediate effect after migration. Cost reduction in software and hardware.	High over period of time	High over period of time	Medium
Single version of the truth	Does not provide single version of the truth (data not integrated)	High (Provides consistent and conformed data)	High (Provides consistent and conformed data)	Medium to High
Cost of Implementing	Low	Very high (Requires high level of commitment, skills, resources)	Medium to high	Medium

4.3 Combining data schemas

Consolidating data marts involves varying degrees of data schema change of the existing independent data marts. Depending upon the consolidation strategy you choose for your enterprise, you would need to make schema changes to existing independent silos before they can be integrated into the EDW.

The following sections discuss the ways in which schemas can be designed for the various consolidation approaches.

4.3.1 Simple migration approach

In the simple migration approach of consolidating data, there is no change in the schemas of existing independent data marts. The existing analytical structures are simply moved to a single platform. As shown in Figure 4-10, we see that there is no schema change for the sales and marketing independent data marts.

Archived

In addition to the schema creation process, some of the objects that will be needed to be ported to the consolidated platform are:

- Stored procedures
- Views
- Triggers
- User defined data types
- ETL logic

Note: If the ETL was hand-written, it may make good sense to re-implement it using a modern ETL tool such as WebSphere DataStage rather than to port it.

Figure 4-10 No schema change for the simple migration approach

4.3.2 Centralized consolidation approach

In the centralized consolidation approach, the data can be consolidated in two ways, as described next:

- Centralized consolidation using *redesign:* In this approach we redesign the EDW schema. As shown in Figure 4-11, the different independent data marts are basically merged into the EDW. When we design an EDW, then we may create a normalized or denormalized model. We may use the existing independent data marts for gaining understanding of the business, but we redesign the schema of the EDW independently.

Archived

Figure 4-11 Schema changes in centralized consolidation approach (using redesign)

- Centralized consolidation — Merge with primary technique: In this approach we identify one *primary* data mart among all the existing independent data marts. This primary data mart is then chosen to be first migrated into the EDW environment. All other independent data marts migrated later are conformed according to it. The primary data mart schema is used as the base schema. Other independent data marts are folded into this primary schema, as is done when one enterprise acquires another enterprise. As shown in Figure 4-12, then the best organized of the data marts is assigned the *primary* responsibility around which other data marts are transformed and merged.

Archived

Figure 4-12 Centralized consolidation - Merge with primary

Data schemas

There are basically two types of schemas that can be defined for the data warehouse data model. They are an ER (entity, attribute, relationship) model and dimensional data model. Although there is some debate as to which method is best, a standard answer would be “it depends”. It depends on the environment, the type of access, performance requirements, how the data is to be used, user skills and preferences, and many other criteria. This is probably a debate you have already had and a decision you have already made, as most organizations will have arrived at a preference in this area. In fact, in looking at independent surveys of data warehouse implementations, it is clear that most implementations use a mix of these techniques.

Our position is that there are advantages to both, and they can, and probably should, easily coexist in any data warehousing implementation.

4.3.3 Distributed consolidation approach

In distributed consolidation approach, the information across various independent data marts is consolidated without *physically* integrating the data marts. This is done by restructuring the dimensions in each data mart so that they *conform* with each other. Once the tables all use dimensions that are consistent, then data can be joined between the tables by using data federation technology such as WebSphere II.

The data federation technology allows you to write SQL requests that span multiple databases, and multiple DBMS types. It also allows for the physical access to multiple databases. Clearly, to be joined the data in those databases must be compatible.

As shown in Figure 4-13, the sales and marketing data marts (independent) are conformed for the customer dimension. The sales and marketing data marts schema may require a very minimal change for the conformed customer table present in the two marts. The independent data marts continue to exist in distributed consolidation. There is also not much change in the ETL processes except that the conformed dimensions are managed centrally.

Note: There is no schema change in the distributed consolidation approach. Only certain dimensions are changed to a standardized conformed dimension.

Figure 4-13 Customer table is conformed in the distributed consolidation for sales and marketing data marts

Archived

4.4 Consolidating the other analytic structures

In 4.1, “What are good candidates for consolidation?” on page 68, we have discussed the various analytic structures that are present in enterprises. These analytic structures are candidates for consolidation.

In Table 4-2 , we discuss how various consolidation approaches (see 4.2, “Approaches to consolidation” on page 71) can be used to consolidate the various analytic structures of information. A mix of these approaches can be applied over a period of time to achieve a centralized EDW.

Archived

Table 4-2 Consolidating analytical structures

Consolidation Approaches	Analytical Structures	Description of consolidation process
Simple Migration	Independent data mart	Independent data marts are generally primary candidates for consolidation. Simple migration approach, can be used to consolidate independent data marts from heterogeneous platforms into a single platform. Using the simple migration approach, the data and objects are transferred to the single platform. Data integration is not done. Simple migration alone does not help the enterprise in achieving the single version of the truth. Simple migration approach is generally used as a first step to a broader centralized consolidation approach.
	Data warehouse	We can consolidate two data warehouses using the simple migration approach. Generally two data warehouses are consolidated when a large enterprise acquires a smaller one. In such a scenario, the smaller data warehouse is consolidated into the bigger one. Using the simple migration approach, we move the data and objects of smaller data warehouse into the bigger data warehouse. However, data is not integrated. Simple migration alone does not help the enterprise in achieving the single version of the truth. Simple migration approach is generally used as a first step to broader centralized consolidation approach.
	Dependent data marts	Dependent data marts can be migrated back to the EDW using the simple migration approach. The advantage of moving the dependent data marts are cost reductions in hardware/software, maintenance, staffing requirements and operations. The data quality is assumed to be good because the dependent data marts were fed from the EDW.
	Spreadsheets	We can use the simple migration approach to consolidate the data from several spreadsheets into the EDW. This process would involve creating objects on the EDW to load the data from these spreadsheets, and then providing an alternative front-end presentation/query capability such as Cognos, Business Objects, or DB2 Alphablox. Note that DB2 Alphablox contains components that allow you to construct Java-enabled reports with the same “look and feel” as Excel, but with the data now held securely on a server rather than distributed across many PCs.
	Others	Using the simple migration approach we can consolidate other sources of analytical structures such as Microsoft Access databases, flat files, denormalized databases etc.

Archived

Consolidation Approaches	Analytical Structures	Description of consolidation process
Centralized Consolidation	Independent data marts	Independent data marts can be consolidated using the centralized consolidation approach. There are two way of consolidating: a) Redesign In this approach we create a new EDW schema for the two or more data marts being consolidated. b) Merge with primary In this approach we use one of the main data marts as the primary data mart and merge other data marts into it. We make use of conformed dimensions and conformed facts to ensure that the data is consistent and accurate. To achieve the benefits of centralized EDW, we may use a mix of simple migration along with centralized consolidation approaches.
	Data warehouse	We may consolidate two or more data warehouses or marts in the same way we consolidate two independent data marts. The focus is on metadata standardization and data integration. Centralized consolidation approach helps in achieving the single version of the truth for the enterprise.
	Dependent data marts	When an enterprise is moving the dependent data marts from their existing platforms to the EDW, a simple migration approach is sufficient provided that the quality of data in the dependent data marts is consistent, accurate and conformity of dimensions has been maintained. Generally it is observed that some dependent data marts over a period of time might have inaccurate data due to mismatch in update cycles or wrong ETL logic. In such cases the data in these dependent marts need to be cleansed and conformed before consolidating them into the EDW.
	Spreadsheets	The data from several spreadsheets may be cleansed, conformed, quality assured and loaded into one or more dimensions of the EDW. An example of this process is described in Chapter 5, “Spreadsheet data marts” on page 117.
	Others	Sources such as Microsoft Access and flat files may be consolidated using this approach.

Around

Consolidation Approaches	Analytical Structures	Description of consolidation process
Distributed Consolidation	Independent data marts	In distributed consolidation, we consolidate independent data marts by changing some of their existing dimensions and conforming them to some standard source dimensions. The independent data marts continue to exist on the same hardware/software platform.
	Data warehouse	We can conform two independent data warehouses in the same way we conformed two independent data marts as discussed in the above row.
	Dependent data marts	Dependent data marts are already designed to use conformed dimensions. Data federation allows access to both the EDW and data marts, with joins across them as required (with or without physical consolidation). Note that having conformed dimensions does not mean you can physically join across the databases. For that you need distributed access.
	Spreadsheets	In order to consolidate spreadsheets, we would use centralized consolidation. Distributed consolidation is used to conform independent data marts.
	Others	In order to consolidate sources such as Microsoft Access and flat files, we would use centralized consolidation. Distributed consolidation is used to conform independent data marts. It is highly unlikely that multiple data sources such as these have consistent dimensions, and conforming them is probably as much effort as converting them.

4.5 Other consolidation opportunities

Data mart consolidation brings with it the opportunity to consolidate other enterprise processes and equipment, as well as the data marts themselves.

4.5.1 Reporting environments

As shown in Figure 4-14, we see that with independent data marts, there are typically also independent report applications which are built around each. This is where many of the issues raised regarding data marts are surfaced. That is, you get reports with information that is inconsistent across the enterprise.

Even though there is typically data flowing between the various OLTP systems, as shown in Figure 4-14, the delivery time may not be consistent and so the data in each database will not be consistent. In addition, each reporting system may in

fact get their data independently from each OLTP system. When that happens, it is unlikely that they will take a consistent approach to the calculations. This is because the systems have been developed at different times, by different people, using different tools, and working to different objectives. So we should not be surprised when the reports do not agree.

Figure 4-14 Reporting from independent data marts

And, with the proliferation of independent data marts has also come the proliferation of reporting environments. The management of such reporting applications becomes a redundant, costly, and time-consuming process for the enterprise.

In addition, this means that each data mart may have a report server, security, templates, metadata, backup procedure, print server, development tools and other costs associated with the reporting environment (see Figure 4-15).

Figure 4-15 Consolidating the reporting environment

Once we consolidate our independent data marts into an EDW, we can also benefit from creating a more consolidated reporting environment. This is shown in Figure 4-16.

Figure 4-16 Consolidated reporting from an EDW

Archived

Other concerns of having diverse reporting tools and reporting environments within the same organization are as follows:

- High cost in IT infrastructure both in terms of software and hardware needed to support diverse reporting needs. Multiple Web servers are used in the enterprise to support reporting needs of each independent data mart.
- No common reporting standards.
- Duplicate and competing reporting systems present.
- Multiple backup strategies for the various reporting systems
- Multiple repositories, that are different for each reporting tool.
- No common strategy for security. Each reporting tool builds its own security domain to secure the data of its data mart.
- High cost of training for multiple skill sets.
- High cost of training developers to work with multiple reporting tools, or alternatively coping with the inflexibility of only being able to use certain developers on certain projects due to skills limitations.
- Higher development cost of each report.

Assessing the consolidation impact on reports

We have previously described three approaches to consolidation. Each will have a different impact on the reporting environments. Here, we summarize that impact:

- **Simple migration approach:** There is no impact on existing reports when independent data marts are transferred into a consolidated platform. The only difference is that the existing reports connection needs to be redirected to the new platform, and the necessary data format changes needed to move the data structures to a new DBMS.
- **Centralized consolidation approach:** The concerns here are as follows:
  - With redesign: The existing reports change completely.
  - Merge with primary: The existing reports change completely. However, reports that depend only on the primary mart might not change.
- **Distributed consolidation approach:** There is minimal or no impact on existing reports. The minimal impact comes from some dimensions changing as a result of conformed dimensions being introduced in place of non-conformed dimensions. That would, however, be a significant enabler for new reports that draw their data from multiple databases.

Archived

It is clear that with a data mart consolidation strategy, the reporting environments must also be consolidated. We have observed reductions in the number of reports an order of magnitude in size in specific implementations. This, in itself, is a huge savings in time, money, and resource requirements.

The benefits of consolidating can be summarized as follows:

- Reduced cost of IT infrastructure in terms of software and hardware.
- Reduced cost of report development.
- Single and integrated security strategy for the entire enterprise.
- Single reporting repository for the single reporting solution.
- Elimination of duplicate, competing report systems.
- Faster easier development by having a smaller, simpler consistent environment.
- Greater user productivity by using modern tools, having standardized reports, and having consistent quality data.
- Reduced number of queries executed per month.
- Reduced training costs of developers in learning a single reporting solution in comparison to multiple tools.
- Reduced training cost for business users in learning various reporting tools.
- A common standardization in reports is introduced to achieve consistency across the enterprise, which eliminates delay and indecision over what information should be used for what purpose.
- Reduced number of errors and anomalies in comparison to multiple reporting tools accessing multiple independent data marts.

4.5.2 BI tools

As shown in fig Figure 4-17, there are a number of tools, and categories of tools, that are generally involved with any data mart implementation. Some examples of these tool categories are:

- Databases
- ETL Tool
- Reporting and Dashboard tools
- Data Modeling
- Operating systems also vary depending up the tool being used
- Tools used for software version control
- OLAP Tools for building cubes based on MOLAP, ROLAP, HOLAP structure.
- Project management tools

Archived

Figure 4-17 Tool categories in data mart implementations

Enterprises can begin to standardize on tools in each category as a part of their consolidation project. This can yield huge cost reductions and help the IT department more effectively manage and support the systems.

Issues faced in consolidating tools

Consolidating the above mentioned tools may be easier said than done. Typically there are function and feature differences, and strong preferences among the user communities for tools with which they have gained a high level of competence. There may also be significant technical support expertise on particular tools that needs to be considered.

However, as we have previously mentioned, it is not so much a case of consolidating on one tool. Rather, it may still make sense to consolidate from many to fewer.

Any decision should consider these factors, as well as the long term goals of the enterprise.

4.5.3 ETL processes

The ETL processes involved in consolidating data marts are broadly divided into two steps, as shown in Figure 4-18.

Archived

Figure 4-18 Consolidating independent data marts

- In Step 1, the ETL process is designed to transfer data from the two data marts (sales and inventory) into the EDW.
- In Step 2, the ETL process is designed to feed the EDW directly from the data sources for the sales and inventory data marts. As shown in Figure 4-18, the sales and inventory data marts can be eliminated after consolidation.

It is probably more typical to go directly to step 2 as the only step. However, since it is an option to have both steps we have shown them. In any case, the reports have to be redeveloped before the old data mart can be shutdown.

The report change process is similar to that described for the data. That process flow is depicted in Figure 4-19, in a generic way.

Figure 4-19 Report change process

Archived

Here are the steps for the report change process:

1. Data is extracted from the data mart and loaded into the EDW. Reports continue to come from the data mart.
2. The ETL process is changed to extract data directly from OLTP to the EDW. But the reports continue to come from the data mart.
3. The reports are changed to get data from the EDW. When they are validated by the users, we can shut down the data mart process.
4. The data mart is shut down and the reports now come from the EDW.

The steps mentioned for ETL are valid only for the following two consolidation approaches:

- Simple migration approach
- Centralized consolidation approach:
  - Redesign
  - Merge with primary

Note: In the case of distributed consolidation, only certain dimensions in existing independent data marts are conformed. There is no change in the existing ETL process.

4.6 Tools for consolidation

IBM has a number of tools and products to help consolidate your data mart environment, as shown in Figure 4-20. In this section, we provide a brief overview of some of these tools and their capabilities.

Figure 4-20 Tools used in consolidation effort

4.6.1 DB2 Universal Database

DB2 Universal Database is a database management system that delivers a flexible and cost-effective database platform to build robust on demand business applications. DB2 UDB further leverages your resources with broad support for open standards and popular development platforms such as J2EE and Microsoft. NET. The DB2 UDB family also includes solutions tailored for specific needs like business intelligence and advanced tooling. Whether your business is large or small, DB2 UDB has a solution built and priced to meet your unique needs.

4.6.2 DB2 Data Warehouse Edition

Since we have a focus of data mart consolidation in this redbook, we want to focus on the fact that DB2 supports data warehousing environments of all sizes. IBM provides software bundles (editions) that includes many of the other required products and capabilities for data warehousing implementations. In particular, in this redbook, we will focus on the DB2 Data Warehouse Edition (DWE). It is a powerful business intelligence platform that includes DB2, federated data access, Data Partitioning, integrated online analytical processing (OLAP), advanced data mining, enhanced extract, transform, and load (ETL), workload management, and provides spreadsheet integrated BI for the desktop. DWE works with and enhances the performance of advanced desktop OLAP tools such as DB2 OLAP Server and others from IBM partners. The features included in this edition are:

- DB2 Alphablox, for rapid assembly and broad deployment of integrated analytics. It provides a component-based, comprehensive framework for integrating analytics into existing business processes and systems. The Alphablox open architecture is built to integrate with your existing IT infrastructure, enabling you to leverage existing resources and skill sets to deliver sophisticated analytic capability customized to each individual user and role. We provide more information on this topic in 4.6.5, “DB2 Alphablox” on page 108.
- DB2 Universal Database Enterprise Server Edition is designed to meet the relational database server needs of mid- to large-size businesses. It can be deployed on Linux®, UNIX®, or Windows® servers of any size, from one CPU to hundreds of CPUs. DB2 ESE is an ideal foundation for solutions, such as large data warehouses of multiple terabyte size or high performing, high availability, high volume transaction processing business solutions, and for Web-based solutions.

  It is important to understand that DB2 implements a shared-nothing massively parallel processing model. This is acknowledged to be the best model for ensuring scalability for the large data volumes typical in data warehousing. And, it is implemented in a highly cost effective way.

- DB2 Universal Database, Database Partitioning Feature (large clustered server support). The Database Partitioning Feature (DPF) allows you to partition a database within a single server or across a cluster of servers. It provides benefits including scalability to support very large databases or complex workloads and increased parallelism for administration tasks.
- DB2 Cube Views (OLAP acceleration) is the latest generation of OLAP support in DB2 UDB. It includes features and functions that make the relational database a platform for managing and deploying multidimensional data across the enterprise. With DB2 Cube Views, the database becomes multi dimensionally aware by:
  - Including metadata support for dimensions, hierarchies, attributes, and analytical functions
  - Analyzing the dimensional model and recommending aggregates (such as MQTs - also known as summary tables) that improve OLAP performance
  - Adding OLAP metadata to the DB2 catalogs, providing a foundation for OLAP to speed deployment and improve performance

  Cube Views accelerate OLAP queries by using more efficient DB2 materialized query tables. DB2 MQTs can pre-aggregate the relational data and dramatically improve query performance for OLAP tools and applications. It enables faster and easier development, enables OLAP data to be scaled to much greater volumes, and helps users to share data among multiple tools.
- DB2 Intelligent Miner Modeling, Visualization, and Scoring (powerful data mining and integration of mining into OLTP applications). For example, DB2 Intelligent Miner Modeling delivers DB2 Extenders™ for the following modeling operations:
  - Associations discovery, such as product associations in a market basket analysis, site visit patterns an eCommerce site, or combinations of financial offerings purchased.
  - Demographic clustering, such as market segmentation, store profiling, and buying-behavior patterns.
  - Tree classification, such as profiling customers based on a desired outcome such as propensity to buy, projected spending level, and the likelihood of attrition within a period of time.

  With DB2, data mining can be performed within the database without having to extract the data to a special tool. Therefore it can be run on very large volumes of data.
- DB2 Office Connect Enterprise Web Edition, which provides, as an example, Spreadsheet integration for the desktop. Spreadsheets are used by essentially every business enterprise. A primary issues with spreadsheets is their inability to seamlessly transfer information between the spreadsheet and

Archived

a relational databases — such as DB2. Often users require complex macros to do this. But now DB2 provides it for you. We discuss the integration of spreadsheet data in Chapter 5, “Spreadsheet data marts” on page 117.

- DB2 Query Patroller, for rule-based predictive query monitoring and control. It is a powerful query management system that you can use to proactively and dynamically control the flow of queries against your DB2 database in the following key ways:
  - Define separate query classes for queries for better sharing of system resources and to promote better managed query execution environment.
  - Provide a priority scheme managing query execution.
  - Automatically put large queries on hold so that they can be canceled or scheduled to run during off-peak hours.
  - Track and cancel runaway queries

  In addition, information about completed queries can be collected and analyzed to determine trends across queries, heavy users, and frequently used tables and indexes.
- DB2 Warehouse Manager Standard Edition, for enhanced ETL services and support for multiple agents. It is part of the DB2 Data Warehouse Enterprise Edition, as well as being orderable separately. DB2 Warehouse Manager provides an infrastructure that helps you build, manage, and access the data warehouses that form the backbone of your BI solution.
- WebSphere Information Integrator Standard Edition. In conjunction with DB2 Warehouse Manager it provides native connectors for accessing data from heterogeneous databases, such as Oracle, Teradata, Sybase, and Microsoft SQL server. We discuss this further in 4.6.3, “WebSphere Information Integrator” on page 106.

4.6.3 WebSphere Information Integrator

WebSphere Information Integrator (WebSphere II) provides the foundation for a strategic information consolidation and integration framework that helps customers speed time to market for new applications, get more value and insight from existing assets, and control IT costs. It can reach into multiple data sources, such as Oracle, SQL Server 2000, Teradata, Sybase, Text files, Excel spreadsheets, and the Web.

WebSphere II is designed to meet a diverse range of data integration requirements for business intelligence and business integration, it provides a range of capabilities such as:

- Data transformation
- Data federation
- Data placement (caching and replication)

It provides access to multiple heterogeneous data sources as if they resided on DB2. WebSphere II uses constructs called wrappers to enable access to Relational (such as Oracle, SQL Server, Sybase, and Teradata) and Non-Relational (such as MS-Excel, text files, and XML files) data sources. With the classic edition, WebSphere II can also provide access to multiple types of legacy systems, such as IMS™, IDMS, VSAM, and Adabase.

The structure of WebSphere II is depicted in Figure 4-21.

Figure 4-21 Data federation with WebSphere II

There are different editions of WebSphere Information Integrator available for specific purposes. As examples:

- WebSphere Information Integrator Omnifind Edition
- WebSphere Information Integrator Content Edition
- WebSphere Information Integrator Event Publisher Edition
- WebSphere Information Integrator Replication Edition
- WebSphere Information Integrator Standard Edition
- WebSphere Information Integrator Advanced Edition
- WebSphere Information Integrator Advanced Edition Unlimited
- WebSphere Information Integrator Classic Federation for z/OS®

Please use the following URL for more details regarding each of the editions:

http://www-306.ibm.com/software/data/integration/db2ii/

4.6.4 DB2 Migration ToolKit

The IBM DB2 Migration ToolKit (MTK) helps you migrate from heterogeneous database management systems such as Oracle (versions 7, 8i and 9i), Sybase ASE (versions 11 through 12. 5), Microsoft SQL Server (versions 6, 7, and 2000), Informix (IDS v7. 3 and v9), and Informix XPS (limited support) to DB2 UDB V8. 1 and DB2 V8. 2 on Windows, UNIX and Linux and DB2 iSeries™ including iSeries v5r3. The DB2 Migration ToolKit is available in English on a variety of platforms including Windows (2000, NT 4. 0 and XP), AIX®, Linux, HP/UX and Solaris.

The MTK enables the migration of complex databases with its full functioning GUI interface, providing more options to further refine the migration. For example, you can change the default choices that are made about which DB2 data type to map to the corresponding source database data type. The toolkit also converts to and refines DB2 database scripts. This model also makes the toolkit very portable, making it possible to import and convert on a machine remote from where the source database and DB2 are installed.

MTK converts the following source database constructs into equivalent DB2:

- Data types
- Tables
- Columns
- Views
- Indexes
- Constraints
- Packages
- Stored procedures
- Functions
- Triggers

The MTK is available free of charge from IBM at the following URL:

http://www-306.ibm.com/software/data/db2/migration/mtk/

4.6.5 DB2 Alphablox

Enterprises have long understood the critical role that business intelligence plays in making better business decisions. To succeed, today's enterprises not only need the right information, they need it delivered right at the point of opportunity to all decision makers throughout the enterprise. Integrated analytics help unleash the power of information within customer and partner-developed applications.

DB2 Alphablox is an industry-leading platform for the rapid assembly and broad deployment of integrated analytics embedded within applications. It has an open,

extensible architecture based on J2EE (Java 2 platform, Enterprise Edition) standards, an industry standard for developing Web-based enterprise reporting applications. It is a very cost-effective tool for implementing reports, such as Web delivered balanced scorecards.

DB2 Alphablox for UNIX and Windows adds new capabilities to the IBM business intelligence portfolio, a key foundation for our on demand capabilities:

- It adds a set of components, based on open standards, that allow you to deliver on the vision of integrated analytics.
- It enables you to broaden and deepen business performance management capabilities across enterprises.
- It provides dynamic insight into your respective business environment.
- It allows you to quickly take advantage of new opportunities and overcome challenges while you still have the opportunity to make significant adjustments.

Alphablox differs from more traditional business intelligence solutions, as it addresses issues such as Excel proliferation found in many businesses today. Alphablox provides closed loop analysis allowing business users to update and modify individual data cells and values, providing they hold the relevant security credentials.

Typical business problems addressed include elongated period to close on financial position, proliferation of Excel spreadsheets for budgeting, allocations, forecasts, and actuals. The full potential value of the data warehouse cannot be maximized with incumbent traditional BI or analysis solutions since these products have no closed loop capability or write back functionality. These limitations result in manual collation, aggregation and reconciliation for updates. The manual approach lacks security, data integrity and personalization resulting in many weeks passing before an accurate business position is available. The whole process is disconnected and the Web is not being utilized for fulfilment of such application processes.

An Alphablox solution provides dynamic interfaces for populating and distributing information over the Web. Personalizing the information to the recipient is a key characteristic of such solutions, with security and integrity being key factors considered in the application design. The data relates specifically to the recipient and all fields that a business user cannot change or insert values are locked. Business users have the capabilities to add, edit and delete data on-line via the 100% thin client user interface. All amendments and inserts are immediately written back to either a temporary staging area or directly to the underlying data source. The former optionally encompassing the full ETL process once the new values have been collected. Business user interfaces include upload data facilities and the ability to manage the validation and correction of errors on-line.

User profiles and systems maintenance are managed, reference data is created and edited, and data conversion activities are carried out where appropriate. Alphablox solutions remove the manual collation, aggregation and update overhead providing closed loop analytical solutions. Writeback of changes to the data warehouse are facilitated through this process. Alphablox analytic solutions can therefore provision an application integrated and customized to the customer environment. An essential requirement of the such solutions, is to perform duplicate data checks on the records in the details being submitted, by comparing them against any data already submitted in earlier files for the current (reporting) period.

Data value commentary allows solutions to provide cell commenting (also known as cell annotations) functionality to applications using specifically designed out of box components. Comments are stored in a JDBC accessible relational database.This data source is predefined by the application designer. When the commentary functionality is set up and enabled on a user interface, a Comments menu item becomes available from the right-click menu and a drawing pin indicator appears in the corner of the cells that has comments associated with values. The commentary can be optionally viewed by colleagues, and this saves the cumbersome task of attaching figures to an e-mail and forwarding to team members. Instead, users can remain in the operational application.

The business benefits achieved include reducing the close period typically from weeks to hours, and immediate visibility and accuracy of business position relative to amendments and inserts. More confidence in the numbers is achieved by the business. And from a security standpoint, unlike Excel, Alphablox only enables modification or update to selected fields by recipient. The closed loop Web deployment eliminates the cost and time in the prior manual process.

4.6.6 DB2 Entity Analytics

Although DB2 Entity Analytics is not a tool specifically for data mart consolidation, it is another DB2 tool that can satisfy specific data warehousing requirements. For example, if your organization needs to merge multiple customer files in order to provide the best view of the total customer base, identifying all the unique individuals, then this may be a useful part of your data warehousing solution.

This software allows an enterprise to take multiple data sources and merge them together to build a single entity server and resolve identities. Consolidation of multiple customers from different independent databases is a major task done during the consolidation project.

The DB2 Entity Analytics software helps companies understand a basic question: “Who is who?”. As an example, this software helps in performing analysis such as, “The person with this credit card on data mart A is also the same person with this passport on data mart B, which is the same passport number on data mart C.”

The key feature of this software is that the more data sources you add to it, the more the accuracy tends to go up.

4.6.7 DB2 Relationship Resolution

DB2 Relationship Resolution answers the question “Who Knows Who?” IBM DB2 Relationship Resolution software begins where most solutions leave off, extending the customer view to identify and include the non-obvious relationships among individuals and organizations. An individual's relationships can provide a more complete view of their risk or value to your enterprise, whether they're a customer, prospect, or employee — even if an individual is trying to hide or disguise his or her identity.

DB2 Relationship Resolution has tremendous application in industries such financial services, insurance, government, law enforcement, health care and life sciences, and hospitality. enterprises in these and other industries can use Relationship Resolution to:

- Connect insiders to external threats.
- Find high and low value customer relationships.
- Give fraud detection applications x-ray vision.
- Determine “network” value of the customer.
- Protect customers, employees, and national security.

4.6.8 Others...

In this section we describe various other tools that may be used in the consolidation process:

- IBM WebSphere DataStage may be used to transform, cleanse, and conform data from existing data marts or data sources and load it into the data warehousing environment. This process is commonly called ETL. WebSphere DataStage delivers four core capabilities, all of which are necessary for successful data transformation within any enterprise data integration project.

  These are the core capabilities:

  – Connectivity to a wide range of mainframe, legacy and enterprise applications, databases, and external information sources — to ensure that every critical enterprise data asset can be used.

Archived

- Comprehensive, intrinsic, pre-built library of 300 functions — to reduce development time and learning curves, increase data accuracy and reliability, and provide reliable documentation that lowers maintenance costs.
- Maximum throughput from any hardware investment used in the completion of bulk tasks within the smallest batch windows, and the highest volumes of continuous, event-based transformations using a single high-performance parallel processing architecture.
- Enterprise-class capabilities for development, deployment, and maintenance with no hand-coding required; and high-availability platform support — to reduce on-going administration and implementation risk.

WebSphere DataStage is part of the WebSphere Data Integration Suite, and is integrated with best-of-class data profiling and data quality and cleansing products for the most complete, scalable enterprise data integration solution available.

- WebSphere ProfileStage can be used understand the structure and the content that is stored and held in disparate databases, and then get it ready to re-purpose. It is a data profiling and source system analysis solution that completely automates this all-important first step in data integration, dramatically reducing the time it takes to profile data. ProfileStage also drastically reduces the overall time it takes to complete large scale data integration projects, by automatically creating ETL job definitions that are subsequently run by WebSphere DataStage.
- WebSphere QualityStage provides a broad offering for data standardization and matching of any data. There is a full range of functions to convert data from disparate legacy sources into consolidated high quality information that can be utilized throughout a complex enterprise architecture. Sophisticated investigation processing ensures that all the input data values are strongly typed and placed into fixed fielded buckets and includes complete standardization, verification and certification for global address data.
- Data Modeling tools such as ERWin can connect to many existing database systems and document their structure and definitions. It creates both a logical and physical model from any existing database. All tables, indexes, views, code objects (where applicable), along with other metadata, is captured and stored in the ERWin Repository, which allows data modeling to be performed on the consolidated target platform.

4.7 Issues with consolidation

When consolidating data marts from diverse heterogeneous platforms we may come across the following issues:

- Inadequate business and technical metadata definitions across the data marts.
- Data quality issues may require changes to business processes as well as IT data handling processes.
- Resistance to consolidation. Almost every data mart consolidation project faces some degree of cultural resistance.
- Lack of sufficient technical expertise on the EDW platform as well as on some of the, particularly old and perhaps obsolete, existing data mart platforms.
- Performance and scalability is always an issue. Care must be taken in selection of tools and products to assure expectations can be met. This may require development of new, and agreed to, service level agreements.
- Lack of a strong business sponsor. The effort must be part of a strategic and supported business direction.
- Reliability and time cycle data management issues. It is important to have synchronized update cycles for fact tables which are using common conformed dimensions. In order to understand this let us take an example of sales and inventory business process. The sales data mart has a fact table “sales” whereas as an inventory data mart has fact table named “inventory”. Both these data marts share a common dimension “product”.

  Now assume that product called “product-1” has sold under category “Dairy” until January 25th, 2005. On January 26th, 2005, the product called “product-1” is moved under category “Cheese”. The business needs to maintain the history of sales of “product-1” for all sales prior to January 26th, 2005. So it inserts a new row for the “product-1” under the category “Cheese”. This is shown in Table 4-3.

  It is important that the two fact tables for sales and inventory business should start using the surrogate key equal to “2” as shown in Table 4-3 simultaneously from January 26th, 2005. Any updates to fact tables should ensure that both the businesses report the present state of “product-1”.

Table 4-3 Sample Product table

Product_ID (Surrogate)	Product_OLTP	Product Name	Category
1	9988	Product-1	Dairy
2	9988	Product-1	Cheese

- Security. After the consolidation of data marts, it is very important to be certain that security rules are still valid regarding who can see what data.
- Operational considerations. Depending upon the SLAs (service level agreements), for activities such as data loads and backups/restore are current and understood.

4.7.1 When would you not consider consolidation?

Every enterprise can typically benefit by consolidating their data marts and analytic data silos. However there may be situations where the current environment is not conducive to consolidation. In these situations it may serve you well to wait until appropriate actions have been taken to correct those situations.

Here are a few examples of situations where it may not be advisable to start a data mart consolidation project:

- Lack of strong business sponsor:

  Before starting a data mart consolidation project, it is very important to have executive sponsorship. These projects can cross functional lines and require changes, as an example, in the data definitions to enable development of common terms and definitions, as well as changes to the physical data environment structure. It will typically also require changes to the business processes and sharing of data. This will require negotiations and inter-departmental cooperation. These can be difficult issues to resolve without good executive support.
- Quality of data:

  Prior to starting a consolidation project, the quality of data must be analyzed and understood. Where the data is not of high quality, it is not advised to begin such a project. The better approach is to define a project aimed at cleansing the data, but also to correct those conditions leading to the poor data quality. This can be a project that needs to be complete prior to beginning data consolidation, or at least one that runs concurrently. One of the considerations here is the cost and time required to correct the data quality issues. Typically these situations, and their potential costs, have already been considered and factored into the decisions, prior to moving to a consolidation effort.
- Metadata management:

  Good standardized, centralized, metadata management is one of the keys to a successful data consolidation project. With such management, hopefully any data quality issues would be minimal. At least any issues revolving around the integration of data across the enterprise would have been identified and resolved.

- Support from the business functional areas:

  Support from these important areas may also be a determinant of success in a data consolidation project.

Although these are situations where consolidation might not be advised, the answer is not to avoid consolidation. The answer is to first address these issues!

4.8 Benefits of consolidation

We discuss a number of the benefits of data consolidation throughout this redbook. Here we provide a high level summary of the opportunities for improvement by data mart consolidation, and best practices for doing so. They are listed by category in Table 4-4.

Table 4-4 Summary of data mart consolidation benefits

System Level	Opportunities to improve with consolidation	Best Practice
User Productivity	1. Time finding data and reports. 2. Time determining quality and meaning of the data.	1. Users have set of reports tailored to their needs. 2. Reports simple to invoke, and use modern query tools.
Reports	1. Many reports that are not clear and not current. 2. Unclear which data is/was used by which reports.	Standard set of reports.
Development Methods	1. No standard development methods. 2. User staff develops inefficient reports. Maintenance difficult, and lacks documentation.	Center of excellence (COE) prepares reports with modern tools and skilled staff.
Query Tools	1. Multiple tools, no volume discounts. 2. Developers only work with specific systems. 3. Tools not specific to tasks.	1. Ad Hoc facilities available to users with support from COE. 2. Standard set of 1-3 query tools.
ETL	1. Hand written ETL processes. 2. Redundant processes for data sources. 3. Uncertainty about what data is available.	1. Standard modern ETL tool. 2. Non-redundant ETL

Archived

System Level	Opportunities to improve with consolidation	Best Practice
Data Model	1. Multiple disjoint data models used. 2. Hard to join data across systems. 3. Expensive and time consuming to develop new reports.	1. Corporate data model in place. 2. Central system with dependent data marts all conforming to standard data model.
DBMS	1. Multiple DBMS, no commercial leverage. 2. Multiple support costs. 3. Not suited to BI.	DBMS suited to large BI system on DB2.
Operating System	1. Multiple operating systems.	
Hardware Platforms		

Archived

5

Spreadsheet data marts

In this chapter we discuss the importance of managing and controlling your spreadsheet data as an enterprise asset. The key topics are included are:

- Spreadsheet usage in enterprises:
  - Developing standards for spreadsheets
- Consolidating spreadsheet data:
  - Storing spreadsheet data in DB2
  - Transferring spreadsheet data to DB2 using XML conversion
  - Direct data transfer from spreadsheets to DB2
  - Consolidating spreadsheet data using DB2 OLAP Server
- Sample scenarios using IBM WebSphere Information Integrator:
  - Accessing Excel data from DB2 using WebSphere II
- Sample scenarios using IBM DB2 Warehouse Manager:
  - Transferring Excel data into DB2 using DB2 Warehouse Manager

5.1 Spreadsheet usage in enterprises

Spreadsheets play an important role in most enterprises, for analysis and decision making. The power of the spreadsheet is in the wide range of analytic capabilities and ease of use, and thus understanding by a wide range of people.

© Copyright IBM Corp. 2005. All rights reserved.

It is a tool that requires little training for the end users. However, spreadsheets can be very expensive! Although the spreadsheet software may be inexpensive, the manual overhead of developing and reconciling spreadsheets is very expensive in terms of time.

There are many who regularly use a spreadsheet for analysis in their decision making process. Many managerial surveys have indicated that spreadsheets are the preferred tool for guiding them in their decision making process. And, spreadsheets are used in most enterprises at all levels.

However, in many cases, the strategic decision makers do not know the source of the data they are using to support their decisions. Worse yet, they do not know what has been done to it before it was given to them. This is a significant issue.

So the question arises as to whether the spreadsheets in enterprises can be trusted sources. Consider these questions, as examples:

- Is the data reliable?
- Who created it?
- Under what conditions?
- Who manages and validates it?
- Who can see and use the data?

That last question can pose a major issue, because, without management or control, spreadsheets can become what are called data silos — meaning data that is not shared across the enterprise, and that may actually be inconsistent with other data in the enterprise.

This is an ongoing issue of major concern to enterprise management. It is one of the primary contributors to the scenario where management sees two reports from two organizations within the enterprise, concerning the same issue, and they are not in agreement. Whom does management believe? How can they make a decision based on either of the reports? This is one of the issues that needs to be addressed and corrected.

5.1.1 Developing standards for spreadsheets

There are many ways of analyzing data on spreadsheets. And, the spreadsheet may not always depict the entire picture of the analysis that was done. For example, some part of the analysis may be in the mind of the person who created it. So when such spreadsheets are presented to another individual, it is very difficult to interpret the analytical purpose and figures contained in the spreadsheet. So a major issue involves just how to make the spreadsheets understandable!

Since spreadsheets are typically designed by individuals, and for their particular purpose, it is in the hands of those individuals to set their own standards in creating the spreadsheets. This in itself can lead to misinterpretations of the data, and lack of consistency in their use.

Another major issue is how to be sure the data presented in the spreadsheet is an accurate representation of the business. The typical complete lack of control in developing spreadsheets makes it difficult to be sure they are accurate.

To gain control of these situations and create consistency in interpretation, it would be wise for the enterprise to develop standards for their development and use. Let us look at how a sample spreadsheet might be modified to enable a wider audience to understand the data it represents.

Figure 5-1 shows an example spreadsheet containing data for analysis. But, who can interpret what it means? There is not enough information, or standardization of content, to enable anyone to understand it.

1	23154.34	85769.65	75433.73	63612.75
2	6656.78	7456.36	45224.64	2362.76
3	4458.32	475682.8	85889.75	63462
4	8685.4	45568.54	573457.6	86546.75
5	3556.76	48546.35	235737.8	24682.75
6	7587.87	24243.23	743283.6	76438.78

Figure 5-1 Unstructured spreadsheet

Figure 5-2 shows a spreadsheet with column and row identifiers included. We may have a better idea, but it is still not clear exactly what the data represents. The identifiers help us to understand that the rows represent states and the columns represent monthly data. But, we still do not know what the data actually represents.

Archived

	A	B	C	D	E	F
1		**January**	**February**	**March**	**April**	**May**
2	**Alabama**	23154.34	85769.65	75433.73	63612.75	97678
3	**Atlanta**	6656.78	7456.36	45224.64	2362.76	87578.37
4	**California**	4458.32	475682.75	85889.75	63462	7868.74
5	**Chicago**	8685.4	45568.54	573457.62	86546.75	88432.46
6	**Cleveland**	3556.76	48546.35	235737.76	24682.75	56576.27
7	**Kentucky**	7587.87	24243.23	743283.57	76438.78	87464.34

Figure 5-2 Spreadsheet with column and row identifiers

Figure 5-3 takes the spreadsheet data to another level of understanding because it now includes a heading label that clearly defines what the data represents.

	A	B	C	D	E	F
1	**Almaden Book shop - Monthly sales figure of books by state**					
2		**January**	**February**	**March**	**April**	**May**
3	**Alabama**	23154.34	85769.65	75433.73	63612.75	97678
4	**Atlanta**	6656.78	7456.36	45224.64	2362.76	87578.37
5	**California**	4458.32	475682.75	85889.75	63462	7868.74
6	**Chicago**	8685.4	45568.54	573457.62	86546.75	88432.46
7	**Cleveland**	3556.76	48546.35	235737.76	24682.75	56576.27
8	**Kentucky**	7587.87	24243.23	743283.57	76438.78	87464.34

Figure 5-3 Spreadsheet representing books sales figures by state

The examples in this section demonstrate a simple scenario that emphasizes the importance of implementing standards for spreadsheets. They can enable a more general understanding of the spreadsheet data. The data then becomes a more valuable asset to the enterprise. And, it enables use of the data by a wider audience of users. Enabling this more common understanding of the data can also contribute to the ability to consolidate it and manage the use and distribution, which can enhance the value to the enterprise.

It is very important to note, however, that even though the spreadsheet is now more understandable, there is still no explicit connection between the titles and values shown, and corporate data. The author of the spreadsheet is free to input and amend any values they please.

5.2 Consolidating spreadsheet data

In this and the following sections we describe some useful techniques for consolidating spreadsheets. As examples:

- DB2 Connect™
- DB2 OLAP Server, for easy access to consolidated data from Excel
- DB2 Alphablox for server-based and Java-based presentation of server-based data with Excel-like “look and feel”.
- XML
- WebSphere Information Integrator for two-way access between DB2 and Excel

After we have done some standardization to make the information in spreadsheets understandable on a more common basis, we need to make it available to others in the enterprise. Today, the disparate data contained in most spreadsheets serves only the purpose of the individual who created it. And, it typically only resides on the personal workstation of the creator. That needs to change.

There are several ways to make data more accessible. For example, it could be hosted on a central server that is easily accessible by those who need it. Access for update capability could be controlled and managed to assure data quality. And, that server should be managed by IT. That way, the administrative work to maintain it could be off-loaded from the business analyst. In reality, most of the administrative work is probably not being done. By administrative work, we refer to such activities as backup, recovery, performance tuning, data cleansing, and data integrity.

Consolidating spreadsheet data is a widely debated topic, primarily because it would typically require changes to many business operations. However, this is something that needs to be addressed. By consolidating the data and making it available throughout the enterprise, it would be of much greater value. But, how can it be done?

A good choice as a consolidation platform is DB2. This is one approach that can enable the data to be shared across the enterprise by those authorized to do so. There are various tools and technologies available in IBM, as well as IBM business partners, that can enable the process of easily storing and retrieving the data.

As you may know, the data in spreadsheets has a different representation than data stored in a relational database. Therefore, we will need to convert the data before moving it from a spreadsheet to DB2. In the following sections we discuss several methods for accomplishing this task. We also include a discussion on the process of converting and moving data from DB2 back to a spreadsheet format.

Archived

5.2.1 Using XML for consolidation

One means of converting and moving data from spreadsheets to a DB2 database is by first converting the spreadsheet data to XML format and then loading the XML data to the DB2 database. XML has been accepted as a standard method for exchanging data across heterogeneous systems, and DB2 has many built-in functions to support the XML format of data transfer. There are also a number of IBM tools and technologies available for the purpose of converting data from XML format to the DB2 database. As examples, Java APIs and VBA/VB scripts. Though the spreadsheet can be directly converted to an XML document, the output XML file may not be of the desired format for reading the data into the relational database.

XML-enabled databases, such as DB2, typically include software for transferring data from XML documents. This software can be integrated into the database engine or external to the engine. As examples, DB2 XML Extender, the WebSphere II - XML Wrapper, and SQL/XML can all transfer data between XML documents and the DB2 database. The DB2 XML Extender and XML Wrapper are external to the database engine, while SQL/XML support is integrated into the DB2 database engine.

Note: The primary advantage of using a DB2 database is that it keeps existing data and applications intact. That is, adding XML functionality to the database is simply a matter of adding and configuring the software that transfers data between XML documents and the database. There is no need to change existing data or applications.

For further information regarding XML support in DB2, please refer to the IBM Redbook, *XML for DB2 Information Integration*, SG24-6994.

Solution overview

The solution involves conversion of spreadsheet documents to an intermediate XML format and then loading the data into the DB2 database. Figure 5-4 shows the general idea behind the solution.

Figure 5-4 Solution overview

The advantage of using this approach is that it is generic and is not dependent on any specific vendor. Here the DB2 database serves as an XML-enabled database. In an XML-enabled database existing data can be used to create XML documents, a process known as publishing. Similarly, data from an XML document can be stored in the database, a process known as shredding.

This scenario is depicted in Figure 5-5. In an XML-enabled database, no XML is visible inside the database and the database schema must be mapped to an XML schema. XML-enabled databases are used when XML is used as a data exchange format and requires no changes to existing applications.

Figure 5-5 Shredding and Publishing

Important: Figure 5-5 shows shredding and publishing an XML document:

- Shredding is the process of transferring data from the XML documents to relational tables in a database.
- Publishing is the process of creating XML documents by reading the data from the relational tables and is the reverse process of shredding.

Table 5-1 shows the IBM products available for the process of shredding and publishing XML documents.

Table 5-1 Product overview

Task	Product
Publishing data as XML (Composition)	SQL/XML
	XML Extender
	Write your own code
Shredding XML documents (Decomposition)	XML Extender
	XML Wrapper (WebSphere Information Integrator)
	Write your own code

Archived

SQL/XML

For XML-enabled relational databases, the most important query language is SQL/XML, which is a set of extensions to SQL for creating XML documents and fragments from relational data. It is part of the ISO SQL specification (Information technology - Database languages - SQL - Part 14: XML-Related Specifications (SQL/XML) ISO/IEC 9075-14:2003). SQL/XML support can be found in DB2 UDB for Linux, UNIX, and Windows, and in DB2 for z/OS V8.

SQL/XML adds a number of new features to SQL. The most important of these are a new data type (the XML data type), a set of scalar functions for creating XML (XMLELEMENT, XMLATTRIBUTES, XMLFOREST, and XMLCONCAT), and an aggregation function (XMLAGG) for creating XML. It also defines how to map database identifiers to XML identifiers.

DB2 XML Extender

The XML Extender is a DB2 Extender that provides both XML-enabled and native XML capabilities for DB2. The XML-enabled capabilities are provided through XML collections, while native XML capabilities are provided through XML columns. The DB2 XML Extender consists of a number of user-defined data types, user-defined functions, and stored procedures. These must be installed in each database (for DB2 for z/OS that is a DB2 subsystem or a data sharing group) on which they are used. This process is known as enabling the database for XML use. The DB2 XML Extender is shipped with DB2 UDB for Linux, UNIX, and Windows, V7 and later. In version 7, it is installed separately, while in version 8, it is installed as part of DB2 (although you still have to enable the database for XML use). The XML Extender is also a free, separately installable, component of DB2 for z/OS V7 and later.

XML Wrapper

The XML Wrapper is shipped with WebSphere II. The XML Wrapper provides XML-enabled capabilities for WebSphere II by treating an XML document as a source of relational data. In XML terms, it shreds a portion of the XML document according to an object-relational mapping and returns the data as a table. Note that XML Wrapper queries are potentially expensive because the XML Wrapper must parse each document it queries. Thus, to query a large number of documents, or if you frequently query the same XML document, you may want to shred these documents into tables in your database if possible.

Local and Global XML schemas

One issue with using XML-enabled storage is that you are often constrained as to what XML schemas your documents can use. When the XML schema does not match the database schema, you will need two XML schemas: a local XML schema and a global XML schema.

The local XML schema is used when transferring data to and from the database, and must match the database schema.

The global XML schema is used by the applications, as well as to exchange data with other applications or databases. It might be an industry-standard schema, or a schema that all external users of your XML documents have agreed upon.

When using local and global XML schemas, the application must transform incoming documents from the global schema to the local schema before storing the data in those documents in the database. The application must also transform outgoing documents from the local schema to the global schema after those documents have been constructed from data in the database.

To convert between these two schemas, the application generally uses XSLT. That is, it uses XSLT to convert incoming documents from the global XML schema to the local XML schema. Similarly, it converts outgoing documents from the local XML schema to the global XML schema. This is shown in Figure 5-6.

Figure 5-6 Transforming XML documents between local and global schemas

There are several ways to transform XML documents. These are:

- **XSLT:** This is the most common way to transform XML. Refer to:

  http://www.w3.org/TR/xslt

  The advantage of XSLT is that it is a standard technology and is widely available. Furthermore, it only requires you to write XSLT style sheets, not code, in order to transform documents. The disadvantage of XSLT is that it can be slow and may need to read the entire document into memory. The latter problem prohibits its use with very large XML documents.

- **Custom SAX applications:** If your transformation is simple and can be performed while reading through the document from start to finish, then you might be able to write a simple SAX program to perform the transformation. The advantage of SAX is that it is generally faster than XSLT. Furthermore, it does not read the entire document into memory, so it can be used with arbitrarily large documents.

- **Third-party transformation packages:** Some third-party packages are available for performing specific types of transformations. For example, the Regular Fragmentations package uses regular expressions to create multiple elements from a single element. For example, you can use this to create Year, Month, and Day elements from a date element. Please see:

  http://regfrag.sourceforge.net/

Of course, if you can use a single XML schema, as is generally the case when using SQL/XML, and sometimes the case when using the XML Extender or the XML Wrapper, then you should use only a single XML schema. The reason is that transformations can be expensive, so your application will generally perform better without them.

Solution

As we have seen in the above sections, there are several tools and technologies available in order to construct a solution for centralizing spreadsheet data. It entirely depends on the enterprise to choose which one suits their need and is appropriate for the environment. When arriving at the solutions, there are lot of considerations, such as the current systems set up, maintenance, cost, and many other factors. Figure 5-7 shows an outline of the proposed solution. Each activity in the architecture, such as transforming the spreadsheet to XML and back, converting global XML schema to local XML schema, shredding, and publishing, can be achieved using combinations of various tools and custom built applications using APIs.

The basic idea behind the architecture is to consolidate spreadsheet data from the clients into a central database so that the intelligence in the spreadsheet can be shared across the enterprise. The architecture also has the provision to transform relational data back to spreadsheet data for further analysis.

An important consideration is whether the enterprise is willing to minimize spreadsheet analysis and is seeking a more centralized access model for future purposes. If this is the case, the data transfer from the spreadsheets on the client machines to the relational database will be a one way process that is the reverse transformation of relational database to spreadsheets, and can be avoided. Once the data is moved to the relational database, further analysis of data can be done using reporting tools such as Business Objects, Cognos, or any of the many others available. However, it is only an option, and the enterprise must decide on setting its own standards.

Figure 5-7 Solution architecture - Centralizing spreadsheet data using XML conversion

These are the functional components of the architecture shown in Figure 5-7:

- Transformation of spreadsheet data to XML format
- Transformation of global XML schema to local XML schema
- Mapping the local XML schema to the relational tables in DB2
- Publishing data from the DB2 database back to local XML schema format
- Transformation of local XML schema to global XML schema
- Transformation of global XML schema back to spreadsheet data
- Transforming data directly from DB2 database to spreadsheets

Transforming spreadsheet data to XML format

The spreadsheet data can be transformed to XML formats using third party tools or VBA/VB scripts. Though there may be an option to save the spreadsheet document in XML format, it might not be of the desired structure and further conversion would be complex. Hence using a tool or in-house developed application that uses APIs, would result in a better formatted XML file.

Transforming global XML schema to local XML schema

The transformation of global XML schema to the local XML schema can be done using XSLT, SAX, or any custom built application using XML APIs. The Global XML schema is structured more like the source. For example, the spreadsheet file and data cannot be mapped directly from the global schema to the relational tables. Hence the global schema is converted to a format (a local schema) which can be easily deciphered when reading data into the relational tables.

Mapping the XML schema to DB2

The local XML schema is structured in a manner in which the data from the local schema can be easily mapped to the columns of the relational table in the database. For this reason, an ideal local schema should have a table, row, and column format.

The process of reading data in the XML document and writing into the corresponding relational table column is known as *shredding*. The tools with which we can shred the data into a DB2 database are WebSphere II (XML wrapper), MQ Series, and the DB2 XML extender. Custom built applications using APIs are also an effective method of shredding an XML document.

The XML statement shown in `Example 5-1` is constructed out of the spreadsheet shown in Figure 5-3. It clearly defines the purpose of the data. The table name and column names are well defined with row identifiers. The structure of the XML document is very easy to interpret by any application and the data can be easily transferred into the relational tables in the database.

Example 5-1 XML generated for the spreadsheet shown in Figure 5-3

```
<monthly_book_sales>
     <city name="Alabama">
         <Month="Jan">23154.34</Month>
         < Month="Feb">85769.65</Month>
         < Month="March">75433.73</Month>
         < Month="April">63612.75</Month>
     </city>
      (etc…)
 </monthly_book_sales>
```

Publishing data from DB2 to a local XML schema

To allow provision to move the relational data back into the spreadsheet, the process has to be reversed. Publishing the data from the relational database back into XML format can done easily using DB2 SQL/XML - which is a set of extensions to SQL for creating XML documents and also using tools such as DB2 XML extender, or using custom built applications within the APIs.

Transforming the local XML schema to global XML schema

The local XML schema should be transferred back to the global XML schema in order to easily export the data back to the spreadsheet format. This can be done using XSLT/SAX or custom built application using XML APIs. The reverse process is usually easy, as it would require less improvising on the existing process of transformation.

Transforming the global XML schema back to a spreadsheet

The last step in the reverse process is to transform the global XML schema back into the spreadsheet. The existing programs used for other conversion processes can be improvised to transform the global XML schema back to spreadsheets, or by using the import data utility of the spreadsheet.

Transforming data directly from DB2 to the spreadsheet

Alternatively the data from the DB2 database can be copied into the spreadsheet file by querying using DB2 SQL/XML extensions to create a XML file, and then import the XML data into the spreadsheet using an import data utility.

The other easy way of importing data from DB2 to a spreadsheet is to connect to the DB2 database directly using functionality such as an OLE DB driver. Following are the steps to import DB2 data into a spreadsheet using an OLE DB driver.

1. Open the spreadsheet. Select **Data** → **Import External Data** → **Import Data**.
2. Select **Connect to New Data Source.odc** and click **New source**.
3. Select **Other/Advanced** and click **Next**.
4. Scroll through the list of OLE DB providers and select **IBM OLE DB provider for DB2** and then click **Next**. The Data link properties window opens.
5. Select the **Existing data source** radio button and the name of the database that you want to connect to using the **Data Source** drop-down box. Also specify values for a **User Name** and **Password** that you want to use for your database connection. You can also use the **Direct server connection** option to discover a database server that is associated with your selected OLE DB driver on a specific system.
6. Test the connection by clicking **Test connection,** and click **OK**.
7. In the data connection wizard, select the table that contains the data you want to import into a DB2 database and click **Finish**.
8. Now select where you want to put the DB2 data. Whether in the existing worksheet, in a specific range, or in a new worksheet.

5.2.2 Transferring spreadsheet data to DB2 with no conversion

The following alternative solution does not use XML conversion. The alternative solution reads data directly from the spreadsheet and writes into the database. In the case of using XML transformation, the spreadsheet document was first converted to an XML global schema format and then to the local schema format.

This conforms to the base table structure in the relational database so that data can be easily mapped from the Local XML document to the relational tables in the database. But when we try to read data directly from the spreadsheet document into the database, the application should be robust enough to handle the transformation and mapping of the spreadsheet data to the relational table structure in the database.

Solution overview

The solution is focussed on different alternative approaches that can be used to implement an architecture that uses IBM tools and technologies to centralize spreadsheet data in the enterprise without having any intermediate conversion of data format. The solution for each enterprise would differ based on the amount of data and standards implemented in the enterprise. Many tools and technologies available in the market can be used for the architecture. Custom built applications using APIs are the most effective method for developing such a solution as the application would be designed to meet the requirements of the enterprise.

Solution

The proposed solution is based on two IBM data integration tools, though there are many alternatives. For example, it could be in-house developed applications for reading data from the spreadsheet and writing into the database. The proposed solutions have to be implemented in conjunction with custom built applications to achieve best results.

Figure 5-8 shows how data can be read from spreadsheets using a combination of IBM and third party tools, such as Openlink, for reading data from the spreadsheets and writing into the DB2 database. The spreadsheets are copied from the client to the file server using either FTP/Copy script. The spreadsheet files in the file server are then accessed by executing an SQL query in the DB2 database, which refers to nicknames created on the centralized spreadsheet files in the file server. Since the spreadsheet files are on Windows and the database is on AIX, where there is no spreadsheet ODBC driver, we have to install a multi-tier ODBC driver. In this architecture we use an Openlink multi-tier ODBC client on AIX and Openlink multi-tier ODBC server on the file server to access the Excel files.

If we consider keeping the database and the centralized spreadsheet files on the same Windows server, then we do not require a multi-tier ODBC driver for accessing the spreadsheet files using WebSphere II.

Figure 5-8 Centralizing spreadsheets using WebSphere II

Another available solution is to use the DB2 Warehouse Manager for transferring data from the spreadsheets to the DB2 database. The spreadsheet files from the client, as in the previous solution, have to be copied to a centralized file server, where the DB2 Warehouse Manager is installed. Then we can create steps to transfer the data from the spreadsheet files to the DB2 database. The data warehouse steps will use the warehouse agent on the file server. Figure 5-9 shows an architecture where the DB2 Warehouse Manager is used for populating data from the file server to the database server.

Figure 5-9 Centralizing spreadsheets using DB2 Warehouse Manager

Conclusion

As mentioned earlier, this solution where there is no intermediate conversion of data would require the application to control the actions of tools such as DB2 Warehouse Manager or WebSphere II, to have robust features to process the data to the relational table format. The application, for example, if written in Java,

should trigger the data warehouse steps, in the case of DB2 Warehouse Manager, or query on the nicknames in case of WebSphere II, with embedded logic to process the data from the spreadsheet source files. This solution might not be suitable for enterprises where the spreadsheet analysis process is not standardized, as the source spreadsheets have to be in a required format for processing. In both cases the data from the DB2 database can be transformed back into spreadsheet format by simply using the OLE DB driver.

5.2.3 Consolidating spreadsheet data using DB2 OLAP Server

The DB2 OLAP Server is a multidimensional database with a proprietary format of data storage. The analytic services multidimensional database stores and organizes data. It is optimized to handle applications that contain large amounts of numeric data and that are consolidation-intensive or computation-intensive. In addition, the database organizes data in a way that reflects how the user wants to view the data.The OLAP server has specialized features for multidimensional reporting using spreadsheets.

As an example, the Essbase XTD Spreadsheet Add-in is a software program that merges seamlessly with Microsoft Excel. After Analytic Services is installed, a special menu is added to the spreadsheet application. The menu provides enhanced commands such as Connect, Pivot, Drill-down, and Calculate. Users can access and analyze data on Analytic Server by using simple mouse clicks, and drag-and-drop operations.Spreadsheet Add-in enables multiple users to access and to update data on OLAP Server concurrently, shown in Figure 5-10.

Figure 5-10 Centralizing spreadsheets using DB2 OLAP Server

As part of centralizing spreadsheet data, the spreadsheet files can be copied to the server hosting the OLAP server, and then converted into a CSV (comma separated values) file format, and then loaded into the multidimensional database using the ESSCMD command-line interface or Essbase application manager.

The spreadsheet add-in can then be used in the client system to connect to the multidimensional database for direct multidimensional reporting.

5.3 Spreadsheets and WebSphere Information Integrator

WebSphere II supports spreadsheets from Excel 97, Excel 2000, and Excel 2002. You can configure access to Excel data sources by using the DB2 Control Center or by issuing SQL statements.

Figure 5-11illustrates how the Excel wrapper connects the spreadsheets to a federated system.

Figure 5-11 WebSphere II - Excel wrapper

5.3.1 Adding spreadsheet data to a federated server

To configure the federated server to access spreadsheet data sources, you must provide the federated server with information about the data sources and objects that you want to access. You can configure the federated server to access spreadsheet data sources by using the DB2 Control Center or the DB2 command line. The DB2 Control Center includes a wizard to guide you through the steps required to configure the federated server.

Prerequisites:

These are the prerequisites that must be in place:

- WebSphere II must be installed on a server that acts as the federated server.
- A federated database must exist on the federated server.
- Excel worksheets must be structured properly so that the wrapper can access the data.

Procedure:

Use this procedure to add the spreadsheet data sources to a federated server:

- Register the Excel wrapper.
- Register the server for Excel data sources.
- Register the nicknames for Excel data sources.

Registering the spreadsheet wrapper

Registering the Excel wrapper is part of the larger task of adding spreadsheet data sources to a federated server. You must register a wrapper to access spreadsheet data sources. Wrappers are used by federated servers to communicate with and retrieve data from data sources. Wrappers are implemented as a set of library files.

Restrictions:

These are the restrictions that apply:

- The Excel wrapper is available only for Microsoft Windows operating systems that support DB2 UDB Enterprise Server Edition.
- The Excel application must be installed on the server where WebSphere II is installed before the wrapper can be used.
- Pass-through sessions are not allowed.

Procedure:

To register a wrapper, issue the **CREATE WRAPPER** statement with the name of the wrapper and the name of the wrapper library file. For example, to register a wrapper with the name excel_wrapper, issue the statement shown in Example 5-2:

Example 5-2 Create wrapper statement

```
CREATE WRAPPER excel_wrapper LIBRARY 'db2lsxls.dll';
```

You must specify the wrapper library file, db2lsxls.dll, in the CREATE WRAPPER statement.

When you install WebSphere II, the library file is added to the directory path. When registering a wrapper, specify only the library file name that is listed in Table 5-2 .

Table 5-2 Wrapper library location and file name

Operating system	Directory path	Wrapper library file
Windows	%DB2PATH%\bin	db2lsxls.dll

%DB2PATH% is the environment variable that is used to specify the directory path where WebSphere II is installed on Windows. The default Windows directory path is C:\Program Files\IBM\SQLLIB.

Table 5-3 lists the DB2 data types supported by the Excel wrapper.

Table 5-3 Excel data types that map to DB2 data types

Excel data type	DB2 data type
Date	DATE
Number	DOUBLE
Number	FLOAT (n) where n is >= 25 and <= 53
Integer	INTEGER
Character	VARCHAR

Registering the server for spreadsheet data sources

Registering the server for spreadsheet data source is part of the larger task of adding Excel to a federated system. After the wrapper is registered, you must register a corresponding server. For Excel, a server definition is created because the hierarchy of federated objects requires that data source files (identified by nicknames) are associated with a specific server object.

Procedure:

To register the Excel server to the federated system, use the **CREATE SERVER** statement. Suppose that you want to create a server object called biochem_lab for a spreadsheet that contains biochemical data. The server object must be associated with the spreadsheet wrapper that you registered using the **CREATE WRAPPER** statement.

The **CREATE SERVER** statement to register this server object is shown in Example 5-3.

Example 5-3 Create server statement

```
CREATE SERVER biochem_lab WRAPPER excel_wrapper;
```

Registering the nicknames for Excel data sources

Registering nicknames for Excel data sources is part of the larger task of adding Excel to a federated system. After you register a server, you must register a corresponding nickname. Nicknames are used when you refer to an Excel data source in a query.

Procedure:

To map the Excel data source to relational tables, create a nickname using the **CREATE NICKNAME** statement. The statement in Example 5-4 creates the compound nickname from the spreadsheet file named CompoundMaster.xls. Example 5-4 contains three columns of data that are being defined to the federated system as Compound_ID, CompoundName, and MolWeight.

Example 5-4 Create NICKNAME command

```
CREATE NICKNAME Compounds ( Compound_ID INTEGER, CompoundName VARCHAR(50),
MolWeight FLOAT) FOR SERVER biochem_lab OPTIONS (FILE_PATH
'C:\data\CompoundMaster.xls', RANGE 'B2:D5');
```

These are the CREATE NICKNAME options:

- File path
- Range

File path

This specifies the fully qualified directory path and file name of the spreadsheet that you want to access. Data types must be consistent within each column and the column data types must be described correctly during the register nickname process.

The Excel wrappers can only access the primary spreadsheet within an Excel workbook. Blank cells in the spreadsheet are interpreted as NULL. Up to 10 consecutive blank rows can exist in the spreadsheet and be included in the data set. More than 10 consecutive blank rows are interpreted as end of the data set.

Blank columns can exist in the spreadsheet. However, these columns must be registered and described as valid fields even if they will not be used. The database codepage must match the file's character set; otherwise, you could get unexpected results.

Range

This specifies a range of cells to be used in the data source, but this option is not required.

In Table 5-4, B2 represents the top left of a cell range and D5 represents the bottom right of the cell range. The letter B in the B2 designation is the column designation. The number 2 in the B2 representation is the row number.

The bottom right designation can be omitted from the range. In this case, the bottom right valid row is used. If the top left value is omitted, then the value is taken as A1. If the range specifies more rows than are actually in the spreadsheet, then the actual number of rows is used.

5.3.2 Sample consolidation scenario using WebSphere II

This section demonstrates a sample implementation of the WebSphere II V8.2 Excel wrapper, installed with DB2 V8.2 on Windows and accessing an Excel 2000 worksheet located in the *C:\Data directory* of the same server. The scenario registers the wrapper, registers a server, and registers one nickname that will be used to access the worksheet. The statements shown in the scenario are entered using the DB2 command line. After the wrapper is registered, queries can be run on the worksheet.

The scenario starts with a compound worksheet, called Compund_Master.xls, with 4 columns and 9 rows. The fully-qualified path name to the file is *C:\Data\Compound_Master.xls*. Table 5-4 shows an example Excel worksheet to be used for the sample scenario.

Table 5-4 Sample worksheet Compound_Master.xls

	A	B	C	D
1	Compound_name	Weight	Mol_Count	Was_Tested
2	compound_A	1.23	367	tested
3	compound_G		210	
4	compound_F	0.000425536	174	tested
5	compound_Y	1.00256		tested
6	compound_Q		1024	
7	compound_B	33.5362		
8	compound_S	0.96723	67	tested
9	compound_O	1.2		tested

Procedure to access a spreadsheet worksheet

Follow this procedure:

1. Connect to the federated database.
2. Register the Excel_2000 wrapper:

   db2 => **CREATE WRAPPER** Excel_2000 LIBRARY 'db2lsxls.dll'
3. Register the server:

   db2 => **CREATE SERVER** biochem_lab WRAPPER Excel_2000

Archived

4. Register a nickname that refers to the Excel worksheet:

```
db2 => CREATE NICKNAME Compound_Master (compound_name VARCHAR(40), weight
FLOAT, mol_count INTEGER, was_tested VARCHAR(20)) FOR biochem_lab OPTIONS
(FILE_PATH 'C:\Data\Compound_Master.xls')
```

The registration process is complete. The Excel data source is now part of the federated system, and can be used in SQL queries.

Example 5-5 is an example of an SQL query retrieving compound data from the Excel data source.

Example 5-5 Sample query - data

```
Sample SQL query: "Give me all the compound data where mol_count is greater
than 100."

SELECT * FROM compound_master WHERE mol_count > 100

Result: All fields for rows 2, 3, 4, 6, and 8.
```

Example 5-6 is an example of an SQL query retrieving a compound name.

Example 5-6 Sample query - name

```
Sample SQL query: "Give me the compound_name and mol_count for all compounds
where the mol_count has not yet been determined."

SELECT compound_name, mol_count FROM compound_master WHERE mol_count IS NULL

Result: Fields compound_name and mol_count of rows 5, 7, and 9 from the
worksheet.
```

Example 5-7 is an example of an SQL query retrieving a count.

Example 5-7 Sample query - count

```
Sample SQL query: "Count the number of compounds that have not been tested and
the weight is greater than 1."

SELECT count(*) FROM compound_master WHERE was_tested IS NULL AND weight > 1

Result: The record count of 1 which represents the single row 7 from the
worksheet that meets the criteria.
```

Archived

5.4 Data transfer example with DB2 Warehouse Manager

The IBM DB2 Warehouse Manager (WM) is an ETL tool used for transforming and moving data across different data sources. For example, WM data from Excel can be transferred to any DB2 database irrespective of the operating system. The data transferred from the Excel file is stored in relational table format in the DB2 database. As such, data from disparate Excel sources can be consolidated into a DB2 database. The following initial steps are required for accessing data from Excel sources:

1. Preparing the Excel file.
2. Setting up connectivity to the source file.
3. Setting up connectivity to the target DB2 database.

5.4.1 Preparing the source spreadsheet file

If using the Microsoft Excel ODBC driver to access the Excel spreadsheets, you need to create a named table for each of the worksheets within the spreadsheet.

Procedure to create named tables

Follow this procedure:

1. Select the columns and rows to include in the table.
2. Click **Insert** → **Name** → **Define**.
3. Verify that the *Refers to field* in the Define Name window contains the cells that you have selected in step 1. If not, click the icon on the far right of the *Refers to field* to include all the cells that you selected.
4. Type a name (or use the default name) for the marked data.
5. Click **OK**.

5.4.2 Setting up connectivity to the source file

After creating a spreadsheet to use as a data warehouse source, you catalog the source in ODBC so that you can access it from the Data Warehouse Center.

Procedure to catalog a Microsoft Excel spreadsheet in ODBC

Follow this procedure:

1. Click **Start** → **Settings** → **Control Panel**.
2. Double-click ODBC.
3. Click **System DSN**.

Archived

4. Click **Add**.
5. Select **Microsoft Excel Driver** from the Installed ODBC Drivers list.
6. Click **OK**.
7. Type the spreadsheet alias in the Data Source Name field.
8. Optional: Type a description of the spreadsheet in the Description field.
9. Select **Excel 97-2000** from the Version list.
10. Click **Select Workbook**.
11. Select the path and file name of the database from the list boxes.
12. Click **OK**.
13. Click **OK** in the ODBC Microsoft Excel Setup window.
14. Click **Close**.

5.4.3 Setting up connectivity to the target DB2 database

The target DB2 database can be on any operating system. The database has to be cataloged as an ODBC data source in the system which has the Excel worksheet. The DB2 client configuration assistant can be used to catalog the database. The client configuration assistant creates an ODBC entry under system DSN. After setting up the connectivity to the database, create a table in the database to hold the data from the Excel worksheet. Before creating the table, the data types of Excel should be mapped to the column data types of the table to be created. Table 5-3 shows the mapping between the Excel and DB2 data types. `Example 5-8` shows the create table syntax for data to be populated from the sample Excel worksheet in Table 5-4 .

Example 5-8 Create table syntax

```
CREATE TABLE Compound_Master (compound_name VARCHAR(40),
weight FLOAT,
mol_count INTEGER,
was_tested VARCHAR(20));
```

5.4.4 Sample scenario

In this section we demonstrate a sample implementation of DB2 Warehouse Manager accessing data from an Excel spreadsheet. The scenario includes configuring the Excel worksheet as sources and the DB2 database on AIX as target, and then transferring the data from Excel to the target database by defining a Warehouse step.

The scenario starts with a compound worksheet, called *Compund_Master.xls*, with 4 columns and 9 rows. The fully-qualified path name to the file is

C:\Data\Compound_Master.xls. Table 5-4 shows a sample Excel worksheet to be used for the sample scenario.

Configure the Excel worksheet as Warehouse source

The configuration involves preparing the Excel file as in section 5.4.1, “Preparing the source spreadsheet file” on page 139, and setting up connectivity to the source Excel file as in section5.4.2, “Setting up connectivity to the source file” on page 139. After the initial set up, the Excel worksheet *Compound_Master.xls* has to be configured as a warehouse source in DB2 Warehouse Manager.

Procedure to create a data warehouse source:

1. Click **Start** → **Programs** → **IBM DB2** → **Business Intelligence tools** → **Data Warehouse Center**.
2. Log in to the Data Warehouse Center.
3. Right-click **Warehouse sources** and select **Define** → **ODBC** → **Generic ODBC**.
4. In the *Warehouse source* tab, specify the appropriate name for the Excel source. Choose the default agent site.
5. The Data source tab should be filled in with information, such as Data Source Name (ODBC name), System name, or IP address of the local host where the Excel file is located, and the Userid/Password. Figure 5-12 shows the Data source tab filled in with the required information.

Figure 5-12 DB2 Warehouse Manager - Data source tab information

6. The **Tables and Views** tab lists the table structures defined in the Excel work sheet.
7. Expand the **Tables** node to select the *Compound_Excel table*.
8. Select the table and click the **>** button to move table to the selected list.
9. Click **OK**, and the Excel source is successfully defined. Right-click the **Compound_Excel** table and select **Sample contents** to see the Excel Worksheet data. Figure 5-13 shows the Compound_Excel table under the Excel source and the data from the Excel worksheet.

Figure 5-13 Sample data from the source Excel spreadsheet

Configure the DB2 database on AIX as a Warehouse target

The DB2 database to which the data from the Excel worksheet is to be transferred is referred to as the data warehouse target database. The table *Compound_Master*, created in `Example 5-8`, is the target table to be populated.

Procedure for creating a Warehouse target:

Follow this procedure:

1. Right-click the data warehouse target and select appropriate operating system version of the DB2 database. In our example we use a DB2 database on AIX. Select **Define** → ODBC → **DB2 Family** → **DB2 for AIX**.

2. In the Database tab, specify the target database name, system name, or IP address and Userid/Passwd. Figure 5-14 shows the entries for the Database tab for the sample scenario.

Figure 5-14 Target Database tab information

3. Click the **Tables** tab to expand the table node and select the target table. In this case we select the table **Compound_Master**.
4. Click **OK** to complete the table selection. You can see that the table appears in the right pane of the DB2 Warehouse Center.
5. Right-click the table **Compound_Master** and select sample contents. There are no records displayed, as the table is empty.

Creating a Warehouse step for data transfer

The data from the Excel worksheet table *Compound_Excel,* defined as a Warehouse source, has to be transferred to the target table *Compound_Master* in the DB2/AIX database which is defined as a Warehouse target.

Procedure for creating a Warehouse step:

Follow this procedure:

1. Before creating a Warehouse step, you have to create a Subject area and Process for the Warehouse step. Right-click **Subject Areas** and select **Define**. Then define the name of the subject area as *Data transfer*.

Right-click **Processes**, under the newly created subject area, and define the name as *Excel to DB2/AIX*.

2. Right-click the process **Excel to DB2/AIX** and open the process model. Click the icon to select the source and target for the Warehouse step. Figure 5-15 shows the icon for selecting the source and the target.

Figure 5-15 Icon in process model for selecting the source and target

In the sample scenario, the source table would be *Compound_Excel* from the Excel source, and the target table is *Compound_Master* from the DB2/AIX target as shown in Figure 5-16.

Figure 5-16 Source and target tables

3. Click the SQL icon in the Process model, then click **SQL Select and insert** to create a warehouse step for selecting data from the source Excel file and inserting data into the target DB2 database.

4. Click the **Link tools** icon, as shown in Figure 5-17.

Figure 5-17 Data link icon for creating the warehouse step

5. The data link is then used to draw the link from the source to the Warehouse step, and the warehouse step to the target as shown in Figure 5-18.

Figure 5-18 Warehouse step

Archived

6. Right-click the **SQL** step and select properties, then click the **SQL Statement** tab. Click the **Build SQL** button and select the **Columns tab** to select the required columns from the table defined in the source Excel worksheet.

   The **Build SQL** page also has other options for joining multiple Excel worksheets, grouping, and ordering data.

   For the sample scenario, we select all the columns listed to be mapped to the target table. Click **OK** to close the Build SQL page.

7. Select the **Column Mapping** tab in the properties sheet and map the source Excel table columns to the Target DB2 table. Figure 5-19 shows the column mapping between the source and target tables. Click **OK** to close the properties page.

Figure 5-19 Source to target column mapping

8. Right-click the **SQL** step and select **Mode** → **Test** to promote the step.
9. Right-click the **SQL** step and select **Test** to populate the target DB2 table with the data from the from the Excel table.
10.. Right-click the Target table **Compound_Master** and select sample contents to check data in the table as shown in Figure 5-20.

Archived

Figure 5-20 Sample data from the target table

The sample scenario demonstrated the transfer of data from Excel file to a relational table in a DB2 database. The Excel data can now be analyzed by running select statements on the DB2 table containing the excel data. The data from Excel can also be transformed using Warehouse transformers before populating the table in the DB2 database.

Further information

You can find more information about the topics discussed in this chapter in the following materials:

- *XML for DB2 Information Integration*, SG24- 6994.
- *Data Federation with IBM DB2 Information Integrator V8.1*, SG24-7052.
- *Data Warehouse Center Administration Guide Version 8.2*, SC27-1123.

Archived

Archived

6

Data mart consolidation lifecycle

In this chapter we discuss the phases and activities involved in data mart consolidation (DMC). DMC is a process, and one that could have quite a lengthy duration. For example, a 6-18 month duration would not be unusual. This is particularly true in those situations where there have been:

- Significant proliferation of data mart (and other analytic structures)
- Acquisitions and mergers, requiring integration of heterogeneous data sources, data models, and data types
- Development by multiple non-integrated IT organizations within an enterprise
- Little attention to metadata standardization
- Changes in the product mix developed by the enterprise

In practice, our observation is that these circumstances apply to most enterprises.

But, where do you start? What do you do first? To help in that decision process, we have developed a Data Mart Consolidation Lifecycle. We take a look at that lifecycle in the next section.

© Copyright IBM Corp. 2005. All rights reserved.

6.1 The structure and phases

The data mart consolidation lifecycle has been developed to help identify the phases and activities you will need to consider. It is depicted in Figure 6-1. We discuss those phases and activities in this section to aid in the development of your particular project plans.

To give you a better example, we have used this lifecycle as a guide in the development of this redbook. You will see the phases and activities discussed as they apply to other chapters. We also used it to help in the example consolidation project we have documented in Chapter 9, “Data mart consolidation: A project example”.

Figure 6-1 Data mart consolidation lifecycle

A data mart consolidation project will typically consist of the following activities:

- **Assessment:** Based on the findings in the assessment phase, the “DMC Assessment Findings” report is created. This yields a complete inventory and assessment of the current environment.
- **Planning:** After reviewing the “DMC Assessment Findings” report, we enter the planning phase. As a result of those activities, the “Implementation Recommendation” report is created. This specifies how the consolidation is to be carried out, and estimates resource requirements.
- **Design:** Now we are ready to design the project and the system and acceptance tests, using the recommendations from the planning phase.

- **Implementation:** This consists of the actual implementation of the project.
- **Testing:** At this point, we are ready to test what we have implemented. Not much is documented regarding this phase, the assumption being it is similar to any other implementation testing activity - and a familiar process to IT.
- **Deployment:** This is much the same story as with testing. That is, IT is quite familiar with the processes for deploying new applications and systems.
- **Continuation:** This is not really a phase, but simply a connector we use to document the fact that this may be an ongoing process, or a process with numerous phases.

6.2 Assessment

During this assessment phase we are in a period of discovery and documentation. We need to know what data marts and other analytic structures exist in the enterprise — or that area of the enterprise that is our current focus.

It is at this time that we need to understand exactly what we will be dealing with. So we will need to document, as examples, such things as:

- Existing data marts
- Existing analytic structures
- Data quality
- Data redundancy
- ETL processes that exist
- Source systems involved
- Business and technical metadata, and degree of standardization
- Reporting, and ad hoc query, requirements
- Reporting tools and environment
- BI tools being used
- Hardware, software, and other tools

In general, we want to make an inventory of every object, whether it is code or data, that will have to be transformed as part of the project.

6.2.1 Analytic structures

As we have seen in previous chapters, there are several types of analytic structures present inside any enterprise. The larger the enterprise, the higher the number of analytic silos that are present. These analytic structures store information which is too often redundant, inconsistent, and inaccurate. During the data mart consolidation process we need to analyze these various analytic structures and identify candidates that may be suitable for consolidation.

Some of the analytic structures that we analyze are:

- **Data warehouses:** There may be more than one data warehouse in an enterprise due to such things as acquisitions and mergers, and numerous other activities.
- **Independent data marts:** These are typically developed by a business unit or department to satisfy a specific requirement. And, they are typically funded by that business unit or department — for many different reasons.
- **Spreadsheets:** These are very valuable tools, and widely used. However, we need to share the information and make it available to a larger base of users.
- **Dependent data marts:** These are developed from the existing enterprise data warehouse. As such, the data is more reliable that many of the other sources, but there are still issues; for example, data currency.
- **Operational data stores:** Although the data here has been transformed, it is still operational in nature. The question is, how is it used?
- **Other analytic structures:** We consider structures such as databases, flat files, and OLTP databases that are being used for reporting.

We need to understand these various analytic structures before we can decide whether or not they are candidates for consolidation. To help with this process, we have developed a table of information for evaluating analytic structures. This is depicted in Table 6-1.

Table 6-1 Evaluating analytic structures

Parameter	Description
Name of analytic structure	The name of the analytic structure, which could be any of these: -Data warehouse -Independent data marts -Dependent data marts -ODS -Spreadsheets -Other structures such as databases, flat-files, OLTP systems
Business process	The business process using the analytic structure.
Granularity	The level of detail of information stored in the analytic structure.
Dimensions	The dimensions that define the analytic structure.
Facts	The facts contained in the analytic structure.
Source systems	The source systems used to populate the structure.
Source owners	Organization that owns the data source.
Analytical structure owner	Organization owning the analytic structure.

Parameter	Description
Reports being generated	Reports being generated from the data, number and type.
Number of tables, files, entities, attributes, and columns	Yields a measure of data complexity, which is a prime determinant of project effort and resource requirements.
Size of data volume in GB	A measure of the data volume to be accommodated.
Input data volatility (records/day)	A measure of current volume of data storage.
Number of ETL routines	To assess complexity of data feed processes to modify.
Data quality	Level of quality of the source data.
Scalability	The scalability of the system.
Number of business users	Total number of business users accessing data.
Number of reports produced and percent not being used	The total number of reports being generated by the analytic structure.
Annual maintenance costs	The annual costs to maintain the analytic structure.
Third party involvement	It is important to understand if their is a third party contract for maintaining an existing analytic structure.
Annual Hardware/software costs	Annual costs for hardware and software.
Scalability of the analytic structure	The ability of the system to scale with growth in data and number of users.
Technical complexity	The more complex the system (volume of complex objects being used), the more effort to train IT maintenance staff and users.
Trained technical resources	Availability of trained resources is a critical requirement for maintaining the data quantity, quality, integrity and availability.
Business influence to resist change	Politics and strong enterprise influence is always a factor in the ability to consolidate. In such scenarios, one good way to introduce consistency in data is by standardizing these independent data marts by using, or introducing, conformed dimensions. The control of these data marts may remain with the business organization, rather than IT, we can improve the data consistency.
Return on investment (ROI)	Quantify the ROI, by business area or enterprise.
Age of the analytic structure	Older systems are typically hosted on technologies for which technical expertise is rare and maintenance cost is high.

Archived

Parameter	Description
Lease expiration	We need to identify any forthcoming lease expiration for hardware, software, or third party contract associated with the analytic structures. Generally, old hardware/software systems that have a contract expiration date are prime candidates for consolidation.
Performance or scalability problems	We need to identify any existing problems being faced by these analytic systems due to growth in data or number of users accessing the system.
Data archiving	Data archiving rules for the analytic structure.
Metadata	The metadata includes both technical and business metadata.
Data refresh and update	Criteria used for refreshing and updating data.
ETL	This includes knowing the following things: - Whether ETL is handwritten or created using a tool - Complexity of ETL - Error handing inside ETL - Transformation rules - Number of ETL processes
Physical location	The physical location of analytic structures.
Network connectivity	This specifies how the analytic structure is connected to the IT infrastructure. For very large companies, the data marts may be spread across the globe and connectivity may be established using Virtual Private Network (VPN).
% Sales increase (Value-add to business)	This basically specifies the value-add of the analytic structure to the business. We basically identify the time-span of the analytic structure. As an example, let us say that an independent data mart has been in service since the last 5 years. We need to identify the business value this data mart has provided for the last 5 years. It need not be an exact figure, but it should mention to some extent how the business has been conducted with this structure in place.

6.2.2 Data quality and consistency

We have discussed some of the various analytic structures and some of the attributes and issues with each. There are many that need to be considered, and that may be overlooked initially. To start, we assess the quality of data in the existing analytic structures based on the parameters identified in Table 6-2.

Table 6-2 Data quality assessment

Parameter	Description
Slowly changing dimensions	The dimension metadata must be maintained over time as changes occur. This is critical to track and audit continuity, and maintain validity with historical data.
Dimension versioning and changing hierarchies	Another metadata issue that must be addressed is to track changes to the data model, and relate those changes to the current and historical data, to maintain data integrity.
Consistency	Common data, such as customer names and addresses as examples, should be consistently represented. The data present in the analytic structures should be consistent with the business rules. As an example, a home owners policy start date should not be before the purchase date of the home.
Completeness of data	The data should be completely present as per the business definition and rules.
Data timeliness	Data timeliness is becoming more and more important as a competitive edge. Typically data marts, particularly independent data marts, can experience data latency issues, and irregular and inconsistent update cycles.
Data type and domain	The data should be defined according to business rules. Analyze the data types to see if business conformance is met. There is some data that should be within a particular range to be reasonable. For example a value of 999 in the age of a home owner would be outside any reasonable criteria.

Quality: Slowly changing dimensions

There are various issues in maintaining data quality, which must be addressed. Most are well understood, but must be planned for and a strategy adopted in how to address them. For example, consider the parameter in Table 6-2, *slowly changing dimensions*. How do you maintain those changes? Consider the following discussion.

In a dimensional model, the dimension table attributes are not fixed. They typically change slowly over a period of time, but can also change rapidly. The dimensional modeling design team must involve the business users to help them

determine a change handling strategy to capture the changed dimensional attributes. This basically describes what to do when a dimensional attribute changes in the source system. A change handling strategy involves using a surrogate (substitute) key as its primary key for the dimension table.

We now present a few examples of dimension attribute change-handling strategies. They are simplistic, and do not cover all possible strategies. We just wanted to better familiarize you with the types of issues being discussed:

1. **Type-1:** Overwrite the old value in the dimensional attribute with the current value. Table 6-3 depicts an example involving the stored employee data. For example, it shows that an employee named John works in the Sales department of the company. The Employee ID column uses the natural key, which is different from the surrogate key.

Table 6-3 Employee table - Type-1

S-ID	name	Department	City	Join Date	Emp ID
1	John	Sales	New York	06/03/2000	2341
2	David	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Los Angeles	03/05/1992	7872

Assume that John changed departments in December 2004, and now works in the Inventory department. With a Type-1 change handling strategy, we simply update the existing row in the dimension table with the new department description as shown in Table 6-4:

Table 6-4 Employee table - Type-1 changed

S-ID	name	Department	City	Join Date	Emp ID
1	John	Inventory	New York	06/03/2000	2341
2	David	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Los Angeles	03/05/1992	7872

This would give an impression that John has been working for the Inventory department since the beginning of his tenure with the company. The issue then with a Type-1 strategy is that history is lost. Typically, a Type-1 change strategy is used in those situations where mistake has been made and the old dimension attribute must simply be updated with the correct value.

2. **Type-2:** Insert a new row for any changed dimensional attribute. Now consider the table shown in Table 6-5 with the employee data. The table shows that John works in the Sales department.

Table 6-5 Employee table - Type-2

S-ID	name	Department	City	Join Date	Emp ID
1	John	Sales	New York	06/03/2000	2341
2	David	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Los Angeles	03/05/1992	7872

Assume that John changed departments in December 2004, and now works in the Inventory department. With a Type-2 strategy, we insert a new row in the dimension table with the new department description, as depicted in Table 6-6. When we insert a new row, we use a new surrogate key for the employee John. In this scenario it now has a value of '4'.

Table 6-6 Employee table - Type-2 changed

S-ID	name	Department	City	Join Date	Emp ID
1	John	Sales	New York	06/03/2000	2341
2	David	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Los Angeles	03/05/1992	7872
4	John	Inventory	New York	06/03/2000	2341

The importance of using a surrogate key is that it allows changes to be made. Had we just used the Employee ID as the primary key of the table, then we would have not been able to add a new record to track the change because there cannot be any duplicate keys in the table.

3. **Type-3:** There are two columns to indicate the particular dimensional attribute to be changed. One indicating the original value, and the other indicating the new current value.

   Assume that we need to track both the original and new values of the department for any employee. Then we create an employee table, as shown in Table 6-7, with two columns for the capturing the current and original department of an employee. For an employee just joining the company, both current and original departments are the same.

Table 6-7 Employee table - Type-3

S-ID	name	Original Dept	New Dept	City	Join Date	Emp ID
1	John	Sales	Sales	New York	06/03/2000	2341
2	David	Sales	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Marketing	Los Angeles	03/05/1992	7872

Assume that John changed department from Sales to Inventory in December 2004. We simply update the original department column the value Sales, and the current department column with Inventory. This is depicted in Table 6-8.

Table 6-8 Employee table - Type-3 changed

S-ID	name	Original Dept	New Dept	City	Join Date	Emp ID
1	John	Sales	Inventory	New York	06/03/2000	2341
2	David	Sales	Sales	San Jose	05/27/1998	1244
3	Mike	Marketing	Marketing	Los Angeles	03/05/1992	7872

Note that the Type-3 change does not increase the size of the table when new information is updated, and this strategy allows us to at least keep part of the history.

A disadvantage is that we will not be able to keep all history when an attribute is changed more than once. For example, assume that John changes his department again in January 2005 from Inventory to Marketing. In this scenario, the Sales history would be lost.

IBM has a comprehensive data quality methodology that can be used to put in place an end-to-end capability to ensure sustainable data quality and integrity.

Data integrity

Maintaining data integrity is imperative. It is particularly important in business intelligence because of the wide-spread impact it can have. Experience indicates that as an initiative increases the enterprise dependence on the data, the integrity issues quickly begin to surface. High data quality optimizes the effectiveness of any initiative, and particularly business intelligence.

The enterprise needs a foundation for data integrity, which should be based on industry best practices. IBM has a white paper on the subject, titled “Transforming enterprise information integrity”, that can be found at the following Web site:

http://www.ibm.com/services/us/bcs/pdf/g510-3831-transforming-enterprise-information-integrity.pdf

The IBM enterprise information integrity framework recognizes that information integrity is not solely a technology issue, but that it arises in equal measures from process and organizational issues. It endeavors to achieve and sustain data quality by addressing organization, process, and technology. That framework is depicted in Figure 6-2.

Figure 6-2 IBM information integrity framework

And, IBM has a methodology that captures the process for designing and implementing the IBM enterprise information integrity framework. It is comprised of five phases, as depicted in Figure 6-3.

Archived

Figure 6-3 Information integrity methodology

Leveraging data is critical to competing effectively in the information age. Enterprises must invest in the quality of their data to achieve information integrity. That means fixing their data foundation through a systematic, integrated approach.

For more information on information integrity, and a detailed description of the framework components and the methodology, refer to the IBM white paper.

6.2.3 Data redundancy

Data mart consolidation moves data from disparate, non-integrated analytic structures and can put it in a more centralized, integrated EDW. Here, data is defined once, and as a standard. Data redundancy is minimized or eliminated, which can result in improved data quality and consistency. And, it can also result in fewer systems (servers and storage arrays), leading to lower maintenance costs.

It is important to identify analytic structures that store redundant information. A simple way of identifying redundant information is by using a matrix method.

For example, you can list the EDW tables horizontally and all the other analytic structures (such as independent data marts, dependent data marts, spread sheets, and denormalized databases) vertically, as shown in Figure 6-4.

	Data Mart #1				Data Mart #2			….. Data Mart # N				
Existing EDW Tables	Product (2 Years)	Stores	Revenue (Last 5 Yrs)	Customer	Revenue (4 Years)	Employee	Calendar	Supplier	Calendar	Product (2 Years)	Revenue (10 Years)	Supplier
Product	X									X		
Vendor								X				X
Calendar							X		X			
Currency												
Customer_Shipping												
Merchant												
Merchant_Group												
Class												
And More ⋮												

Figure 6-4 Identifying redundant sources of data

You could, for example, use a cross symbol to identify common data existing in the various analytic structures. In Table 6-9 they are identified as Data Mart #1, Data Mart #2. . . . Data Mart #N, and the EDW.

To see if the data is redundant, you obviously need to analyze it. For example, observe that the Data Mart #1 has the last two years of revenue data, Data Mart #2 has the last four years of revenue data, and Data Mart #N has the last ten years of revenue data. The EDW, on the other hand, consists of revenue information for the all the years the enterprise has been in business.

6.2.4 Source systems

You can assess the various source systems in the enterprise based on the parameters such as those shown in Table 6-9. Responses to this assessment can determine metadata standardization, currency and consistency issues, and compliance with business rules.

Archived

Table 6-9 Source system analysis

Parameter	Description
Name	Name of the source system
Business owner	The source may be owned and controlled by a department, business unit, or the enterprise IT organization.
Maintenance	Organization responsible for performing systems maintenance.
Business process	Description of the business process using this source system.
Time-span of source system	Total time-span for which the system has existed. This will help when tracking data integrity, metadata standardization, and dimension and data integrity over time.
Lease expiration	Determine hardware and software lease expiration dates to help prioritize actions and maximize potential savings.
Upgrades	Understand which systems are planned to be upgraded. This can help prioritize when, or if, an upgrade should occur and can impact the consolidation dates.

Note: Decisions on potential consolidation candidates and implementation priorities can be impacted by understanding software/hardware associated with systems whose licenses are about to expire.

6.2.5 Business and technical metadata

You need to study the existing analytic structures with the help of their business and technical metadata. The business and technical metadata is defined next:

- Business metadata helps users identify and locate data and information available to them in their analytic structure. It provides a roadmap to users to access the data warehouse. Business metadata hides technological constraints by mapping business language to the technical systems.

  We assess business metadata, which generally includes data such as:

  - Business glossary of terms
  - Business terms and definitions for tables and columns.
  - Business definitions for all reports
  - Business definitions for data in the data warehouse

- Technical metadata includes the technical aspects of data, such as table columns, data types, lengths, and lineage. It helps us understand the present structure and relationship of entities within the enterprise in the context of a particular independent data mart.

  The technical metadata analyzed generally includes the following items:

  - Physical table and column names
  - Data mapping and transformation logic
  - Source System details
  - Foreign keys and indexes
  - Security
  - Lineage Analysis that helps track data from a report back to the source, including any transformations involved.

6.2.6 Reporting tools and environment

In 4.5.1, “Reporting environments” on page 96, we discussed in detail the reporting environment infrastructure usually associated with each data mart.

Figure 6-5 shows that each independent data mart typically has its own reporting environment. This means that each data mart may have its own report server, security, templates, metadata, backup procedure, print server, and development tools, that comprise the costs associated with that environment.

Figure 6-5 Reporting environment for analytic structures

Next, assess the various analytic structures on the basis of their reporting infrastructure needs. Some of the parameters to be assessed are shown in Table 6-10.

Table 6-10 Reporting environment by analytic structure

Parameter	Description
Name of analytic structure	This specifies the name of the analytic structure using the reporting tool.
Name of reporting tool	Name of the tool.
Type of analyses supported	Ad hoc, standard, Web based, statistical, data mining, and others.
Number of developers and users	This impacts maintenance costs and determines level of support needed to satisfy requirements.
Number of reports	Determines migration and maintenance costs for reports.
Number of new reports created per month	Determines on-going development workload for creating new reports.
Tool status	This specifies whether this is a standard tool which is extensively used or is a non-standard tool that may have been purchased by the department or business process owner. Typically, it is observed that non-standard tools are more difficult to consolidate because business users have developed confidence using them and show resistance to any change. The same resistance to standardization of the tool is shown by technical users who have developed expertise supporting the tool.
Name of tool vendor	Name of enterprise that sells the tool.
Number of Web servers used	Number of Web servers used to host the reports.
Print servers	Number of print servers being used.
Maintenance cost	Total annual cost of maintenance.
Training cost	Total cost of training business and technical users.
Metadata database	Database used for reporting tool metadata management.
Report server	Report server where reporting tool is installed.
Availability issues	Issues relating to downtime and availability.
Satisfaction level	This specifies the business users overall satisfaction level with using the reporting tool.
License expiration date	This specifies the license expiration date of the reporting tool.

Reporting needs

Now you need to assess all existing reports that are being generated by the various analytic systems we assessed in 6.2.1, “Analytic structures” on page 151.

Existing reports can be assessed based on the following parameters:

- Name of analytic structure (could be independent or dependent data mart or others assessed in 6.2.1, “Analytic structures” on page 151)
- Number of reports and names
- Business description and significance of report
- Importance of report
- Frequency of report
- Business users profile accessing the reports:
  - Normal business user
  - Very important business user
  - Senior management
  - Board of management
  - CIO or CEO

In Table 6-11 we see some of the sample data that can be used for assessing reports.

Table 6-11 Assessing the reports

Report number	Name	Business description and significance	Analytic structure	Importance	Frequency
1	Sales Report by Region	Sales report for the retail stores	Sales Mart #1 (independent data mart)	High	Weekly
2	Inventory by Brand	Inventory report by brand on a store basis	Inventory #1 (independent data mart)	High	Bi-Weekly
3	Inventory by Dairy Products	Inventory levels of dairy products on daily basis for each store	Inventory #2 (independent data mart)	High	Daily
4	Revenue for Stores	Revenue report for every store for all products	Sales Mart #2 (independent data mart)	Medium	Monthly

Archived

6.2.7 Other BI tools

As shown in Table 6-6, there are several tools that are generally involved with any data mart. Some examples of these tools are:

- Databases, query/reporting
- ETL tool
- Dashboard
- Statistical analysis
- Specific applications
- Data modeling tools
- Operating systems also vary depending up the tool being used
- Tools used for software version control
- OLAP tools for building cubes based on MOLAP, ROLAP, HOLAP structures, data mining, and statistical analysis
- Project management tools

Figure 6-6 Other tools associated with analytic structure

Each independent analytic data structure is assessed for all tools that are associated with it. This is shown in Table 6-12.

Archived

Table 6-12 Other tools

Parameter	Description
Analytic structure	Specifies the name of the analytic structure.
Data modeling tool	Name of data modeling tool.
ETL tool	Name of ETL tool.
Dashboard	Name of dashboard tool.
Database	Name of database tool.
OLAP tool	Name of OLAP tool.
Client tools	Name of client tools involved to access analytic structure. Mostly these are client side applications of the reporting tool. Also a Web browser is used.
Version control	Software used for version control.
Operating system	Name of operating system.
Project management tool	Name of project management tool.
Others	Other specific tools purchased.

Note: It is important to look for license expiration dates for all software involved. In some cases it may be helpful to use an existing tool rather than purchasing a new one. But there should also be a focus on standardizing tools.

6.2.8 Hardware/software and other inventory

It will be necessary to track other information relating to inventory and other hardware/software associated with the analytic structures. As examples, consider the following items:

- Hardware configurations
- License expiration date of all hardware/software involved
- Processor, memory and data storage information
- Storage devices
- WinSock information, such as TCP/IP addresses and host names
- Network adapters and network shares
- User accounts and security
- Other information pertaining to the particular hardware/software configurations

6.3 DMC Assessment Findings Report

During the assessment phase (see 6.2, “Assessment” on page 151) we analyzed the following activities listed in Table 6-13.

Table 6-13 A review of assessment

Number	What we assessed
1	Existing analytic structures of enterprise such as: - Independent data marts - Spreadsheets - Data warehouses - Dependent data marts - Other data structures such as Microsoft Access databases and flat files.
2	Data quality and consistency of the analytic structures
3	Data redundancy
4	Source systems involved
5	Business and technical metadata
6	Existing reporting needs
7	Reporting tools and environment
8	Other tools
9	Hardware/software and other inventory

The DMC assessment report based on the above investigation gives detailed information and findings on the data mart consolidation assessment initiative and describes the possible areas where the enterprise could benefit from consolidation.

In short, the DMC assessment findings report gives you the “Analytical Intelligence” capabilities of the analytic structures. The BI of the enterprise is directly proportional to the health of the existing analytic intelligence capability. The current capability depends upon a number of factors, such data quality, data integrity, data mart proliferation, and standardization of common business terms and definitions.

This report helps you to understand the quality of our analytic structures and the level of fragmentation that occurs in them. It shows the health of the enterprise from the data mart proliferation standpoint.

Archived

Important: The DMC assessment findings report gives management a concise understanding of the current state of analytic capability in the enterprise. This report serves as a communicating vehicle to senior management and can be used to quantify the benefits of consolidation, including the single version of the truth. This report can help in securing both long term commitment and funding, which is critical to any project.

Using the DMC assessment findings report, management has a tool that can help determine the level of data mart proliferation. It can also help them answer questions such as, which data marts:

- Have the highest maintenance cost and lowest quality of data?
- Are heavily used, but have data that is highly redundant?
- Have low annual maintenance costs and have high quality data?
- Have maximum number of reports which are critical to business?
- Have standard software/hardware whose licenses are about to expire?
- Have non-standard software/hardware whose licenses are about to expire?
- Use standard reporting tools?
- Use non-standardized reporting tools?

It also helps with questions such as:

- What are the annual training costs associated with the data marts?
- What reporting tools are most/least used?

The DMC assessment findings report helps us analyze the current analytic structures from various aspects such as data redundancy, data quality, ROI, annual maintenance costs, reliability, and overall effectiveness from a business and technical standpoint.

Key elements of the DMC assessment findings report

The DMC assessment findings report can help management understand the current analytic landscape from a much broader perspective. This clear understanding can then help them to decide on their consolidation targets and goals.

The purpose then is to show the context of the problem for senior management. Whether or not such a report can be produced for your enterprise depends on whether the historical information is available. But even if it must be estimated by the IT development staff, it can still be useful.

The DMC report helps us to prioritize a subset of data marts which are candidates for consolidation. Some of the key findings of the DMC assessment report are listed next in detail.

Archived

Data mart proliferation — example:

Figure 6-7 gives a quick view of the existing analytic landscape and the speed of data mart proliferation.

For example, it also shows that the number of independent data marts have risen dramatically in the enterprise. On the other hand, there is a minor growth in the dependent data marts.

We have included dependent and independent data marts in the Figure 6-7, however we recommend that you also include other analytic structures such as data warehouses, denormalized reporting databases, flat files, and spreadsheets.

Figure 6-7 Enterprise data mart proliferation

Archived

Data quality of data marts — example:

Figure 6-8 shows the data quality of all data marts in the enterprise.

Figure 6-8 Data quality of various data marts

Annual maintenance costs and data quality analysis — example:

Figure 6-9 helps us identify data marts with high annual maintenance costs and high data quality. Such data marts would be good candidates for consolidation, and should provide a high ROI.

Archived

Figure 6-9 Annual cost versus data quality

Data redundancy and data quality — example:

This report helps with the analysis of the various existing data marts from a data quality and redundancy standpoint. The redundancy is relative to data that would be present in the EDW. Independent data marts that have more common data with the EDW are more redundant by their nature. You can also see that the quality of data in redundant data marts is typically very low.

As shown in Figure 6-10, the data marts are positioned on the quadrant according to their data quality and data redundancy. Data marts having poor quality data should be identified and the source of the poor quality corrected.

Those with high redundancy (Data Mart#1 and Data Mart#2 as examples), and good quality of data, are good candidates for consolidation. The high redundancy indicates excess ETL processing and maintenance costs, and can result in data consistency issues depending on the synchronization of the update cycles.

Archived

Figure 6-10 Analysis from data quality and redundancy standpoint

Date of expiration for hardware/software/contracts — example:

Figure 6-11 shows the various expiration dates for the hardware/software contracts associated with the data marts.

Archived

Figure 6-11 License expiration for hardware/software/contracts

Note: Data marts whose hardware/software licenses are about to expire are typically good candidates for consolidation

% Sales increase for various business processes — example:

Figure 6-12 shows the % sales increase for business processes using different data marts. It has been observed that the % sales increase (or decrease) is typically directly related to the quality of data in the analytic structures.

However, it is clear that more data would be required to credibly determine the true results. For example, intangibles such as management and market conditions would certainly also be factors to be considered.

Figure 6-12 % Sales increase

Hardware/operating system example:

Figure 6-13 shows the number of dependent and independent data marts in the enterprise using different hardware and operating system combinations. This can help management know what hardware and operating systems are most and least used. This can help with decision making regarding the consolidation of the lesser used hardware and operating systems to more standardized ones in the enterprise, which can result in lower operating costs.

Figure 6-13 Hardware/operating system

Archived

Data and usage growth of data marts — example:

In this simplistic example, note that performance of certain data marts decreases with the increase in data or usage. Figure 6-14 helps to analyze which data marts are experiencing data growth over the years.

Another simple approach would be to plot the data size in GB by year, which would make it easy to see and understand the actual growth at a glance.

Figure 6-14 Data and usage growth of data marts

Figure 6-14 shows that data mart titled “Data Mart #1” has seen a dramatic reduction in annual growth in data. A detailed analysis of the data growth may signify their importance to the business and thereby helps prioritize the candidates for consolidation.

Archived

Total costs — example:

Figure 6-15 shows in % the distribution of total costs of all data marts in the enterprise. This enables you to see which is the most expensive and least expensive data marts. A primary component of data mart cost is the software/hardware and maintenance cost. If you can consolidate independent data marts from their diverse platforms to the EDW, you would be able to achieve significant savings for the enterprise.

However, be careful. In many cases the hardware/software component may be less, but could be much more expensive in terms of the people cost for support and maintenance.

Figure 6-15 % cost distribution of data marts

Total hardware costs — example:

Figure 6-16 shows the total costs for different types of hardware involved to support the various analytic structures in the enterprise.

Figure 6-16 Total hardware costs for data marts

Looking at Figure 6-16, its easy for management to decide how much non-standard hardware is costing the enterprise. Also, Figure 6-11 on page 174, shows the license expiration dates for the various hardware. This combination of information can make it easier to select candidates for consolidation.

Archived

Standardizing reporting tools — example:

Figure 6-17 shows standardized and non-standardized reporting tools available within the enterprise and the number of business users using these. This can help the management help in standardizing and consolidating the reporting tools across the enterprise.

Figure 6-17 Reporting tools and users

Other information:

Several other factors can be included in the DMC report, such as these:

- Which organizations want, or need, to keep their data mart?
- Which non-standard BI tools (such as ETL, data cleansing, and OLAP) need to be standardized or consolidated?
- Which data marts are logically, and which are physically, separate?

6.4 Planning

The primary goal of the planning phase is to define the consolidation project. The focus is on defining scope, budget, resources, and sponsorship. A detailed project plan can be developed during this phase.

The planning phase takes inputs from the DMC assessment findings report that we discussed in 6.3, “DMC Assessment Findings Report” on page 168. Based on those findings, an appropriate consolidation strategy can be determined.

Archived

A key deliverable you can develop during this phase is an implementation recommendation report. The types of information to be contained in such a report are listed in 6.6, “Design” on page 183.

6.4.1 Identify a sponsor

Before starting a data mart consolidation project, it is very important to have a good sponsor — one that not only has a strong commitment, but also a strong presence across different business processes. A strong sponsor, who is well known across the enterprise, can more easily emphasize the vision and importance of data mart consolidation. The sponsor will also need to exhibit a strong presence to gain consensus of business process leaders on such things as common definitions and terms to be used across the enterprise. This standardization of business metadata is crucial for the enterprise to achieve a single version of the truth.

6.4.2 Identify analytical structures to be consolidated

To identify the analytic structures that will be candidates to be consolidated, again look to the DMC assessment findings report. Table 6-14 depicts an example of a few candidates chosen.

Table 6-14 Candidates for consolidation

Number	Name of analytical structure	Owner	Hardware	OS	Database
1	Inventory_DB	North Region	System X	UNIX	Oracle 9i
2	Sales_DB	West Region	System Z	UNIX	Oracle 9i

6.4.3 Select the consolidation approach

As discussed in Chapter 4, “Consolidation: A look at the approaches”, we have defined three approaches for consolidation:

- Simple migration (platform change with same data model)
- Centralized consolidation (platform change with new data model or changes to the existing data model)
- Distributed consolidation (no platform change, but dimensions are conformed across existing data marts to achieve data consistency)

Based on the DMC Assessment Findings report, a particular consolidation approach can be recommended.

Note: Rather than only using one approach, each of them may be used, depending on the size of the enterprise, required speed of deployment, cost savings to be achieved, and, most importantly, the current state of the analytic environment.

It is important to understand the association between metadata standardization and the speed with which a particular consolidation strategy can be deployed. Generally, strategies such as simple migration and distributed consolidation are faster to deploy because they require no metadata standardization. The centralized approach is typically the slowest to deploy because of the required effort to achieve metadata standardization.

It is for these reasons that an enterprise may choose a mix of approaches. For example assume that an organization wants to cleanse and integrate all the data in their independent data marts. One approach could be to start off with simple migration. This would initially result in reducing costs of multiple hardware and software platforms involved. Later a centralized approach could be used to cleanse, standardize, and integrate the data. So, the simple migration approach can provide a starting point, leading up to the more complex and time demanding effort required in the centralized integration approach.

6.4.4 Other consolidation areas

In addition to consolidating the data marts, and the data contained in them, there are other areas that you will need to consider for consolidating. For example, when looking at the data mart, look at every system to which they connect, and every other system to which those systems connect. This will provide an exhaustive list of components to be consolidated.

Consolidating reporting environments

Each independent data mart generally has its own reporting environment. This means that each data mart implementation may also include a report server, security, templates, metadata, backup procedure, print server, development tools, and all the other costs associated with the reporting environment. We have discussed this in detail in 6.2.6, “Reporting tools and environment” on page 163.

As a summary, we recommended that you consolidate your reporting tools and environments into a more standard environment. This can yield savings in license cost, maintenance, support, and ongoing training. There is also additional discussion on the benefits of a consolidated reporting environment in Chapter 4, “Consolidation: A look at the approaches”, specifically in 4.5.1, “Reporting environments” on page 96.

Consolidating other BI tools

As we discussed in the assessment phase in 6.2.7, “Other BI tools” on page 166, there are several tools and requirements that are typically involved with any independent data mart implementation.

Some examples of these are:

- Database management systems
- ETL tools
- Dashboards
- Data modeling tools
- Specific operating systems
- Tools for software version control
- OLAP tools for building cubes based on MOLAP, ROLAP, HOLAP structures
- Project management tools

An enterprise can gain huge benefits by consolidating and standardizing their environment to some degree. And, it sets the direction of standardization for future implementations.

6.4.5 Prepare the DMC project plan

In our sample DMC project plan, we identify the following aspects:

- DMC project purpose and objectives
- Scope definition
- Risks, constraints, and concerns
- Data marts to be consolidated
- Effort required to cleanse and integrate the data
- Effort required to standardize the metadata
- Reporting tools or environments to consolidate
- Other BI tools and processes to standardize
- Deliverables
- Stakeholders
- Communications plan

6.4.6 Identify the team

The team is selected based on the scope and complexity of the consolidation project. If there is need to integrate and cleanse data, integrate metadata, and reduce redundancy, then the centralized consolidation approach will be selected and a team with a high level of expertise will be required. If we plan an initial phase that is only to consolidate using the simple migration approach, then the team skill levels required will not be as great.

Archived

In this section we consider the roles and skill profiles needed for a successful data mart consolidation project. We describe the roles of the business and development project groups only. Not all of the project members will be full-time members. Some of them, typically the business project leader and the business subject area specialist, are part-time members.

The number of people needed depends on the enterprise and scope of the project. There is no one-to-one relationship between the role description and the project members. Some project roles can be filled by one person, whereas, others need to be filled by more than one person.

The consolidation team could be grouped as follows:

- Business Group, which typically includes:
  - Sponsor
  - Business project leader
  - End users (1-2 per functional area)
- Technical Group, which typically includes:
  - Technical project manager
  - Data mart solution architect
  - Metadata specialist
  - Business subject area specialist
  - Platform specialists (full-time and part-time)

    Typically more than one platform specialist is needed. For each existing system (for example, OS/390® hosts, AS/400®, and/or UNIX systems) source, a specialist will be needed to provide access and connectivity. If the consolidation environment will be multi-tiered (for example, with UNIX massive parallel processing (MPP) platforms or symmetrical multiprocessing (SMP) servers, and Windows NT® departmental systems) then more platform specialists will be required.
  - DBA
  - Tools
  - ETL programmer

6.5 Implementation recommendation report

Based on the activities determined in the planning phase, an implementation report can be created. This report identifies the following aspects:

- Name of the sponsor for the project
- Team members
- Scope and risks involved
- Approach to be followed

Archived

- Analytical structures to be consolidated
- Reporting and other BI tools to be consolidated
- EDW where the analytic structures will be consolidated
- Detailed implementation plan

6.6 Design

The design phase may include the following activities:

- Target EDW schema design
- Standardization of business rules and definitions
- Metadata standardization
- Identify dimensions and facts to be conformed
- Source to target data mapping
- ETL design
- User reports

6.6.1 Target EDW schema design

Consolidating data marts may involve minimal or a high level of data schema changes for the existing analytic structures. The target EDW architecture design varies from one approach to another. As an example, we now describe several approaches:

Simple Migration approach

In the Simple Migration approach, there is no change in the existing schema of the independent data marts. We simply move the existing analytic structures to a single platform. This is depicted in Figure 6-18.

In addition to the schema creation process on the target platform, some of the objects that may be needed to be ported are:

- Stored procedures written may need to be converted based on the requirements of the new platform.
- Views
- Triggers
- User defined data types may need to be converted to compatible data types on the consolidated database (DB2 in the example depicted in Figure 6-18).

Archived

Figure 6-18 Simple Migration Approach

Centralized Consolidation approach

In the Centralized Consolidation approach, the data can be consolidated in two ways. A detailed discussion of this can be found in Chapter 4, “Consolidation: A look at the approaches” in 4.3, “Combining data schemas” on page 88. Those two ways of consolidating data in the centralized approach are:

- **Redesign:** In this approach, the EDW schema is redesigned. This is depicted in Figure 6-19, where you can see the new schema.

Figure 6-19 Centralized Consolidation -Using Redesign

- **Merge with Primary technique:** In this approach, we identify one *primary* data mart among all the existing independent data marts. This *primary* data mart is then chosen to be first migrated into the EDW environment. All other independent data marts migrated later will be conformed according the *primary* data mart - which now exists in the EDW.

  The primary data mart schema is then used as the base schema. The other independent data marts are migrated to merge into the primary schema. This is depicted in Figure 6-20.

Archived

Figure 6-20 Centralized consolidation-merge with primary

Distributed Consolidation approach

In the Distributed Consolidation approach, the data across various independent data marts is consolidated without *physically* integrating the data marts. This is done by restructuring the dimensions in each data mart so that they *conform* with each other. There is no schema change, and only certain dimensions are changed to a standardized conformed dimension.

6.6.2 Standardize business definitions and rules

Table 6-15 shows in detail the level of standardization of business definitions and rules which may be involved in the three consolidation approaches.

Table 6-15 Standardization of business definitions and rules

No.	Approach	Standardization?
1	Simple Migration	None. In case of simple migration, there is no level of standardization of Business definitions and rules. Only the data is migrated from one platform to another without any data integration or metadata standardization. The business definitions and rules remain the same.
2	Centralized Consolidation - with Redesign	Yes. With this approach, business definitions and rules are standardized across different business processes. For example, different business processes agree on common definitions for terms such as product and customer. The scope and boundary of the definition is large enough to keep the customer definition standardized but also to satisfy each business process. This is accomplished by conforming the dimensions.

Archived

No.	Approach	Standardization?
3	Centralized Consolidation - Merge with Primary	Yes. The business definitions and rules are standardized across the different business processes.
4	Distributed Consolidation	None. Business definitions and rules are standardized for conformed dimensions. Other business rules and definitions remain the same.]

6.6.3 Metadata standardization

Table 6-16 shows in detail the level of metadata standardization involved in the three consolidation approaches.

Table 6-16 Metadata standardization

No	Approach	Metadata Standardization
1	Simple Migration	None. Only the data is migrated from one platform to another without any data integration or metadata standardization.
2	Centralized Consolidation - with Redesign	Yes. Metadata management and standardization are the key activities of this approach.]
3	Centralized Consolidation - Merge with Primary	Yes. Metadata management and standardization are the key activities of this approach.]
4	Distributed Consolidation	There is no metadata standardization across different data marts. Different metadata exists for different data marts, and only the data marts dimensions are conformed.

Metadata management includes the following types:

- **Business metadata:** This provides a roadmap for users to access the data warehouse. Business metadata hides technological constraints by mapping business language to the technical systems.

  Business metadata includes:

  - Glossary of terms
  - Terms and definitions for tables and columns.
  - Definition for all reports
  - Definition of data in the data warehouse

- **Technical metadata:** This includes the technical aspects of data such as table columns, data types, lengths, and lineage. Examples are:
  - Physical table and column names.
  - Data mapping and transformation logic.
  - Source system details.
  - Foreign keys and indexes.
  - Security.
  - Lineage analysis: Helps track data from a report back to the source, including any transformations involved.
- **ETL execution metadata:** This includes the data produced as a result of ETL processes, such as the number of rows loaded and number rejected, errors during execution, and time taken. Some of the columns that can be used as ETL process metadata are:
  - Create date: Date the row was created in the data warehouse.
  - Update date: Date the row was updated in the data warehouse.
  - Create by: User name.
  - Update by: User name.
  - Active in operational system flag: used to indicate whether the production keys of the dimensional record are still active in the operational source.
  - Confidence level indicator: Helps user identify potential problems in the operational source system data.
  - Current flag indicator: Used to identify the latest version of a row.
  - OLTP system identifier: Used to track origination source of data row in the data warehouse for auditing and maintenance purposes.

Note: The metadata is standardized across the business processes only when we use the Centralized Consolidation approach. With Simple Migration and Distributed Consolidation, metadata is not standardized.

6.6.4 Identify dimensions and facts to be conformed

One way to integrate and standardize the data environment, with a dimensional model, is called conforming. Simply put, that means the data in multiple tables conform around some of the attributes of that data.

Archived

The main concept here is that the keys used to identify the same data in different tables should have the same structure and be drawn from the same domain. For example, if two data marts both have the concept of store, region, and area, then they are conformed if the attributes in the keys that are used to qualify the rows of data have the same definition, and draw their values from the same reference tables.

For example, a conformed dimension means the same thing to each fact table to which it can be joined. A more precise definition would be to say that two dimensions are conformed if they share one, more, or all attributes that are drawn from the same domain. In other words, a dimension may be conformed even if contains only a subset of attributes from the primary dimension.

Fact conformation means that if two facts exist in two separate locations in the EDW, then they must be the same to be called the same. As an example, revenue and profit are each facts that must be conformed. By conforming a fact we mean that all business processes must agree on a common definition for the revenue and profit measures so that revenue and profit from separate fact tables can be combined mathematically. Table 6-17 details which of the three consolidation approaches involve the introduction of conformed dimensions and facts.

Table 6-17 Conformed dimensions and facts

No	Approach	Dimensions and Facts to be Conformed
1	Simple Migration	None. Only the data is migrated from one platform to another, without any data integration or metadata standardization. There are no conformed dimensions and facts in the simple migration approach.
2	Centralized Consolidation - with Redesign	Yes. The dimensions and facts are conformed across different business processes.]
3	Centralized Consolidation - Merge with Primary	Yes. The dimensions and facts are conformed across different business processes.]
4	Distributed Consolidation	Yes. The dimensions and facts are conformed across different business processes. However, there is no platform change.]

Here we describe how to create conformed dimensions and facts in the two approaches mentioned:

- **Consolidating data marts into an EDW:**

  When consolidating data marts into an existing EDW, you need to look for any already conformed standard source of information available before designing a new dimension. Let us assume that we are consolidating two independent data marts called data mart#1 and data mart#2 into an EDW.

  First, list all dimensions of the EDW and independent data marts as depicted in the example in Figure 6-21. Note that the information pertaining to calendar, product, and vendor is present in the data marts (Datamart#1 and Datamart#2) and also the EDW.

  The next step is to study the calendar, product, and vendor dimension tables to identify whether these existing tables have enough information (columns) to answer the queries relating to the Datamart#1 and Datamart#2 business processes. When information is missing from the existing EDW tables, then information would have to be added to the EDW dimension tables.

  Also compare all existing facts present in the EDW with those present in the data marts being consolidated to see if any facts can be conformed.

	Data Mart #1								Data Mart #2			
Existing EDW Tables	Product	Stores	Stores_Category	Customer	Customer_Category	Employee	Calendar	Supplier	Calendar	Product	Stores	Supplier
Product	X									X		
Vendor								X				X
Calendar							X		X			
Warehouse												
Carrier												
Merchant												
Merchant_Group												
Bank Account												
And More ⋮												

Figure 6-21 Identifying dimensions to conform

Archived

- **Consolidating data marts without an EDW (or creating a new EDW):**

  In situations where we are consolidating two independent data marts into a new EDW or new schema, we need to identify common elements of information between these two independent data marts. We identify common dimensions by listing the two independent data marts (DataMart#1 and DataMart#2) as shown in Table 6-22. We also identify any common facts to be conformed between the two independent data marts.

Note: Conformed dimensions and facts are created only in the Centralized and Distributed consolidation approaches.

	Data Mart #2			
Data Mart #1	**Product**	**Store**	**Suppliers**	**Calendar**
Product	X			
Customer				
Customer_Category				
Store		X		
Store_Category		X		
Calendar				X
Supplier			X	
Employee				

Figure 6-22 Conformed dimensions when no EDW exists

The ultimate success of data integration, or consolidation, ends with the delivery of reliable, relevant and complete information to end users. However, it starts with understanding source systems. This is accomplished by data profiling, which is typically the first step in data integration or consolidation. Until recently, data profiling has been a labor-intensive, resource-devouring, and many times error-prone process. Depending on the number and size of the systems, the data discovery process can add unexpected time to project deadlines and significant ongoing costs as new systems are deployed.

Archived

IBM WebSphere ProfileStage, a key component in IBM WebSphere Data Integration Suite, is our data profiling and source system analysis solution. IBM WebSphere ProfileStage completely automates this first step in data integration, and can dramatically reduce the time it takes to profile data. It can also drastically reduce the overall time it takes to complete large scale data integration projects, by automatically creating ETL job definitions - subsequently run by WebSphere DataStage.

6.6.5 Source to target mapping

Before the ETL process can be designed, the detailed ETL transformation specifications for data extraction, transformation, and reconciliation have to be developed.

A common way to document the ETL transformation rules is in a source-to-target mapping document, which can be a matrix or a spreadsheet. An example of such a mapping document template is shown in Table 6-18.

Table 6-18 Source to target mapping template

Table Name	Column Name	Data Type	Data Mart Name	Table Name	Column Name	Data Type	Data Conversion Rules

Alternatively, you could use the combination of the WebSphere ProfileStage and DataStage products to generate and document the source to target mappings.

6.6.6 ETL design

Table 6-19 shows key ETL design differences for the consolidation approaches that have been defined in this redbook:

Archived

Table 6-19 ETL design for consolidation

No.	Approach	Key points in the ETL design
1	Simple Migration	The ETL involves the following procedures: - First, an ETL process is created which involves migrating data from source data marts to the target schema. This is typically a one-time load process to transfer historic data from the data marts to the EDW. - Second, an ETL process is created to tie the sources of data marts to the target schema. After the second ETL process is completed successfully, then the data marts can be eliminated. Other key features of the ETL process of simple migration approach are: - All objects, such as stored procedures, of the data marts being consolidated are migrated to the new target platform. - There is no change in the schema of the consolidated data marts. - There is no integration of data. - There is no standardization of metadata. - There is no improvement in data quality and consistency. Generally the quality of data available in the consolidated platform is the same as with the individual data mart. - The ETL follows a conventional database migration approach. For more details on data conversion, please refer to Chapter 7, “Consolidating the data”.
2	Centralized Consolidation - with Redesign	The ETL involves the following procedures: - First, an ETL process is created which involves migrating data from source data marts to the target schema. - Second, an ETL process is created to tie the sources of data marts to the target schema. After the second ETL process is completed successfully, then the data marts can be eliminated. Other key features of the ETL process for the centralized consolidation approach (using redesign) are: - Surrogate keys are used to handle dimension versioning and history. - History is maintained in the EDW using the type1, type2, or type3 approach. - Standardized metadata is made available. - Data is integrated and made consistent. - Data quality is improved.

Archived

No.	Approach	Key points in the ETL design
3	Centralized Consolidation - Merge with Primary	The ETL involves the following procedures: - First, an ETL process is created which involves migrating data from source data marts to the target schema. - Second, an ETL process is created to tie the sources of data marts to the target schema. The other features of ETL are same as “Centralized Consolidation Approach -With Redesign”.
4	Distributed Consolidation	In this approach, only the dimensions are conformed to standardized dimensions. There is no schema or platform change, except those required to conform the dimensions. Generally, a staging area is created that contains the conformed dimensions. This area feeds the various distributed data marts.

The ETL process for consolidating data marts, using simple migration and centralized consolidation approach, into the target schema or EDW is broadly divided into two steps as shown Figure 6-23.

Figure 6-23 ETL process in consolidation

- In Step 1, the ETL process is designed to transfer data from the two data marts (Data Mart #1 and Data Mart #2) into the EDW.
- In Step 2, the ETL process is designed to feed the EDW directly from the original sources for the Data Mart #1 and Data Mart #2. As shown in Figure 6-23, Data Mart #1 and Data Mart #2 can be eliminated after the data from these data marts has been successfully populated into the EDW.

6.6.7 User reports requirements

Reports based on the existing data marts may be impacted depending upon the consolidation approach chosen. This is described in Table 6-20.

Table 6-20 Reports

No	Approach	How are existing reports affected?
1	Simple Migration	No change in existing reports. Only the client reporting applications are made to point to the new consolidated platform.
2	Centralized Consolidation - with Redesign	Existing reports change. This is due to the fact that the target schema is redesigned.
3	Centralized Consolidation - Merge with Primary	Existing reports change for the data mart that is merged into the primary mart. The reports for the secondary data mart change as there is a change in the target schema.
4	Distributed Consolidation	No change, or very minor change. There is a minor change in the dimensional structure of existing data marts as they are conformed to standardized dimensions. Otherwise, the rest of the schema structure remains the same. Due to this reason there is usually little or no change in existing reports.

We have given you a few guidelines, but you need to validate all report requirements with the business processes.

Archived

6.7 Implementation

The implementation phase includes the following activities:

- Construct target schema:

  This involves creating the target EDW tables or schema using the design created in 6.6.1, “Target EDW schema design” on page 183.

- Construct ETL process:

  This involves constructing the ETL process based on:

  – Source to target data mapping matrix (as described in 6.6.5, “Source to target mapping” on page 191).

  – ETL construction (as described in 6.6.6, “ETL design” on page 191) involves the following ETL processes for moving data:

    - From data marts to the target schema
    - From source (OLTP) systems to target schema

- Modifying/constructing end user reports
- Adjusting and creating operational routines for activities such as backup and recovery

Based on the consolidation approach and reporting requirements, we may need to modify existing reports or redesign reports. This is done based on the input we got in 6.6.7, “User reports requirements” on page 194.

- Standardizing reporting environment:

  This process involves using the standardized reporting tools that were finalized by the data mart consolidation team during the planning phase. We discussed this in “Consolidating reporting environments” on page 180.

  In order to start using a standardized reporting tool, we need to perform activities such as:

  – Product installation
  – Reporting tool repository construction
  – Reporting server configuration
  – Reporting tool client configuration
  – Reporting tool broadcasting server configuration
  – Reporting tool print server configuration
  – Reporting tool Web server configuration

- Standardizing BI tools:

  This process involves using other standardized tools during the consolidation process. Using standardized tools is a first step towards a broader consolidation approach. We discussed this in detail in “Consolidating other BI tools” on page 181.

Archived

6.8 Testing

In this phase, we are either comparing old and new systems to ensure that they give the same results, or we are performing acceptance tests against the new implementation.

In this step we test the entire application and consolidated database for correct results and good performance. Ideally, this would be a set of automated tests and the original source data marts would be available to run the same tests against. Typically, this involves the validation of user interface transactions, data quality, data consistency, data integrity, ETL code, Batch/Script processing, administration procedures, recovery plans, reports and performance tuning. It is often necessary to adjust for configuration differences (such as the number of CPUs and memory size) when comparing the results.

Here is a suggested checklist for acceptance testing:

- Define a set of test reports.
- Write automated test and comparison routines for validation.
- Develop acceptance criteria against which to measure test results.
- Correct errors and performance issues.
- Define known differences between the original and the new system.

6.9 Deployment

When the consolidation testing is complete and the results accepted, it is time to deploy to the users. You will, of course, need to document the new environment and put together an education and training plan. These steps are critical to gaining acceptance of the new environment and making the project a success.

Once the report development and database functionality has been validated and tuned, application documentation will need to be updated. This should include configuration and tuning tips discovered by the development and performance tuning personnel for use by the maintenance and support teams.

Checklist for documentation:

- Finalize all system documentation needed for maintenance and support. Some examples are:
  - Report specifications
  - Operational procedures
  - Schedules

- Finalize documentation needed by the users, such as:
  - Glossary of terms
  - Data model
  - Directory of reports
  - Metadata repository
  - Support procedures

6.10 Continuing the consolidation process

The consolidation lifecycle was designed to accommodate a variety of implementation and operational scenarios. That is why we included a process to make the lifecycle iterative.

It might be that the lifecycle addresses multiple consolidation projects, but it could as well address multiple iterations relative to a single consolidation project. This implies that there is no requirement to complete a consolidation project in one major effort. In those instances where a major consolidation effort would have an impact on the ongoing enterprise operations, the project perhaps should be staged. This approach enables you to start small, and complete the project in planned, painless, and non-disruptive steps.

In addition, you will remember that data warehousing itself is a process. That is, typically you are never really finished. You will undoubtedly continue to add people, processes, products, to your enterprise. Or, specifications about those people, processes, and products will change. As examples, people get married, have children, and move. This requires modifying, for example, dimensional data that describes those people.

So, as with any data warehousing project, this is a continuous and ongoing project. Having a defined lifecycle, and using the tools and suggestions in this redbook, will enable a structured, planned, and cost effective approach to maintaining your data warehousing environment.

Archived

Archived

7

Consolidating the data

In a data consolidation project, particularly one involving heterogeneous data sources, there will be a requirement for some data conversion. This is because the data management systems from the various vendors typically employ a number of differing data types. Therefore, the data types on any of the source systems will have to be converted to comply with the definitions on the target system being used for the consolidation.

In this chapter we discuss the methods available for converting data from various heterogeneous data sources to the target DB2 data warehouse environment. The sources may be either relational or non-relational, and on any variety of operating environments. As examples, conversion can be accomplished by using native utilities such as Import/Export and Load/Unload. However, in these situations there would typically be a good deal of manual intervention required. In addition, specialized tools have been developed by the various data management vendors in order make the process of data conversion easier.

© Copyright IBM Corp. 2005. All rights reserved.

7.1 Converting the data

Data conversion is the process of moving data from one environment to another without losing the integrity or meaning of the data. The data can be transformed into other meaningful types and structures as per the requirements of the target database. Multiple tools and technologies are available for data conversion purposes.

There are many reasons for an enterprise to convert their data from existing systems. In a data warehousing environment there could be disparate sources of data in different formats and databases which have to be converted in order to consolidate into a central database that will provide integrity, consistency, currency, and low cost of maintenance.

During the process of data conversion we might come across many issues, especially if the data volume is large or if the sources contain different formats of data. The process also becomes more difficult if the data in the source is highly volatile. Online data conversion should be done using tools such as WebSphere Information Integrator edition with replication capability. There are a number of methods for data conversion; for example:

- The data from source systems can be converted directly to target systems without applying any transformation logic using tools, such as the IBM DB2 Migration ToolKit.
- During conversion of the data from the source systems, appropriate transformation logic can be applied and moved into the target systems without losing the integrity of the data. This can be achieved by using tools such as WebSphere Information Integrator, DB2 Warehouse Manager, and in-house developed programs.

7.1.1 Data conversion process

The data conversion process is quite complex. Before you define a data conversion method, you should do some tests only with a portion of the data to verify the chosen method. But the tests should cover all potential cases. We recommend that you start early with the testing.

The tasks of the test phase are:

- Calculate the source data size and the required disk space.
- Select the tools and the conversion method.
- Test the conversion using the chosen method with a small amount of data.

With the results of the test, you should be able to:

- Estimate the time for the complete data conversion process.

- Create a plan for development environment conversion. Use the information about this to derive a complete plan.
- Create a plan for production environment conversion. Use the information from development environment conversion.
- Schedule the time.

The following aspects influence the time and complexity of the process:

- **Volume of data and data changes:**

  The more data you have to move, the more time you need. Consider the data changes as well as the timestamp conversions.

- **Data variety:**

  Although volume of data is also a consideration, it is the variety of data that impacts the complexity, because it impacts the development time requirement. Development time to convert one record is the same as for a million records; whereas the development time for one data type is different, or additive, to the development time for another data type.

- **System availability:**

  You can run the data movement either while the production system is down or while the business process is running, by synchronizing the source and target database. The time required depends on the strategy you choose.

- **Hardware resources:**

  Be aware that you need up to three times the disk space during the data movement for:

  – The data in the source database, such as Oracle and SQL/Server
  – The unloaded data stored in the file system
  – The loaded data in the target DB2 UDB

7.1.2 Time planning

After testing the data movement and choosing the proper tool and strategy, you have to create a detailed time plan with tasks such as the following:

- Depending on the data movement method:
  – Implementing or modifying scripts for data unload and load
  – Learning the use of the chosen data movement tools
- Data unload from source database such as Oracle and SQL/Server
- Data load to DB2 UDB
- Backup target database
- Test loaded data for completeness and consistency
- Conversion of applications and application interfaces
- Fallback process in case of problems

Archived

The most sensitive environment is a production system with a 7x24 hour availability requirement. Figure 7-1 depicts one way to move the data to the target database in a high availability environment. The dark color represents the new data, the light color represents the converted and moved data. If possible, export the data from a standby database or mirror database to minimize the impact on the production environment. Here are the tasks:

1. Create scripts that export all data up to a defined timestamp.
2. Create scripts that export changed data since the last export. This includes new data as well as deleted data.
3. Repeat step 2 as often as when all data is moved to the target database.
4. Define fallback strategy and prepare fallback scripts.

Figure 7-1 Data movement strategy in a high availability environment

When the data is completely moved to the target database, you can switch the application and database. Prepare a well defined rollout process for the applications, and the interfaces belonging to DB2 UDB. Allow time for unplanned incidents.

7.1.3 DB2 Migration ToolKit

The IBM DB2 Migration ToolKit (MTK) can help you migrate from Oracle (Versions 7, 8i, and 9i), Sybase ASE (Versions 11 through 12.5), Microsoft SQL Server (Versions 6, 7, and 2000), Informix (IDS v7.3 and v9), and Informix XPS (limited support) to DB2 UDB V8.1 and DB2 V8.2 on Windows, UNIX, Linux, and DB2 iSeries including iSeries v5r3. The MTK is available in English, and on a variety of platforms including Windows (2000, NT 4.0 and XP), AIX, Linux, HP/UX, and Solaris.

This MTK provides a wizard and an easy-to-use, five-step interface that can quickly convert existing Sybase, Microsoft SQL Server and Oracle database objects to DB2 Universal Database. You can automatically convert data types, tables, columns, views, indexes, stored procedures user-defined functions and triggers into equivalent DB2 database objects. The MTK provides database administrators and application programmers with the tools needed to automate previously inefficient and costly migration tasks. It can also reduce downtime, eliminate human error, and minimize person hours and other resources associated with traditional database migration.

The MTK enables the migration of complex databases, and has a full functioning GUI interface that provides more options to further refine the migration. For example, you can change the default choices that are made about which DB2 data type to map to the corresponding source database data types. The MTK also converts and refines DB2 database scripts. This model also makes the MTK very portable, making it possible to import and convert on a machine remote from where the source database and DB2 are installed.

These are some of the key features of the DB2 Universal Database Migration ToolKit, which:

- Extracts database metadata from source DDL statements using direct source database access (JDBC/ODBC) or imported SQL scripts.
- Automates the conversion of database object definitions — including stored procedures, user-defined functions, triggers, packages, tables, views, indexes and sequences.
- Accesses helpful SQL and Java compatibility functions that make conversion functionally accurate and consistent.
- Uses the SQL translator tool to perform query conversion in real-time; or uses the tool as a DB2 SQL learning aid for T-SQL/PL-SQL developers.
- Views and refines conversion errors.
- Implements converted objects efficiently using the deployment option.
- Generates and runs data movement scripts.
- Tracks the status of object conversions and data movement — including error messages, error location, and DDL change reports — using the detailed migration log file and report.

The MTK converts the following source database constructs into equivalent DB2:

- Data Types
- Tables
- Columns
- Views

- Indexes
- Constraints
- Packages
- Stored procedures
- Functions
- Triggers

The MTK is available free of charge from IBM at the following URL:

http://www-306.ibm.com/software/data/db2/migration/mtk/

7.1.4 Alternatives for data movement

Besides the MTK, there are other tools and products for data movement. Here we show you some of them. You should chose the tool according to your environment and the amount of data to be moved.

IBM WebSphere DataStage

The DataStage product family is an extraction, transformation, and loading (ETL) solution with end-to-end metadata management and data quality assurance functions. It supports the collection, integration, and transformation of large volumes of data, with data structures ranging from simple to highly complex.

IBM WebSphere DataStage manages data arriving in real-time as well as data received on a periodic or scheduled basis. It is scalable, enabling 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 WebSphere DataStage Enterprise Edition can scale to satisfy the demands of ever-growing data volumes, stringent real-time requirements, and ever shrinking batch windows.

DataStage supports a virtually unlimited number of heterogeneous data sources and targets in a single job, including: 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 like SAS.

The real-time data integration support 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 WebSphere DataStage SOA Edition 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.

WebSphere Information Integrator

In a high availability environment, you have to move the data during production activity. A practical solution is the replication facility of the WebSphere II.

IBM WebSphere Information Integrator provides integrated, real-time access to diverse data as if it were a single database, regardless of where it resides. You are able to hold the same data both in supported source databases (Oracle, SQL/Server/Sybase/Teradata) and in DB2 UDB. You are free to switch to the new DB2 database when the functionality of the ported database and application is guaranteed.

The replication server, former known as *DB2 Data Propagator*, allows users to manage data movement strategies between mixed relational data sources including distribution and consolidation models.

Data movement can be managed table-at-a-time such as for warehouse loading during batch windows, or with transaction consistency for data that is never off-line. It can be automated to occur on a specific schedule, at designated intervals, continuously, or as triggered by events. Transformation can be applied in-line with the data movement through standard SQL expressions and stored procedure execution.

For porting data, you can use the replication server to support data consolidation, moving data from source databases like Oracle, SQL/Server to DB2 UDB.

You can get more information about replication in the IBM Redbook, *A Practical Guide to DB2 UDB Data Replication V8,* SG24-6828-00.

DB2 Warehouse Manager

IBM DB2 Warehouse Manager is a basic BI tool which includes enhanced extract, transform, and load (ETL) function over and above the base capabilities of DB2 Data Warehouse Center. DB2 Warehouse Manager also provides metadata management and repository function through the information catalog. The information catalog also provides an integration point for third-party independent software vendors (ISVs) to perform bi-directional metadata and job scheduling exchange. DB2 Warehouse Manager includes one of the most powerful distributed ETL job-scheduling systems in the industry. DB2 Warehouse Manager agents allow direct data movement between source and target systems without the overhead of a centralized server.

DB2 Warehouse Manager includes agents for AIX, Windows NT, Windows 2000, IBM iSeries, Solaris Operating Environment, and IBM z/OS servers to efficiently move data between multiple source databases like Oracle, SQL/Server or any ODBC source and target systems without the bottleneck of a centralized server.

Data movement through named pipes

As described in 7.1.1, “Data conversion process” on page 200, you will need additional disk space during the data movement process. To avoid the space for the flat files, you can use named pipes on UNIX-based systems. To use this function, the writer and reader of the named pipe must be on the same machine. You must create the named pipe on a local file system before exporting data from the Oracle database.

Because the named pipe is treated as a local device, there is no need to specify that the target is a named pipe. The following steps show an AIX example:

1. Create a named pipe:

```
mkfifo /u/dbuser/mypipe
```

2. Use this pipe as the target for data unload operation:

```
<data unload routine> > /u/dbuser/mypipe
```

3. Load data into DB2 UDB from the pipe:

```
<data load routine> < /u/dbuser/mypipe
```

The commands in step 2 and 3 show the basic principles of using the pipes.

> **Note:** It is important to start the pipe reader after starting the pipe writer. Otherwise, the reader will find an empty pipe and exit immediately.

Third party tools

There are a number of migration tools available to assist you in moving your database, application, and data from its existing DBMS to DB2 UDB. These tools and services are not provided by IBM, nor does IBM make guarantees as the performance of these tools:

- **ArtinSoft:**

  Oracle Forms to J2EE. The Oracle Forms to J2EE migrating service produces a Java application with an n-tier architecture, thus allowing you to leverage the knowledge capital invested in your original application, preserving functionality and the “look and feel” evolving into a cost-effective, rapid, and secure fashion to a modern platform.

- **Kumaran:**

  Kumaran offers DB2 UDB migration services for IBM Informix (including 4GL), Accell/Unify, MS Access, Oracle (including Forms and Reports), Ingres, and Microsoft SQL Server.

Archived

- **Techne Knowledge Systems, Inc:**

  The Techne Knowledge Systems JavaConvert/PB product is a software conversion solution that transforms Powerbuilder applications into Java-based ones.

- **Ispirer Systems:**

  Ispirer Systems offers SQLWays, a database and data migration tool.

- **DataJunction:**

  DataJunction data migration tool provides assistance in moving data from Source Database to DB2 UDB. This tool accounts for data type differences, and can set various filters to dynamically modify target columns during the conversion process.

7.1.5 DDL conversion using data modeling tools

A number of modeling tools can help you capture the entity-relationship (ER) descriptions of your database. By capturing this information, you can then direct the tool to transform the information into DDL (Data Definition Language) that is compatible with DB2 UDB. A few of these modeling tools are:

- **Rational Rose Professional Data Modeler Edition:**

  Rational® Rose® offers a database design tool that allows database designers, business analysts, and developers to work together through a common language.

- **CA AllFusion ERwin Data Modeler:**

  A data modeling solution that helps create and maintain databases, data warehouses, and enterprise data models.

- **Embarcadero Technologies ER/Studio:**

  ER/Studio can reverse-engineer the complete schema for many database platforms by extracting object definitions and constructing a graphical data model. Other tools are available for application development (Rapid SQL and DBArtisan).

- **Borland Together:**

  Borland's enterprise development platform provides a suite of tools that enables development teams to build systems quickly and efficiently. Borland Together Control center is an application development environment that encompasses application design, development, and deployment. Borland Together Edition for WebSphere Studio offers IBM-centric development teams a complete models-to-code solution. Borland Together Solo provides an enterprise class software development environment for small development teams.

Archived

7.2 Load/unload

Conversion of data can also be performed using the native utilities available in DB2, Oracle, and SQL/Server. For example, data from the source systems can be unloaded into flat files and loaded into the desired target systems. Unload of data from source systems can be performed by using the native export/unload utilities in order to transfer the source data into flat files or other structured format. The data from the flat files can then be transferred into the target system by using native load/import utilities. This method of data conversion requires that the target data types are mapped correctly to that of the source system so that the load operation would be successful. These load utilities are capable of writing out error messages into log files during the load operation and can be useful for troubleshooting when errors occur.

DB2 UDB provides the LOAD and IMPORT commands for loading data from files into the database. You have to be aware of differences in how specific data types are represented by different database systems. For example, the representation of date and time values may be different in different databases, and it is often depends on the local settings of the system. If the source and target database use different formats, you need to convert the data either automatically by tools or manually, Otherwise, the loading tool cannot understand the data to load due to the wrong format. The migration of binary data stored in BLOBs should be done manually because binary data cannot be exported to files in a text format.

7.3 Converting Oracle data

The Oracle DBMS includes specific tools for converting other source system data to Oracle. For example, the Oracle Warehouse Builder is a suite of tools consisting of both data conversion and transformation capabilities.

Figure 7-2 depicts the tools available in Oracle and DB2 for data conversion purposes. There are three scenarios explained there:

- Conversion of data from flat file to Oracle
- Conversion of data from flat file to DB2
- Conversion of data from Oracle to DB2

Conversion of data from flat file to Oracle

This type of data conversion can be performed using the native load utility of Oracle, or tools such as Oracle Warehouse Builder. Before using the load utility, the data types of the Oracle table to be populated have to be modified so that the data types of the columns in the table map with the data types of the data in the flat file. When using Oracle Warehouse Builder, the data type mappings are relatively simple because the tool provide the data type mapping support.

Figure 7-2 Oracle data conversion

Conversion of data from flat files to DB2

There are a number of IBM tools that can be used for converting data from flat files into DB2. The native load and import utilities of DB2 represent the most frequently used mode of data conversion from flat files to DB2 tables. Apart from the database utilities, the DB2 warehouse manager has built-in functions to map the column data types of the flat file to the column data types of the DB2 table, and then transfer data into the DB2 database. IBM WebSphere II can be used in conjunction with the DB2 Warehouse Manager or WebSphere DataStage when transformations are required during the conversion process. The advantage of using these tools together would be that WebSphere II can be used to federate the flat file for query access and DB2 Warehouse Manager or WebSphere DataStage can then be used to create transformation steps by joining the table data and the flat file data in order to produce the desired output.

Conversion of data from Oracle to DB2

Data conversion from Oracle to DB2 can be performed by using many of the IBM tools and database utilities. The DB2 Migration ToolKit provides robust capabilities for helping in such a conversion activity. The MTK internally maps the data types from Oracle to DB2, and performs the data transfer. Data from Oracle tables can also be exported to flat files and then loaded into DB2 using native database load utilities. If more complex transformations are involved when moving the data from source to target then tools such as DB2 Warehouse Manager or WebSphere DataStage, and WebSphere II can be used.

One of the first steps in a conversion is to map the data types from the source to the target database. Table 7-1 summarizes the mapping from the Oracle data types to corresponding DB2 data types. The mapping is one to many and depends on the actual usage of the data.

Table 7-1 Mapping Oracle data types to DB2 UDB data types

Oracle data type	DB2 data type	Notes
CHAR(n)	CHAR(n)	1 <= n <= 254
VARCHAR2(n)	VARCHAR(n)	n <= 32762
LONG	LONG VARCHAR(n)	if n <= 32700 bytes
LONG	CLOB(2GB)	if n <= 2 GB
NUMBER(p)	SMALLINT / INTEGER / BIGINT	– SMALLINT, if 1 <= p <= 4 – INTEGER, if 5 <= p <= 9 – BIGINT, if 10 <= p <= 18
NUMBER(p,s)	DECIMAL(p,s)	if s > 0
NUMBER	FLOAT / REAL / DOUBLE	
RAW(n)	CHAR(n) FOR BIT DATA / VARCHAR(n) FOR BIT DATA BLOB(n)	– CHAR, if n <= 254 – VARCHAR, if 254 < n <= 32672 – BLOB, if 32672 < n <= 2 GB
LONG RAW	LONG VARCHAR(n) FOR BIT DATA / BLOB(n)	– LONG, if n <= 32700 – BLOB, if 32700 < n <= 2GB
BLOB	BLOB(n)	if n <= 2 GB
CLOB	CLOB(n)	if n <= 2 GB
NCLOB	DBCLOB(n)	if n <= 2 GB, use DBCLOB(n/2)
DATE	TIMESTAMP	– Use Oracle TO_CHAR() function to extract for subsequent DB2 load. – Oracle default format is DD-MON-YY
DATE (only the date)	DATE (MM/DD/YYYY)	– Use Oracle TO_CHAR() function to extract for subsequent DB2 load.
DATE (only the time)	TIME (HH24:MI:SS)	– Use Oracle TO_CHAR() function to extract for subsequent DB2 load.

For more information on converting Oracle to DB2, see the IBM Redbook, “Oracle to DB2 UDB Conversion Guide”, SG24-7048.

Archived

7.4 Converting SQL Server

The SQL Server database developed by Microsoft Corporation has specific tools for converting data for SQL Server. The Data Transformation Services (DTS) utility is included in the SQL Server database has the functions to both convert and transform data to SQL Server and to other heterogeneous databases.

Figure 7-3 depicts the tools available in SQL Server and DB2 for data conversion purposes. We discuss the following three scenarios:

- Conversion of data from flat file to SQL Server
- Conversion of data from flat file to SQL Server
- Conversion of data from Oracle to SQL Server

Conversion of data from flat file to SQL Server

The Microsoft Data Transformation Services (DTS) feature of SQL Server makes provides capability to extract, transform, and load data from flat files into SQL Server. The Microsoft Bulk Copy Program (BCP) and the Bulk Insert Utilities are other tools for loading data from flat files into SQL Server. The DTS feature of SQL Server also helps in data type mapping between the source and SQL Server data types. The Bulk Insert and Bulk Copy Programs require that the data types are correctly mapped between the source flat file and the target table in SQL Server.

Figure 7-3 SQL Server data conversion

Conversion of data from flat files to DB2

There are a number of IBM tools which can be used for converting data from flat files into DB2. The native load and import utilities of the DB2 database are the most frequently used tools. Apart from the database utilities, the DB2 Warehouse

Manager has built-in functions to map the data types of the flat file to the column data types of the DB2. IBM WebSphere II can be used in conjunction with DB2 Warehouse Manager if complex transformations are required during the conversion process. The advantage to use the tools together would be that WebSphere II can be used to federate the flat file for query access from the DB2 database and DB2 Warehouse Manager can then be used to create transformation steps by joining the table data and the flat file data in order to produce the desired output.

Conversion of data from SQL Server to DB2

Data conversion from SQL Server to DB2 can also be performed by using many of the IBM tools and database utilities. The DB2 Migration ToolKit provides robust capabilities for helping in such a conversion activity. The MTK internally maps the data types from SQL Server to DB2 and performs the data transfer. Data from SQL Server tables can also be exported to flat files and then loaded into DB2 using native database load utilities. If complex transformations are involved when moving the data from source to target, then tools such as DB2 Warehouse Manager and WebSphere II can be used.

One of the first steps in a conversion is to map the data types from the source to the target database. Table 7-2 summarizes the mapping from the SQL Server data types to corresponding DB2 data types. The mapping is one to many and depends on the actual usage of the data.

Table 7-2 SQL/Server to DB2 data type mapping

SQL Server data type	DB2 data type	Notes
CHAR(m)	CHAR(n)	1 <= m<= 8000 1 <= n <= 254
VARCHAR(m)	VARCHAR(n)	1 <= m<= 8000 1 <= n <= 32762
	LONG VARCHAR(n)	if n <= 32700 bytes
TEXT	CLOB(2GB)	if n <= 2 GB
TINYINT	SMALLINT	- 32768 to 32767
SMALLINT	SMALLINT	- 32768 to 32767
INT INTEGER	INT INTEGER	- 2 to (231 - 1)
BIGINT	BIGINT	
DEC(p,s) DECIMAL(p,s)	DEC(p,s) DECIMAL(p,s)	- (1031+1) to (1031 - 1) (p+s <= 31)

SQL Server data type	DB2 data type	Notes
NUMERIC(p,s)	NUM(p,s) NUMERIC(p,s)	- (1031+1) to (1031 - 1) (p+s <= 31)
FLOAT(p)	FLOAT(p)	
REAL	REAL	
DOUBLE	DOUBLE PRECISION	
BIT	CHAR(1) FOR BIT DATA	0 or 1
BINARY(m)	CHAR(n) FOR BIT DATA	1 <= m<= 8000 1 <= n <= 254
VARBINARY(m)	VARCHAR(n) FOR BIT DATA	1 <= m<= 8000 1 <= n <= 32672
IMAGE	BLOB(n)	if n <= 2 GB
TEXT	CLOB(n)	if n <= 2 GB
NTEXT	DBCLOB(n)	0 <= n <= 2 GB
SMALLDATETIME	TIMESTAMP	Jan 1, 0001 to Dec 31, 9999
DATETIME	TIMESTAMP	Jan 1, 0001 to Dec 31, 9999
TIMESTAMP	CHAR(8) FOR BIT DATA	
	DATE (MM/DD/YYYY)	year: 0001 to 9999 month: 1 to 12 day: 1 to 31
	TIME (HH24:MI:SS)	hour: 0 to 24 minutes: 0 to 60 seconds: 0 to 60
NCHAR(m)	GRAPHIC(n)	1 <= m<= 4000 1 <= n <= 127
NVARCHAR(m)	VARGRAPHIC(n)	1 <= m<= 4000 1 <= n <= 16336
	LONG VARGRAPHIC(n)	1 <= n <= 16336

Archived

SQL Server data type	DB2 data type	Notes
SMALLMONEY	NUMERIC(10,4)	
MONEY	NUMERIC(19,4)	
UNIQUEIDENTIFIER	CHAR(13) FOR BIT DATA	

For more information on converting SQL Server to DB2 UDB, see the IBM Redbook, “Microsoft SQL Server to DB2 UDB Conversion Guide”, SG24-6672.

7.5 Application conversion

Application conversion is a key step in a consolidation project. The application conversion process includes:

- Checking software and hardware availability and compatibility
- Education for the developers and administrators
- Analysis of application logic and source code
- Set-up the target environment
- Conversion of application code, including changing database specific items.
- Application testing
- Application tuning
- Roll-out

Check software and hardware availability and compatibility

The architecture profile is one of the outputs in the migration planning assessment. While preparing the architecture profile, you need to check the availability and compatibility of all involved software and hardware in the new environment. This includes compatibility between software levels for the various components.

Education for the developers and administrators

Ensure that the staff has the skills for all products and for the system environment you will use for the migration project. Understanding the new product is essential when developing the new system.

Analysis of application logic and source code

In this analysis phase you should identify all the Oracle proprietary features, and the impacted data sources. Examples of Oracle proprietary features are direct SQL queries to the Oracle Data Dictionary, Optimizer hints, and Oracle joins, which need to be expressed differently in DB2 UDB. You also need to analyze the database calls within the application for the usage of database API.

Setting up the target environment

The target system has to be set-up and ready for application development. The environment will include things as:

- The Integrated Development Environment (IDE)
- Database framework
- Repository
- Source code generator
- Configuration management tool
- Documentation tool
- Test tools

A complex systems environment typically consists of products from multiple vendors. Check the availability and compatibility at the start of the project.

Change of database specific items

Regarding the use of the database API, you need to change the database calls in the applications. The changes include:

- SQL query changes.
- Oracle supports partly non-standard SQL queries, such as the inclusion of optimizer hints or table joins with a (+) syntax. To convert such queries to standard SQL, consider using the MTL SQL Translator.
- You need to modify the SQL queries to the Oracle Data Dictionary as well, and change them to select data from the DB2 UDB Catalog.
- Changes in calling procedures and functions. Sometimes there is a need to change procedures to functions and vice versa. In such cases, you have to change all the calling commands and the logic belonging to the calls in other parts of the database and of the applications.
- Logical changes. Because of architectural differences between heterogeneous databases, changes in the program flow might be necessary. Most of the changes are related to the different concurrency models.

Application testing

A complete application test is necessary after the database conversion. Application modification and testing will be required to ensure that the database conversion is complete, and all the application functions work properly.

It is prudent to run the migration tests several times in a development system to verify the process. Then you can run the same migration in test system with existing test data. Upon successful completion, then take a subset copy of the production data before releasing the process into production.

Archived

Application tuning

Tuning is a continuous activity for the database environment because the data volume, number of users, and applications change from time to time. After the migration, the application tuning should be concerned with the architectural differences between source database and DB2 UDB.

It is a best practice to allow for separate conversion and tuning tasks in the project plan. In the first task the component is converted. The aim of this is to arrive at a representation of the data or code that is functionally equivalent to the original. We then take the converted object and tune it.

This is generally best done as a two step process. The key here is that extra effort for tuning must be included.

Roll-out

The roll-out procedure varies, and depends on the type of application and the kind of database connection you have. Prepare the workstations with the proper driver (as examples, DB2 UDB Runtime Client, ODBC, and JDBC) and the server according to the DB2 UDB version.

User education

When there are changes in the user interface, the business logic, or the application behavior because of system improvements, user education is required. Providing this education will be critical in assuring user satisfaction with the new systems environment.

7.5.1 Converting other Java applications to DB2 UDB

Coding applications and database stored procedures in Java provides flexibility and benefits over using the native language. Most applications or database stored procedures are being created using Java because of following advantages:

- The applications and database stored procedures that you create are highly portable between platforms.
- You can set up a common development environment wherein you use a common language to create the stored procedures on the database server and the client application that runs on a client workstation or a middleware server (such as a Web server).
- There is great potential for code reuse if you have already created many Java methods in your environment that you now want to run as Java stored procedures.

For Java programmers, DB2 UDB offers two application programming interfaces (APIs), JDBC and SQLj.

JDBC is a mandatory component of the Java programming language as defined in the Java 2 Standard Edition (J2SE) specification. To enable JDBC applications for DB2 UDB, an implementation of the various Java classes and interfaces, as defined in the standard, is required. This implementation is known as a JDBC driver. DB2 UDB offers a complete set of JDBC drivers for this purpose. The JDBC drivers are categorized as the legacy CLI drivers or the new *Universal JDBC Drivers*.

SQLJ is a standard development model for data access from Java applications. The SQLJ API is defined within the SQL 1999 specification. The new Universal JDBC Driver provides support for both JDBC and SQLJ APIs in a single implementation. JDBC and SQLJ can inter-operate in the same application. SQLJ provides the unique ability to develop using static SQL statements and control access at the DB2 UDB package level.

The Java code conversion is rather easy. The API itself is well defined and database independent. For instance, the database connection logic is encapsulated in standard J2EE DataSource objects. The Oracle or DB2 UDB specific things such as user name and database name are then configured declaratively within the application.

However, there is the need to change your Java source code regarding:

- The API driver (JDBC or SQLJ)
- The database connect string
- Oracle proprietary SQL, such as CONNECT BY for recursive SQL, the usage of DECODE() or SQL syntax like the (+) operator instead of LEFT/RIGHT OUTER JOIN. MTK provide support here with the *SQL Translator*.
- Remove or simulate proprietary optimizer hints in SQL queries.

Java access methods to DB2

DB2 UDB has rich support for the Java programming environment. You can access DB2 data by putting the Java class into a module in one of the following ways:

- DB2 Server
  - Stored procedures (JDBC or SQLJ)
  - SQL functions or user-defined functions (JDBC or SQLJ)
- Browser
  - Applets based on JDBC (JDBC)

- J2EE Application Servers (such as WebSphere Application Server)
  - Java Server Pages (JSPs) (JDBC)
  - Servlets (SQLJ or JDBC)
  - Enterprise JavaBeans (EJBs) (SQLJ or JDBC)

Available JDBC driver for DB2 UDB

DB2 UDB V8.2 is J2EE 1.4 and JDBC 3.0 compliant. It also supports the JDBC 2.1. Table 7-3 shows you the JDBC drivers delivered by IBM. An overview of all available JDBC drivers can be found at:

http://servlet.java.sun.com/products/jdbc/drivers

Table 7-3 JDBC Drivers

Type	Driver URL
Type 2	COM.ibm.db2.jdbc.app.DB2Driver (only for applications)
Type 3	COM.ibm.db2.jdbc.net.DB2Driver (only for applets)
Type 4	com.ibm.db2.jcc.DB2Driver(for applications and applets)

The type 3 and 4 drivers require you to provide the user ID, password, host name and a port number. For the type 3 driver, the port number is the applet server port number. For the type 4 driver the port number is the DB2 UDB server port number. The type 2 driver implicitly uses the default value for user ID and password from the DB2 client catalog, unless you explicitly specify alternative values. The JDBC Type 1 driver is based on a JDBC-ODBC bridge. Therefore, an ODBC driver can be used in combination with this JDBC driver (provided by Sun). IBM does not provide a Type 1 driver, and it is not a recommended environment.

After coding your program, compile it as you would with any other Java program. You do not need to perform any special precompile or bind steps.

7.5.2 Converting applications to use DB2 CLI/ODBC

DB2 Call Level Interface (DB2 CLI) is the IBM callable SQL interface to the DB2 family of database servers. It is a C and C++ application programming interface for relational database access that uses function calls to pass dynamic SQL statements as function arguments. It is an alternative to embedded dynamic SQL, but unlike embedded SQL, DB2 CLI does not require host variables or a precompiler.

DB2 CLI is based on the Microsoft *Open Database Connectivity* (ODBC) specification, and the International Standard for SQL/CLI. These specifications were chosen as the basis for the DB2 Call Level Interface in an effort to follow

industry standards, and to provide a shorter learning curve for those application programmers already familiar with either of these database interfaces. In addition, some DB2 specific extensions have been added to help the application programmer specifically exploit DB2 features.

The DB2 CLI driver also acts as an ODBC driver when loaded by an ODBC driver manager. It conforms to ODBC 3.51.

Comparison of DB2 CLI and Microsoft ODBC

Figure 7-4 compares DB2 CLI and the DB2 ODBC driver. The left side shows an ODBC driver under the ODBC Driver Manager, and the right side illustrates DB2 CLI, the callable interface designed for DB2 UDB specific applications.

Figure 7-4 DB2 CLI and ODBC

Archived

In an ODBC environment, the DriverManager provides the interface to the application. It also dynamically loads the necessary driver for the database server that the application connects to. It is the driver that implements the ODBC function set, with the exception of some extended functions implemented by the Driver Manager. In this environment, DB2 CLI conforms to ODBC 3.51.

For ODBC application development, you must obtain an ODBC Software Development Kit. For the Windows platform, the ODBC SDK is available as part of the *Microsoft Data Access Components* (MDAC) SDK, available for download from:

http://www.microsoft.com/data/

For non-Windows platforms, the ODBC SDK is provided by other vendors.

In environments without an ODBC driver manager, DB2 CLI is a self-sufficient driver, which supports a subset of the functions provided by the ODBC driver.

7.5.3 Converting ODBC applications

The Open Database *Connectivity* (ODBC) is similar to the CLI standard. Applications based on ODBC are able to connect to the most popular databases. Thus, the application conversion is relatively easy. You have to perform the conversion of database specific items in your application such as:

- Proprietary SQL query changes
- Possible changes in calling stored procedures and functions
- Possible logical changes

And, then proceed to the test, roll-out, and education tasks as well. Your current development environment will be the same.

7.6 General data conversion steps

This section briefly discusses the different steps needed to prepare the environment to receive the data. The methods employed can vary, but at a minimum, the following steps are required:

- Converting the database structure.
- Converting the database objects/content.
- Modifying the application.
- Modifying the database Interface.
- Modifying the data load and update processes
- Migrating the data.
- Testing

Converting the database structure

After you assess and plan the conversion, the first step to take is to either move or duplicate the structure of the source database onto a DB2 UDB system. Before this can happen, differences between the source and destination (DB2 UDB) structures must be addressed. These differences can result from different interpretation of SQL standards, or the addition or omission of particular functions. The differences can often be fixed syntactically, but in some cases, you must add functions or modify the application.

Metadata is the logical Entity-Relationship (E-R) model of the data, and describes the meaning of each entity, the relations that exist, and the attributes. From this model, the SQL Data Definition Language (DDL) statements that can be used to create a database that can be captured. If the database structure is already in the form of metadata (that is, a modeling tool was used in the design of the system), it is often possible to have the modeling tool generate a new set of DDL that is specific to DB2 UDB. Otherwise, the DDL from the current system must be captured and then modified into a form that is compatible with DB2 UDB. After the DDL is modified, it can be loaded and executed to create a new database (tables, indexes, constraints, and so on).

There are three approaches that can be used to move the structure of a DBMS:

- **Manual methods:** Dump the structure, import it to DB2 UDB, and manually adjust for problems
- **Metadata transport:** Extract the metadata (often called the “schema”) and import it to DB2 UDB
- **Porting and migration tools:** Use a tool to extract the structure, adjust it, and then implement it in DB2 UDB

Manual methods

Typically a DBMS offers a utility that extract the database structure and deposit it into a text file. The structure is represented in DDL, and can be used to recreate the structure on another database server. However, before the DDL will properly execute in DB2 UDB, it is likely that changes are needed to bring the syntax from the source system into line with DB2 UDB. So, after you extract the DDL, and transport it to DB2 UDB, you will likely have to edit the statements.

Besides syntactic differences, there may also be changes needed in data type names and in the structure. It is often easiest to simply run a small portion of the source DDL through DB2 UDB, and examine the errors. Please also see the appropriate DB2 UDB porting guide for more detail on the differences in syntax, names, and structure that you can expect at:

http://www-3.ibm.com/software/data/db2/migration/

Metadata transport

Many database structures are designed and put in place using modeling tools. These tools let the designer specify the database structure in the form of *entities* and *relationships*. The modeling tool then generates database definitions from the E-R description. If the system to be ported was designed (and maintained) using one of these tools, porting the database structure to DB2 UDB can be as simple as running the design program, and specifying an output of the form compatible with DB2 UDB.

Porting and migration tools

Probably the most popular means of porting a database structure (and other portions of a DBMS) today is the use of a porting and migration tool that cannot only connect to and take structural information from the source database, but can also modify and then deposit it in the destination database. As mentioned above, the IBM DB2 Migration ToolKit can be used to perform the migration using this method.

Converting the database objects

Database objects (stored procedures, triggers, and user-defined functions) are really part of the application logic that is contained within the database. Unfortunately, most of these objects are written in a language that is very specific to the source DBMS, or are written in a higher-level language that then must be compiled and somehow associated or bound to the target DBMS for use.

Capturing the database objects can often occur at the same time that the database structure is captured if the objects are written in an SQL-like procedural language and stored within the database (for this, you would use one of the porting and migration tools). For those objects written in higher-level languages (Java, C, and PERL), capture and import means transferring the source files to the DB2 UDB system and finding a compatible compiler and binding mechanism.

Stored procedures and triggers will have to be converted manually unless the tool used to extract the objects understands the stored procedure languages of both the source DBMS and DB2 UDB. The IBM DB2 Migration ToolKit is an example of a tool that can aid in the conversions of stored procedures and triggers from various DBMSs to DB2 UDB. Expect many inconsistencies between the dialects of procedural languages, including how data is returned, how cursors are handled, and how looping logic is used (or not used).

Objects that are written in higher-level languages must usually be dealt with manually. If embedded SQL is included in the objects, it can be extracted and run through a tool that might be able to help convert the SQL code to be compatible with DB2 UDB. After that, each section can be replaced and then compiled with the modified higher-level code.

Note that conversion of objects will require testing of the resulting objects. This means that test data will be needed (and must be populated into the database structure) before testing can occur. Therefore, one of the first tasks will be to generate test data.

After the conversion is completed, some adjustments will probably still be required. Issues such as identifier length may still need to be addressed. This can be done manually (looking at statistics, i.e. all database names over a certain length, and then doing a global search and replace on the names that appeared), or by using a tool (such as the IBM DB2 Migration ToolKit) that understands what to look for and how to fix it.

Modifying the application

While the porting of the database structure and objects can be automated to some extent using porting and migration tools, application code changes will mostly require manual conversion. If all database interaction is restricted to a database access layer, then the scope and complexity of necessary changes is well defined and manageable. However, when database access is not isolated to a database access layer (that is, it is distributed throughout application code files, contained in stored procedures and/or triggers, or used in batch programs that interact with the database), then the effort required to convert and test the application code depends on how distributed the database access is and on the number of statements in each application source file that require conversion.

When porting an application, it is important to first migrate the database structure (DDL) and database objects (stored procedures, triggers, user-defined functions, and so on). It is then useful to populate the database with a test set of data so that the application code can be ported and tested incrementally.

Few tools are available to port actual application code since much of the work is dependent upon vendor-specific issues. These issues include adjustments to logic to compensate for differing approaches to transaction processing, join syntax, use of special system tables, and use of internal registers and values. Manual effort is normally required to make and test these adjustments.

Often, proprietary functions used in the source DBMS will have to be emulated under DB2 UDB, usually by creating a DB2 UDB user defined function and/or stored procedure with the same name as the proprietary one being ported. This way, any SQL statements in the application code that call the proprietary function in question will not need to be altered. Migration tools such as the IBM DB2 Migration ToolKit are equipped with some of the most commonly used vendor-specific functions and will automatically create a DB2 UDB-equivalent function (or stored procedure) during the migration process.

Another issue when porting high-level language code (such as C, C++, Java, and COBOL) involves compiler differences. Modifications to the application code may be required if a different compiler and/or object library are used in the DB2 UDB environment (which may be caused by the selection of a different hardware or OS platform). It is vital to fully debug and test such idiosyncrasies before moving a system into production.

For more information on various application development topics relating to DB2 UDB, and to view various code samples, visit the DB2 Universal Database v8 Developer Domain Web page on the IBM Web site:

`http://www7b.software.ibm.com/dmdd/`

Modifying the database interface

Applications that connect to the source database using a standardized interface driver, such as ODBC and JDBC, usually require few changes to work with DB2 UDB. In most cases, simply providing the DB2 UDB supported driver for these interfaces is enough for the application to be up and running with a DB2 UDB database.

There are certain circumstances where the DB2 UDB-supported driver for an interface does not implement or support one or more features specified in the interface standard. It is in these cases where you must take action to ensure that application functionality is preserved after the port. This usually involves changing application code to remove references to the unsupported functions and either replacing them with supported ones, or simulating them by other means.

Applications that use specialized or native database interfaces (Oracle's OCI as an example) will require application code changes. Such applications can be ported using the DB2 UDB native CLI interface, or by using a standardized interface such as ODBC, and JDBC. If porting to CLI, many native database-specific function calls will need to be changed to the CLI equivalents; this is not usually an issue as most database vendors implement a similar set of functions. The DB2 UDB CLI is part of the SQL standard and mappings of functions between other source DBMS and DB2 UDB CLI can be found in the applicable DB2 UDB porting guide.

DB2 UDB also provides a library of administrative functions for applications to use. These functions are used to develop administrative applications that can administer DB2 UDB instances, backup and restore databases, import and export data, and perform operational and monitoring functions. These administrative functions can also be run from the DB2 UDB Command Line Processor (CLP), Control Center, and DB2 UDB scripts.

Migrating the data

You can move data (often called *migration*) from one DBMS to another by using numerous commercially available tools (including porting and migration tools such as the IBM DB2 Migration ToolKit, IBM WebSphere DataStage, and others).

In many cases, as the data is moved, it is also converted to a format that is compatible with the new DBMS (DATE/TIME data is a good example). This process can be quite lengthy when there is a large amount of data, which makes it quite important to have the conversions well defined and tested.

For large volumes of data, it is a good practice to develop a strategy for matching up the data as it is migrated and converted. Typically we migrate minutes, days, weeks, or months of historic data. Design a process where data can be loaded in terms of the number of days data converted, per elapsed day. Then you can determine how many days are needed for the entire conversion. Each conversion batch should be designed to be recoverable and re-startable.

In addition, it is possible to design routines so that sequences of days can be applied independently of each other. This can greatly help with error recovery if, for example, problems are subsequently found with the data for any particular range of days.

In some cases, it will still be necessary to do some customized conversions (specialized data, such as time series, and geo-spatial, may require extensive adjustments to work in the new DBMS). This is usually accomplished through the creation of a small program or script.

Testing

Once the data migration is completed, various methods, such as executing scripts on both source and target systems, can be employed to check the success rate of the migration effort. Application testing is also required to check if the changes made to the application during conversion are effective. The validity of the data can be checked either by running in-house developed database scripts or using third party testing tools.

Archived

Further information

You can find more information about the topics discussed in this chapter in the following materials:

- Microsoft SQL Server to DB2 UDB Conversion Guide, SG24- 6672
- Oracle to DB2 UDB Conversion guide,SG24-7049
- *DB2 UDB Call Level Interface Guide and Reference, volumn1,* SC09-4849; *volumn2,* SC09-4850
- *DB2 UDB Application Development Guide: Building and Running Applications,* SC09-4825
- *DB2 UDB Application Development Guide: Programming Client Applications,* SC09-4826
- Web site:

  http://www.ibm.com/software/data/db2/udb/ad

Archived

8

Performance and consolidation

In this chapter we discuss the topic of performance. The intent is not to discuss performance in general, but only as it applies to a consolidated data warehouse environment. For example, there may be concerns from users that when their data marts are consolidated into the enterprise data warehouse, their performance may degrade.

This is a valid concern, and steps need to be taken to prevent it. What were once separate workloads will now coexist on the same server. Therefore, there will be increased opportunity for contention between workloads, which could impact server size requirements. Alternatively, each of the smaller workloads are run on a larger server and can draw on the additional capacity. This could result in a significant boost in performance. By proactively setting expectations, and focusing on performance related parameters when planning the consolidation effort, you can avoid these types of issues.

In this chapter, we discuss the following topics:

- Performance management
- Performance techniques
- Refresh considerations
- Impact of loading and unloading the data

© Copyright IBM Corp. 2005. All rights reserved.

Performance tuning is a separate task, and can be defined as the modification of the systems and application environment in order to satisfy previously defined performance objectives. Most contemporary environments range from standalone systems to complex combinations of database servers and clients running on multiple platforms. Critical to all these environments is the achievement of adequate performance to meet business requirements. Performance is typically measured in terms of response time, throughput, and availability.

The performance of any environment is dependent upon many factors including system hardware and software configuration, number of concurrent users, and the application workload. You need well-defined performance objectives, or service level agreements (SLA), that have been negotiated with users to clearly understand the objectives and requirements.

The general performance objectives can be categorized as follows:

- **Realistic:** They should be achievable given the current state of the technology available. For example, setting sub-second response times for applications or transactions to process millions of rows of data may not be realistic.
- **Reasonable:** While the technology may be available, the business processes may not require stringent performance demands. For example, demanding sub-second response times for analytic reports that need to be studied and analyzed in detail before making a business decision, while achievable, may not be considered a reasonable request or responsible use of time, money, and resources.
- **Quantifiable:** The objectives must use quantitative metrics, such as numbers, ratios, or percentages, rather than qualitative metrics such as very good, average, or poor. An example of a quantitative metrics could specify that 95% of the transactions of a particular type, must have sub-second response time. A qualitative metric could specify that system availability should be very high.
- **Measurable:** The particular parameter must be capable of being measured. This is necessary to determine conformance or non-conformance with performance objectives. Units of measurement include response time for a given workload, transactions per second, I/O operations, CPU use, or some combination of these. Setting a performance objective of sub-second response times for a transaction is irrelevant if there is no way it can be measured.

Important: You need well-defined performance objectives, and the ability to measure the relevant parameters, to verify that the SLA is being met.

8.1 Performance techniques

In this section we describe some database techniques, and DB2 tools, that can be used to improve performance in the data warehousing environment. They will also apply to such activities as performing a data refresh process for data marts.

8.1.1 Buffer pools

Buffer pools tend to be one of the major components that can have the most dramatic impact on performance, since they have the potential to reduce I/Os. A buffer pool improves database system performance by allowing data to be accessed from memory instead of from disk. Because memory access is much faster than disk access, the less often the database manager needs to read from, or write to a disk, the better the performance.

A buffer pool is memory used to cache both user and system catalog table and index pages as they are being read from disk, or being modified. A buffer pool is also used as overflow for sort operations.

Note: Large objects (LOBs) and long fields (LONG VARCHAR) data are not manipulated in the buffer pool.

In general, the more memory that is made available for buffer pools, without incurring operating system paging, the better the performance.

DB2 is very good at exploiting memory. Therefore, a small increase in overall system cost to provide more memory can result in a much larger gain in throughput.

Large buffer pools provide the following advantages:

- They enable frequently requested data pages to be kept in the buffer pool, which allows quicker access. Fewer I/O operations can reduce I/O contention, thereby providing better response times and reducing the processor resource needed for I/O operations.
- They provide the opportunity to achieve higher transaction rates with the same response time.
- They reduce I/O contention for frequently used disk storage devices such as frequently referenced user tables and indexes. Sorts required by queries also benefit from reduced I/O contention on the disk storage devices that contain the temporary table spaces.

Archived

Best practices

The following list describes objects for which you should consider creating and using separate buffer pools:

- SYSCATSPACE, system catalog tablespace
- Temporary table spaces
- Index table space
- Table spaces that contain frequently accessed tables
- Table spaces that contain infrequently accessed, randomly accessed, or sequentially accessed tables

For further details on buffer pools, refer to the IBM Redbook: *DB2 UDB ESE V8 non-DPF Performance Guide for High Performance OLTP and BI*, SG24-6432.

8.1.2 DB2 RUNSTATS utility

The **runstats** utility collects statistics about the physical characteristics of a table and its associated indexes, and records them in the system catalog. These characteristics include the number of records, number of pages, average record length, and data distribution statistics. This utility also gathers statistics about data within DB2 tables and indexes, and these statistics are used by the DB2 optimizer to generate optimal query access plans.

The following key options can impact the performance of the **runstats** utility, but provide detailed statistics of significant benefit to the DB2 optimizer in its access path selection:

- **WITH DISTRIBUTION** clause
- **DETAILED** clause
- **LIKE STATISTICS** clause

WITH DISTRIBUTION clause

The **runstats** utility, by default, collects information about the size of the table, the highest and lowest values in the index(es), the degree of clustering of the table to any of its indexes, and the number of distinct values in indexed columns. However, when the optional **WITH DISTRIBUTION** clause is specified, the **runstats** utility collects additional information about the distribution of values between the highest and lowest values, as well.

The DB2 optimizer can exploit this additional information to provide superior access paths to certain kinds of queries when the data in the table tends to be skewed.

Archived

DETAILED clause

The **runstats** utility also provides an optional **DETAILED** clause which collects statistics that provide concise information about the number of physical I/Os required to access the data pages of a table if a complete index scan is performed under different buffer sizes. As **runstats** scans the pages of the index, it models the different buffer sizes, and gathers estimates of how often a page fault occurs. For example, if only one buffer page is available, each new page referenced by the index results in a page fault.

Each row might reference a different page, which could at most result in the same number of I/Os as rows in the indexed table. At the other extreme, when the buffer is big enough to hold the entire table (subject to the maximum buffer size), then all table pages are read once.

This additional information helps the optimizer make better estimates of the cost of accessing a table through an index.

The SAMPLED option, when used with the DETAILED option, allows **runstats** to employ a sampling technique when compiling the extended index statistics. If this option is not specified, every entry in the index is examined to compute the extended index statistics. This can dramatically reduce run times and overhead for runstats when run against very large tables.

LIKE STATISTICS clause

This optional clause collects additional column statistics (**SUB_COUNT** and **SUB_DELIM_LENGTH** in **SYSSTAT.COLUMNS**) for string columns only.

This additional information helps the DB2 optimizer make better selectivity estimates for predicates of the type "**column_name LIKE '%xyz'**" and "**column_name LIKE '%xyz%'**", and thereby generate a superior access path for the query.

The performance of the **runstats** utility depends upon the volume of data, the number of indexes associated with it, and the degree of detailed information requested via the **WITH DISTRIBUTION** and **DETAILED** clauses.

The following performance considerations apply:

- The statistical information collected by the **runstats** utility is critical to the DB2 optimizer selection of an optimal access path, and it is therefore imperative that such information be kept up to date. However, **runstats** consumes significant CPU and memory resources and should only be executed when significant changes have occurred to the underlying data that impact current statistics information and consequently the selection of an optimal access path by the DB2 optimizer.

  This implies that the frequency of **runstats** execution should be managed.

- The degree of statistical detailed information requested has a direct impact on the performance of the **runstats** utility. Specifying the **WITH DISTRIBUTION** clause with some or all columns, and/or the **DETAILED** clause, results in significant CPU and memory consumption. In particular, the database configuration parameter **stat_heap_sz** should be adjusted to accommodate the collection of detailed statistics.

Consider using the SAMPLED option of the DETAILED clause to reduce CPU consumption — this is of particular benefit in BI environments.

8.1.3 Indexing

The efficacy of an index is ultimately measured by whether or not it is used in queries whose performance it is meant to improve.

Indexes provide the following functionality:

- Enforcement of the uniqueness constraints on one or more columns.
- Efficient access to data in underlying tables when only a subset of the data is required, or when it is faster than scanning the entire table.

Indexes can therefore be used to:

- Ensure uniqueness
- Eliminate sorts
- Avoid table scans where possible
- Provide ordering
- Facilitate data clustering for more efficient access
- Speed up table joins

DB2 provides the Design Advisor wizard to recommend indexes for a specific query or workload. It can assist the DBA in determining indexes on a table that are not being used.

DB2 UDB also has an index called a Type 2 index, which offers significant concurrency and availability advantages over the previous index structure (Type 1 index). For details on the structure and concurrency characteristics of Type 2 indexes, refer to the *DB2 UDB Administration Guide: Performance*, SC09- 4821.

> **Note:** Type 1 and Type 2 indexes cannot coexist on the same table. All indexes on a table must be of the same type.

Performance considerations

While indexes have the potential to significantly reduce query access time, the trade-off is in disk space utilization, slower updates (SQL **INSERT, UPDATE**, and

DELETEs), locking contention, and administration costs (**runstats**, **reorg**). Each additional index potentially adds an alternative access path for a query for the optimizer to consider, which increases the compilation time.

> **Note:** Type 2 indexes consume more space than Type 1 indexes.

Best practices

To achieve superior index performance, consider the following methods when adopting best practices:

1. Use the DB2 Design Advisor, to find the best indexes for a specific query or for the set of queries that defines a workload.
2. Consider eliminating some sorts by defining primary keys and unique keys. But as you may be aware, there are trade-offs to consider with indexing. These are based on such things as table size, number and type of queries being run, and overall workload. The trade-off is typically one of improved query performance and workload throughput versus the cost of creating and maintaining the indexes.
3. Add INCLUDE columns to unique indexes to improve data retrieval performance. Good candidates are columns that:
   - Are accessed frequently and would therefore benefit from index-only access.
   - Are not required to limit the range of index scans.
   - Do not affect the ordering or uniqueness of the index key.
   - Are updated infrequently.
4. To access small tables efficiently, use indexes to optimize frequent queries to tables with more than a few data pages. Create indexes on the following:
   - Any column you will use when joining tables.
   - Any column from which you will be searching for particular values on a regular basis.
5. To search efficiently, order the keys in either ascending or descending order depending on which will be used most often. Although the values can be searched in reverse direction by specifying the **ALLOW REVERSE SCANS** parameter in the **CREATE INDEX** statement, scans in the specified index order perform slightly better than reverse scans.
6. To save index maintenance costs and space:
   - Avoid creating indexes that are partial keys of other index keys on the columns. For example, if there is an index on columns a, b, and c, then a second index on columns a and b is typically not useful.

Archived

- Do not arbitrarily create indexes on all columns. Unnecessary indexes not only use space, but also cause large prepare times. This is especially important for complex queries, when an optimization class with dynamic programming join enumeration is used. Unnecessary indexes also impact update performance in OLTP environments.

7. To improve performance of delete and update operations on the parent table, create indexes on foreign keys.
8. For fast sort operations, create indexes on columns that are frequently used to sort the data.
9. To improve join performance with a multiple-column index, if you have more than one choice for the first key column, use the column most often specified with the "=" (equating) predicate, or the column with the greatest number of distinct values as the first key.
10. To help keep newly inserted rows clustered according to an index, define a clustering index. Clustering can significantly improve the performance of operations such as prefetch and range scans. Only one clustering index is allowed per table. A clustering index should also significantly reduce the need for reorganizing the table.

    Use the **PCTFREE** keyword when you define the index to specify how much free space should be left on the page to allow inserts to be placed appropriately on pages. You can also specify the pagefreespace **MODIFIED BY** clause of the **LOAD** command.
11. To enable online index defragmentation, use the **MINPCTUSED** option when you create indexes. **MINPCTUSED** specifies the threshold for the minimum amount of used space on an index leaf page before an online index defragmentation is attempted. This might reduce the need for reorganization at the cost of a performance penalty during key deletions if these deletions physically remove keys from the index page.
12. The **PCTFREE** parameter in the **CREATE INDEX** statement specifies the percentage of each index leaf page to leave as free space. For non-leaf pages, it will choose the value you specify—unless the value that you specify is less than 10%, in which case the 10% value is chosen.

    Choose a smaller value for **PCTFREE** to save space and index I/Os in the following cases:

    - The index is never updated.
    - The index entries are in ascending order and mostly high-key values are inserted into the index.
    - The index entries are in descending order and mostly low-key values are inserted into the index.

A larger value for **PCTFREE** should be chosen if the index gets updated frequently in order to avoid page splits, which reduce performance because they result in index pages no longer being sequential or contiguous. This has a negative impact on prefetching, and potentially space consumption as well, depending upon the key values being inserted/updated.

13. Ensure that the number of index levels in the index tree are kept to a minimum (less than 4, if possible); this is the **NLEVELS** column in the **SYSCAT.INDEXES** catalog table. The number of levels in the index is affected by the number of columns in the key and the page size of the table space in which it is created.

8.1.4 Efficient SQL

SQL is a high level language that provides considerable flexibility in writing queries which can deliver the same answer set. However, not all forms of the SQL statement deliver the same performance for a given query. It is therefore vital to ensure that the SQL statement is written in a manner to provide optimal performance.

Best practices

Here are some considerations for choosing between dynamic and static SQL:

1. Static SQL statements are well-suited for OLTP environments that demand high throughput and very fast response times. Queries tend to be simple, retrieve few rows and stable index access is the preferred access path.

   Note: Keeping table and index statistics up-to-date helps the DB2 optimizer choose the best access plan. However, SQL packages need to be rebound for the DB2 optimizer to generate a new access plan based on these statistics.

2. Dynamic SQL statements are generally well-suited for applications that run against a rapidly changing database, where queries need to be specified at run time. This is typical of BI environments. If literals are used, each time the statement is run using a new value for the literal requires DB2 to do a **PREPARE**. CPU time must be used to do the **PREPARE** and in high volume applications, this can be quite expensive.

   A parameter marker is represented with a question mark (?) in place of the literal in the SQL statement. The parameter marker is replaced with a value at run time by the application. Therefore, the SQL statement can be reused from the package cache and does not require a subsequent **PREPARE**. This results in faster query execution and reduced CPU consumption. Dynamic SQL is appropriate for OLTP environments as well. When used in OLTP

Archived

environments in particular, we strongly recommend the use of parameter markers in dynamic SQL to achieve superior performance.

> **Note:** Keeping table and index statistics up-to-date helps the DB2 optimizer choose the best access plan. However, unlike the case with static SQL, packages with dynamic SQL do *not* need to be rebound after new indexes have been added and/or new statistics have been gathered. But the package cache needs to be flushed via the `FLUSH PACKAGE CACHE` command to ensure that the new statistics are picked up.

3. For OLTP environments characterized by high concurrent activity, simple SQL statements and sub-second response time requirements, the optimization class should be set (`SET CURRENT QUERY OPTIMIZATION` statement) to a lower value such as 1 or 2. If the optimization level is not set in the `CURRENT QUERY OPTIMIZATION` special register, the DB2 optimizer will table the value set in the `DFT_QUERYOPT` database configuration parameter.

Minimize the number of SQL statements issued

Avoid using multiple SQL statements when the same request can be issued using one SQL statement. This minimizes the cost of accessing DB2, and also provides more information in an SQL statement, which enables the DB2 optimizer to choose a more optimal access path.

Limit the volume of data returned: columns and rows

SQL performance is enhanced by specifying only the columns of interest in the select list of a query, and limiting the number of rows accessed using predicates.

Avoid the "`SELECT *`" construct, which specifies that all columns are to be returned in the result set, resulting in needless processing.

8.1.5 Multidimensional clustering tables

Multidimensional clustering (MDC) provide a significant way to improve performance, and a maintenance advantage for data marts.

The performance of an MDC table is greatly dependent upon the proper choice of dimensions, and the block (extent) size of the table space for the given data and application workload.

A poor choice of dimensions and extent size can result in unacceptable disk storage utilization and poor query access performance, as well as `load` utility processing.

Choosing dimensions

The first step is to identify the queries in the in existing or planned workloads that can benefit from block-level clustering:

- For existing applications, the workload may be captured from the dynamic SQL snapshot and the SQL statement Event Monitor. The DB2 Query Patroller or other third party tools may also assist with such a determination.
- For future applications, this information will have to be obtained from requirements gathering.

Choosing the extent size

Extent size is related to the concept of *cell density*, which is the percentage of space occupied by rows in a cell. Since an extent only contains rows with the same unique combination of dimension values, significant disk space could be wasted if dimension cardinalities are very high; the worst case scenario is a dimension with unique values which would result in an extent per row.

The ideal MDC table is the one where every cell has just enough rows to exactly fill one extent. This can be difficult to achieve. The objective of this section is to outline a set of steps to get as close to the ideal MDC table as possible.

> **Note:** The extent size is associated with a table space, and therefore applies to *all* of the dimension block indexes as well as the composite block index. This makes the goal of high cell density for every dimension block index and the composite block index very difficult to achieve.

Defining small extent sizes can increase cell density, but increase the number of extents per cell resulting in more I/O operations, and potentially poorer performance when retrieving rows from this cell. However, unless the number of extents per cell is excessive, performance should be acceptable. If *every* cell occupies more than one extent, then it can be considered to be excessive.

Sometimes, due to data skew, some cells will occupy a large number of extents while others will occupy a very small percentage of the extent. In such cases, it would signal a need for a better choice of dimension keys. Currently, the only way to determine the number of extents per cell requires the DBA to issue appropriate SQL queries or use **`db2dart`**.

Performance might be improved if the number of blocks could be reduced by consolidation. However, unless the number of extents per cell is excessive, this situation is not considered a problem.

Note: The challenge here is to find the right balance between sparse blocks/extents and minimizing the average number of extents per cell as the table grows to meet future requirements.

Best practices

The following things should be considered when your objective is to achieve superior performance with MDCs:

1. Choose dimension columns that are good candidates for clustering, such as:
   - Columns used in high priority complex queries
   - Columns used in range, equality, and IN predicates such as:

     ```
     shipdate>'2002-05-14', shipdate='2002-05-14', year(shipdate) in
     (1999, 2001, 2002)
     ```

   - Columns that define roll-in or roll-out of data such as:

     ```
     delete from table where year(shipdate) = '1999'
     ```

   - Columns with coarse granularity
   - Columns referenced in a GROUP BY clause
   - Columns referenced in an ORDER BY clause
   - Foreign key columns in the fact table of a star schema database
   - Combinations of the above

Note: Avoid columns that are updated frequently.

2. If expressions are used to cluster data with generated columns, then the expression needs to be monotonic.

   Monotonic means that an increasing range of values on the base column corresponds to a range of values on the generated column that is never decreasing. For example:

   ```
   if (A > B) then expr(A) >= expr(B) and
   if (A < B) then expr(A) <= expr(B)
   ```

   In other words, as "A" increases in value, the expression based upon "A" also increases or remains constant.

   Examples of monotonic operations include:

   ```
   A + B
   A * B
   integer(A).
   ```

Examples of non-monotonic operations are:

```
A - B
month(A)
day(A)
```

The expression `month(A)` is non-monotonic because as “A” increases, the value of the expression fluctuates, as follows:

```
month(20010531) equals 05
month(20021031) equals 10
month(20020115) equals 01
```

So, as the date value increases, the value of the month fluctuates.

Note: If the SQL compiler cannot determine whether or not an expression is monotonic, the compiler assumes that the expression is *not* monotonic.

3. Do not choose too many dimensions without determining cell density; avoid too many sparse extents/blocks.
4. Once the dimensions have been selected, order them to satisfy the performance of high priority queries.

   When an MDC table is created, a composite block index is automatically created in addition to the dimension block indexes. While this index is used to insert records into the table, it can also be used like any other multi-column index as an access path for a query. Therefore, an appropriate ordering of the dimension columns can enhance the performance of certain types of queries with ORDER BY clauses and range predicates.
5. For best disk space utilization of an MDC table and best I/O performance, consider the following parameters:
   - Extent size
   - Granularity of one or more dimensions
   - Number of candidate dimensions
   - Different combinations of dimensions

Note: Each of these changes requires the MDC table to be dropped and recreated. Therefore, it is best to consider them during the design process than to change them later. In practice, the guidelines for setting up MDCs are very straightforward.

MDCs are particularly suited for BI environments which involve star schemas, and queries that retrieve large numbers of rows along multiple dimensions. We strongly recommend that anyone considering a migration to an MDC table carefully model space utilization and cell utilization for candidate dimension keys, as well as the performance of high priority user queries, before committing to the selection of the dimension keys and extent size.

8.1.6 MQT

MQTs have the potential to provide significant performance enhancements to certain types of queries, and should be a key tuning option in the arsenal of every DBA. Like any other table, defining appropriate indexes on MQTs and ensuring that their statistics are current will increase the likelihood of their being used by the DB2 optimizer during query rewrite, and enhance the performance of queries that use them.

However, MQTs have certain overheads which should be carefully considered when designing them. These include:

- Disk space, due to the MQTs and associated indexes, as well as staging tables.
- Locking contention on the MQTs during a refresh.
- With deferred refresh, the MQT is offline while the `REFRESH TABLE` is executing.
- The same applies to the staging table if one exists. Update activity against base tables may be impacted during the refresh window.
- With immediate refresh, there is contention on the MQTs when aggregation is involved due to SQL insert, update, and delete activity on the base table by multiple transactions.
- Logging overhead during refresh of very large tables.
- Logging associated with staging tables.
- Response time overhead on SQL updating the base tables when immediate refresh and staging tables are involved, because of the synchronous nature of this operation.

Best practices

Here are some things to consider to achieve superior performance with MQTs. The main objective should be to minimize the number of MQTs required by defining sufficiently granular `REFRESH IMMEDIATE` and `REFRESH DEFERRED` MQTs that deliver the desired performance, while minimizing their overheads:

1. When an MQT has many tables and columns in it, it is sometimes referred to as a “wide” MQT. Such an MQT allows a larger portion of a user query to be matched, and hence provides better performance. However, when the query has fewer tables in it than in the MQT, we need to have declarative or informational referential integrity constraints defined between certain tables in order for DB2 to use the MQT for the query as discussed in. Note that a potential disadvantage of wide MQT is that they not only tend to consume more disk space, but may also not be chosen for optimization because of the increased costs of accessing them.

2. When an MQT has fewer columns and/or tables, it is sometimes referred to as a thin MQT. In such cases, we reduce space consumption at the cost of performing joins during the execution of the query. For example, we may want to only store aggregate information from a fact table (in a star schema) in the MQT, and pick up dimension information from the dimension tables through a join. Note that in order for DB2 to use such an MQT, the join columns to the dimension tables must be defined in the MQT. Note also that referential integrity constraints requirements do not apply to thin MQTs.
3. Incremental refresh should be used to reduce the duration of the refresh process. For the duration of a full refresh, DB2 takes a share lock on the base tables, and a z-lock on the MQT. Depending upon the size of the base tables, this process can take a long time. The base tables are not updatable for this duration, and the MQT may not be available for access or optimization either. Incremental refresh can reduce the duration of the refresh process, and increase the availability of the base tables and the materialized view. Incremental refresh should be considered when one or more of the following conditions exist:
   - The volume of updates to the base tables relative to size of the base tables is small.
   - The duration of read only access to the base tables during a full refresh is unacceptable.
   - The duration of unavailability of the MQT during a full refresh is unacceptable.

For further details on all these recommendations, refer to the IBM Redbook, *DB2 UDB's High Function Business Intelligence in e-business*, SG24-6546.

8.1.7 Database partitioning

Data warehouses are becoming larger and larger, and enterprises have to handle high volumes of data with multiple terabytes of stored raw data. To accommodate growth of data warehouses, Relational Database Management Systems (RDBMS) have to demonstrate near linear scalable performance as additional computing resources are applied. The administrative overhead should be as low as possible.

The Database Partitioning Feature (DPF) allows DB2 Enterprise Server Edition (DB2 ESE) clients to partition a database within a single server or across a cluster of servers. The DPF capability provides the customer with multiple benefits including scalability to support very large databases or complex workloads and increased parallelism for administration tasks. Therefore, we can add new machines and spread the database across them. In detail, we can spend more CPUs, memory and disks for the database. DB2 UDB ESE with DPF is an optimal way, to manage the DWE and as well as the OLTP workloads.

Note: Prior to Version 8, DB2 UDB ESE with DPF was known as DB2 UDB Enterprise Extended Edition (EEE).

DB2 UDB database partitioning refers to the ability to divide the database into separate and distinct physical partitions. Database partitioning has the characteristic of storing large amounts of data at a very detailed level while keeping the database manageable. Database utilities also run faster by operating on individual partitions concurrently. Figure 8-1 compares a single partition to a multi-partition database.

Figure 8-1 Figure 8-1Single partition database compared to a multi-partition database

The physical database partitions can then be allocated across a massively parallel processor (MPP) server, as depicted in Figure 8-2.

Figure 8-2 Figure 8-2 Sample MPP configuration

A single symmetric multi-processor (SMP) server, is shown in Figure 8-3.

Figure 8-3 Figure 8-3 Multi-partition database on an SMP server

A cluster of SMP servers, is shown in Figure 8-4. The database still appears to the end-user as a single image database. However, the database administrator can take advantage of single image utilities and partitioned utilities where appropriate.

Figure 8-4 Figure 8-4 Multi-Partition SMP cluster configuration

Database partitioning can be used to help improve BI performance, and help enable real-time capability. If we partition a single large database into a number of smaller databases for running the operation, the SQL statements and the refresh build phase for the data marts, runs in less time. The reason is, each database partition holds a separate set of the data.

For example, say that an SQL statement has to scan a table with 100 million rows. If the table exists in a singe database, then the database manager scans all 100 million records. But a partitioned database, with 50 database partition servers, then each database manager has only to scan two million rows.

Furthermore, another advantage of database partitioning is to override the memory limit of the 32 bit architecture. Since each database partition manages and owns its own resources, we can overcome the limit by partitioning the database.

The initial or daily load process can drastically reduce the time to build the data marts. That can move the data warehousing environment closer to real- time. The maintenance efforts, such as runstats, reorg, and backup, will be reduced, because each operation is run on a subset of data managed by the partition.

Multiple database partitions can increase transaction throughput by processing the insert and delete statements concurrently across the database partitions. This benefit also applies to the technique of selecting from one table and inserting into another.

8.2 Data refresh considerations

The process to refresh the data in data marts costs time, money, and resources. In addition, it also impacts the availability of those data marts. And, it is very important for the data marts to remain as available as possible. The cost, and availability impact, primarily depends on the load and unload techniques used.

A benefit of DMC is that many of these costs will be reduced or eliminated. For those instances where a data mart is still required, we need to consider the techniques for minimizing the impact. Now let us consider the following items:

- Data refresh types
- Impact analysis

8.2.1 Data refresh types

To keep the information in the data marts current, it must be refreshed on a periodic or continuous basis. It depends on the requirements of the business. Performing a data mart refresh can be similar to the initial load of the data mart, especially when the volume of changing data is very large. The size of the underlying base tables, their granularity, and the aggregation level of the data marts, determine the data refresh strategy.

Basically there are two types of data refresh:

- **Full refresh:** The full refresh process completely replaces the existing data with a new set of data. The full refresh can require significant time and effort, particularly if there are many base tables or large volumes of data.
- **Incremental refresh:** The incremental refresh process only performs insert, update, and delete operations on the underlying base tables rather than replacing the entire contents. Therefore, an incremental refresh is typically much faster — particularly when the volume of changes are small.

With either type of process, there can be an impact on performance and on availability of the data mart.

8.2.2 Impact analysis

Most typically, data marts are implemented to improve the SQL query performance and/or to improve availability. However, rarely is anything free. And it is the same here. There is typically an impact of some type, somewhere. Here are a few of the potential impacts you will need to consider:

- **Network load:** The refresh process can have an impact on the network load, in particular, when there is a high volume of base table data changes to move across the network. This is further exacerbated when the data marts span regions or even countries. The bandwidth of the network is often a limiting factor.
- **Disk capacity:** Additional disk capacity is needed to unload, transform, and stage the data as it goes through the refresh cycle. This of course is dependent on the volumes of data involved.
- **CPU, memory, and I/O load:** During the refresh process we have to consider the higher utilization of CPU, memory, and I/O on the data warehouse environment, as well as on the source systems. This can impact other users and the response time for reporting systems.
- **Availability of the data marts:** A full refresh process has a significant impact of the availability, and in this phase the data marts are not even accessible. Therefore, full refresh processes should be minimized, and only run during periods of low usage — such as during the night.
- **DB2 log files:** The DB2 log files keep records of changes to the database. Refresh methods that do not use the DB2 load utility, can significantly increase the size and number of DB2 logfiles. This can have an impact on performance because change records must be logged.
- **Integrity:** If we use the DB2 load utility for the refresh process, than we have to consider the integrity. This is because, the load utility does not enforce referential integrity, perform constraints checking, or update summary tables that are dependent on the tables being loaded.
- **Indexes:** All refresh processes have an impact of the indexes and also on the index and table statistics. It is important for SQL query performance, that we keep the statistic current. It is highly recommended that you use the DB2 RUNSTATS command to collect current statistics on tables and indexes. This provides the optimizer with the most accurate information with which to determine the best access plan.

8.3 Data load and unload

In this section we provide an overview of the DB2 utility used to move data across the DB2 family of databases. This utility can also be used to build and

refresh data marts, either one time or as a continuous or periodic process. In particular, the DB2 Load and the DB2 High Performance Unload are good candidates to enable the data warehousing environment to move closer to having a real-time capability. However, be aware that data movement tools can have a significant impact on data integrity. The load utility does not fire triggers, and does not perform referential or table constraints checking (other than validating the uniqueness of the indexes), because it writes formatted pages directly into the database and bypass the DB2 log files. As examples, they can impact:

- Column definitions (primary keys, foreign keys, and unique keys)
- Referential integrity
- Table indexes

Important: We have to consider that the data movement tools can work properly with different DB2 versions.

For more detailed information, please refer to the IBM Redbook, *Moving Data Across the DB2 Family*, SG24-6905.

8.3.1 DB2 Export and Import utility

DB2 has utilities to satisfy the requirements to import and export data. In this section we describe these activities.

Export

The DB2 Export utility is used to extract data from a DB2 database. The exported data can then be imported or loaded into another DB2 database, using the DB2 Import or the DB2 Load utility.

The Export utility exports data from a database to an operating system file or named pipe, which can be in one of several external file formats. This file with the extracted data can be moved to a different server.

The following information is required when exporting data:

- An SQL SELECT statement specifying the data to be exported
- The path and name of the operating system file that will store the exported data
- The format (IXF, DEL or WSF) of the data in the input file

The IXF file format results in an extract file consiting of both metadata and data. The source table (including its indexes) can be recreated in the target environment if the CREATE mode of the Import utility is specified. The recreation can only be done if the query supplied to the Export utility is a simple **SELECT ***

Next we show an IXF example of the export command specifying a message file and the select statement:

```
export to stafftab.ixf of ixf messages expstaffmsgs.txt select * from staff
```

At least, SELECT authorization is needed on the tables you export from.

The Export utility can be invoked through:

- The command line processor (CLP)
- The Export notebook in the Control Center
- An application programming interface (API)

The Export utility can be used to unload data to a file or a named pipe from a table residing in the following:

- Distributed database
- Mainframe database through DB2 Connect (only IXF format)
- Nickname representing a remote source table

Note: If performance is an issue, the DEL format covers your needs, and all rows are to be unloaded, then you should consider the High Performance Unload tool for Multiplatforms. The tool must be executed from the machine where the source table resides.

Import

The Import utility inserts data from an input file or a named pipe into a table or updatable view. The Import utility uses the SQL INSERT statement to write data from an input file into a specific table or view. If the target table or view already contains data, you can either replace or append to the existing data.

The following authorization is needed when using the Import utility to:

- Create a new table you must at least have CREATETAB for the database
- Replace data you must have SYSADM, DBADM or CONTROL
- Append data you must have SELECT and INSERT

The Import utility can be invoked through:

- The command line processor (CLP)
- The Import notebook in the Control Center
- An application programming interface (API)

The following information is required when importing data:

- The path and the name of the source file
- The name or alias of the target table or view
- The format of the data (IXF, DEL, ASC or WSF) in the source file

Archived

- Mode:
  - Insert
  - Replace
  - Update, if primary key matches are found
  - create

Among other options, you can also specify:

- Commit frequency
- Number of records to skip from input file before starting to Import

> **Note:** When creating a table from an IXF file, not all attributes of the original table are preserved. For example, referential constraints, foreign key definitions, and user-defined data types are not retained.

The import utility can be used to insert data from a file or a named pipe to a table in a:

- Distributed database
- Mainframe database through DB2 Connect (only IXF format)

Performance considerations

The following performance considerations apply:

- Since the **`import`** utility does SQL inserts internally, all optimizations available to SQL inserts apply to import as well, such as large buffer pools and block buffering.
- By default, automatic commits are not performed, and **`import`** will issue a commit at the end of a successful import. While fewer commits improve overall performance in terms of CPU and elapsed time, they can negatively impact concurrency and re-startability of **`import`** in the event of failure. In the case of a mass import, log space consumption could also become an issue and result in log full conditions, in some cases.

  The **`COMMITCOUNT n`** parameter specifies that a commit should be performed after every n records are imported. The default value is zero.
- By default, **`import`** inserts one row at a time into a target block and checks for the return code. This is less efficient that inserting a block at a time.

  The **`MODIFIED BY COMPOUND = x`** parameter (where x is a number between 1 and 100, inclusive) uses non-atomic compound SQL to insert the data, and x statements will be attempted each time. The **`import`** command will wait for the SQL return code about the result of the inserts after x rows instead of the default one row. If this modifier is specified, and the transaction log is not sufficiently large, the import operation will fail.

The transaction log must be large enough to accommodate either the number of rows specified by **COMMITCOUNT**, or the number of rows in the data file if **COMMITCOUNT** is not specified. It is therefore generally recommended to use **COMMITCOUNT** along with **COMPOUND** in order to avoid transaction log overflows.

Note: For performance, use the Load utility on distributed wherever it is possible, except for small amounts of data.

8.3.2 The db2batch utility

Exporting data in parallel into a partitioned database reduces data transfer execution time, and distributes the writing of the result set, as well as the generation of the formatted output, across nodes in a more effective manner than would otherwise be the case. When data is exported in parallel (by invoking multiple export operations, one for each partition of a table) it is extracted, converted on the local nodes, and then written to the local file system. In contrast, when exporting data serially (exporting through a single invocation of the Export utility) it is extracted in parallel and then shipped to the client, where a single process performs conversion and writes the result set to a local file system.

The **db2batch** command is used to monitor the performance characteristics and execution duration of SQL statements. This utility also has a parallel export function in partitioned database environments that:

- Runs queries to define the data to be exported
- On each partition, creates a file containing the exported data that resides on that partition

A query is run in parallel on each partition to retrieve the data on that partition. In the case of **db2batch -p s**, the original SELECT query is run in parallel. In the case of **db2batch -p t** and **db2batch -p d,** a staging table is loaded with the export data, using the specified query, and a **SELECT *** query is run on the staging table in parallel on each partition to export the data. To export only the data that resides on a given partition, **db2batch** adds the predicate NODENUMBER(*colname*) = CURRENT NODE to the WHERE clause of the query that is run on that partition. The *colname* parameter must be set to the qualified or the unqualified name of a table column. The first column name in the original query is used to set this parameter.

It is important to understand that **db2batch** runs an SQL query and sends the output to the target file, it does not use the Export utility. The Export utility options are not applicable to parallel export. You cannot export LOB columns using the **db2batch** command.

Run **db2batch -h** from the command window to see a complete description of command options.

The **db2batch** command executes a parallel SQL query and sends the output to a specified file. Note that the command is executing a select statement, not the Export utility. LOB columns, regardless of data length, cannot be exported using this method.

To export contents of the staff table in parallel, use the following command:

```
db2batch -p s -d sample -f staff.batch -r /home/userid/staff.asc -q on
```

In this example:

- The query is ran in parallel on a single table (**-p s** option)
- Connection is made to the sample database (**-d sample** option)
- The control file staff.batch contains the SQL select statement (**select * from staff**)
- Output is stored to staff.asc file, default output format is positional ASCII (remember that db2batch is not using the Export utility)
- Only the output of the query will be sent to the file (**-q on** option)

To export into a delimited ASCII file:

```
db2batch -p s -d sample -f emp_resume.batch -r /home/userid/emp_resume.del,
/home/mmilek/userid/emp_resume.out -q del
```

In this example:

- Only non-LOB columns from emp_resume table are selected (`select empno,resume_format from emp_resume`)
- emp_resume.del file contains the query output in delimited ASCII format (**-q del** option),
  , is the default column delimiter and **|** is the default char delimiter
- emp_resume.out contains the query statistics

8.3.3 DB2 Load utility

DB2 Load can load data from files, pipes or devices (such as tape) and from queues and tables. DB2 Load can operate in two load modes, load insert, which appends data to the end of a table, and load replace, which truncates the table before it is loaded. There are also two different indexing modes:

- Rebuild - which newly rebuilds all indexes
- Incremental - which extends the current indexes with the new data

Note: Load operations now take place at the table level. This means that the load utility no longer requires exclusive access to the entire table space, and concurrent access to other table objects in the same table space is possible during a load operation. When the COPY NO option is specified for a recoverable database, the table space will be placed in the backup pending table space state when the load operation begins.

The *ALLOW READ ACCESS* option is very useful when loading large amounts of data because it gives users access to table data at all times, even when the load operation is in progress or after a load operation has failed. The behavior of a load operation in *ALLOW READ ACCESS* mode is independent of the isolation level of the application. That is, readers with any isolation level can always read the pre-existing data, but they will not be able to read the newly loaded data until the load operation has finished. Read access is provided throughout the load operation except at the very end. Before data is committed the load utility acquires an exclusive lock (Z-lock) on the table. The load utility will wait until all applications that have locks on the table, release them. This may cause a delay before the data can be committed. The *LOCK WITH FORCE* option may be used to force off conflicting applications, and allow the load operation to proceed without having to wait.

Usually, a load operation in *ALLOW READ ACCESS* mode acquires an exclusive lock for a short amount of time; however, if the *USE* <tablespaceName> option is specified, the exclusive lock will last for the entire period of the index copy phase.

For large amount of data to be loaded it makes sense to use the *SAVECOUNT* parameter. In this case if a restart is necessary the load program restarts at the last save point instead of from the beginning.

The data can also be loaded from a user defined cursor. This capability, a new Load option with DB2 V8, is often referred as Cross Loader.

The following information is required when loading data:

- The name of the source file
- The name of the target table
- The format (DEL, ASC, IXF or CURSOR) of the source file

Note: The Load utility does not fire triggers, and does not perform referential or table constraints checking. It does validate the uniqueness of the indexes.

The Load utility can be invoked through:

- The command line processor (CLP)
- The Load notebook in the Control Center
- An application programming interface (API)

The Load utility loads data from a file or a named pipe into a table a:

- Local distributed database where the load runs
- Remote distributed database through a locally cataloged version using the CLIENT option

The Load utility is faster than the Import utility, because it writes formatted pages directly into the database, while the Import utility performs SQL INSERTs.

Note: The DB2 Data Propagator does not capture any changes in data done through the Load utility.

Loading MDC table

MDC tables are supported by a new physical structure that combines data, special kinds of indexes, and a block map. Therefore, MDC load operations will always have a build phase since all MDC tables have block indexes.

During the load phase, extra logging (approximately two extra log records per extent allocated) for the maintenance of the block map is performed. A system temporary table with an index is used to load data into an MDC tables. The size of the system temporary table is proportional to the number of distinct cells loaded. The size of each row in the table is proportional to the size of the MDC dimension key.

We recommend the following techniques to enhance the performance of loading an MDC table:

1. Consider increasing the database configuration parameter **`logbufsz`** to a value that takes into account the additional logging for the maintenance of the block map.
2. Ensure that the buffer pool for the temporary table space is large enough in order to minimize I/O against the system temporary table.
3. Increase the size of the database configuration parameter **`util_heap_sz`** by 10-15% more than usual in order to reduce disk I/O during the clustering of data that is performed during the load phase.
4. When the **`DATA BUFFER`** option of **`load`** command is specified, its value should also be increased by 10-15%. If the **`load`** command is being used to load several MDC tables concurrently, the **`util_heap_sz`** database configuration parameter should be increased accordingly.

8.3.4 The db2move utility

This command facilitates the movement of large numbers of tables between DB2 databases located on the distributed platforms.

The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in IXF format. The IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another platform and imported or loaded into a DB2 database on that platform.

This tool calls the DB2 Export, Import, and Load APIs, depending on the action requested by the user. Therefore, the requesting user ID must have the correct authorization required by those APIs, or the request will fail.

This tool exports, imports, or loads user-created tables. If a database is to be duplicated from one operating system to another operating system, **db2move** facilitates the movement of the tables. It is also necessary to move all other objects associated with the tables, such as: aliases, views, triggers, user-defined functions, and so on.

The load action must be run locally on the machine where the database and the data file reside. A full database backup, or a table space backup, is required to take the table space out of backup pending state.

DB2 UDB **db2move** is a common command and option interface to invoke the three utilities mentioned above.

8.3.5 The DB2 High Performance Unload utility

IBM DB2 High Performance Unload (HPU) for Multi-Platforms (MP) is a tool not included in the DB2 UDB product distribution. This product is purchased separately and installed on all DB2 server nodes.

The latest revision level will always be reflected at the Web site:

http://www-306.ibm.com/software/data/db2imstools/db2tools/db2hpu/

HPU for MP can increase performance by circumventing the database manager. Instead of accessing the database by issuing SQL commands against the DB2 database manager, as typical database applications do, HPU itself translates the input SQL statement and directly accesses the database object files. An unload from a backup image may be performed even if the DB2 database manager is not running. Active DB2 database manager is needed to verify that a user not belonging to the sysadm group does have authority needed to run the HPU tool.

Archived

HPU can unload data to flat files, pipes, and tape devices. Delimited ASCII and IXF file formats are supported. The user format option is intended to be used to create a file format compatible with the positional ASCII (ASC) formats used by the other DB2 tools and utilities. Creating multiple target files (location and maximum size can be specified) allows for better file system management.

A partitioned database environment HPU with FixPak 3, offers the following features:

- Data from all partitions can be unloaded to multiple target files.

  The syntax allows you to unload, with a single command, on the machine where the partition is, or to bring everything back to the machine you are launching HPU from. The command **`OUTPUT(ON REMOTE HOST "/home/me/myfile")`** creates a file per partition on the machine where the partition reside. Of course the path /home/me/ must exist on each machine impacted by the unload.

- A partitioned table can be unloaded into a single file.

  The command **`OUTPUT(ON CURRENT HOST "/home/me/myfile")`** creates only the file **`myfile`** on the machine you are running from, and will contain all the data of the unload. This is the default, for compatibility reasons, while multiple files will offer better performance.

> **Note:** The option **`ON "mynamedhost" HOST`** behaves like **`ON CURRENT HOST`**, except that the output file will be created on the specified host rather than the current host. A restriction exists that the named host must be part of the UDB nodes.

- A subset of table nodes can be unloaded by specifying command line options or through a control file or both.

  The OUTPUT command now supports the FOR PARTS() clauses. The appropriate combination of these clauses allows you the needed flexibility.

HPU tool is an executable externally to DB2 UDB. Input parameters are specified either as command line options or through a control file. HPU can also be defined as a Control Center plug-in.

For detailed information about the HPU, command line syntax, and control file syntax, please consult *IBM DB2 High Performance Unload for Multiplatforms and Workgroup - User's Guide, SC27-1623*. To locate this document online, use the following URL:

`http://publib.boulder.ibm.com/epubs/pdf/inzu1a13.pdf`

9

Data mart consolidation: A project example

In this chapter we describe how to practically consolidate disintegrated data marts into a DB2 EDW database. We begin the chapter with an introduction to the project environment, including the hardware and software used. We discuss several issues involved in the present scenario where an enterprise has independent data marts. We then discuss the objectives of this sample consolidation exercise and mention in detail how various issues can be resolved by consolidating information from disintegrated marts in to the EDW.

We describe in detail the star schemas of the two independent data marts. Then we describe the existing EDW model and the enhancements made to it in order to accommodate the independent marts.

© Copyright IBM Corp. 2005. All rights reserved.

9.1 Using the data mart consolidation lifecycle

We discussed the data mart consolidation lifecycle in detail in Chapter 6, “Data mart consolidation lifecycle”. The lifecyle is depicted in Figure 9-1 for familiarization. In this chapter we will use the lifecycle concepts to show a sample consolidation exercise.

Figure 9-1 Data mart consolidation lifecycle

However, in this sample exercise project, we are limiting the scope to consolidating two independent data marts. Also, we must make some assumptions about the IT and user environments of the fictitious data warehouse implementation. Because of this, we will not need to meet all the requirements of a more typical and larger scale project. This means that we will not be required to execute all the activities described in the data mart consolidation life cycle.

In this simplistic exercise, we will perform some of the activities from the various phases of the lifecycle. Our intention is simply to give you an example of how the lifecycle can be used. And, to demonstrate how data marts can be consolidated — albeit in a rather simple example. But, the exercise will provide a good beginning template and demonstrate some guidelines to follow as you begin your consolidation projects.

The activities we will perform, and their phases, are listed here:

- **Assessment phase:** During this phase:
  - We assess the two independent data marts that are hosted on Oracle 9i and Microsoft SQL Server 2000.
  - We analyze the existing EDW.
- **Planning phase:** During this phase, we identify the approach we will use to consolidate the independent data marts.
- **Design phase:** During this phase, we design the target schema, the source-to-target mapping matrix, and define the transformation rules.
- **Implementation phase:** During this phase, we develop the ETL and consolidate the data mart data with the EDW.

9.2 Project environment

In this section we describe the environment used for our consolidation exercise, which includes the present architecture of the EDW and two independent data marts. We discuss the issues that exist with these independent data marts that constitute the objectives of the consolidation, and the results achieved upon completion of the project.

We use a number of products during the consolidation exercise, including the following:

- DB2 UDB V8. 2
- DB2 Migration ToolKit V1. 3
- WebSphere Information Integrator V 8. 2
- Oracle 9i
- SQL Server 2000

9.2.1 Overview of the architecture

The present architecture, shown in Figure 9-2, consists of two independent data marts. One is used for the sales data and the other for the inventory data. The sales data mart currently resides on SQL Server 2000, and the inventory data mart resides on Oracle 9i. These two independent data marts existed before the current EDW was implemented. Management now wants to consolidate the two independent data marts into the EDW to reduce costs and integrate the disparate, redundant sources of data to enable them to continue towards their goal of developing a data source that will provide them with a single version of the truth for their decision making.

Archived

Figure 9-2 Project environment

The following describes the EDW and independent data marts.

- EDW on DB2 UDB:

  The Enterprise Data Warehouse (EDW) is hosted on DB2 UDB. The EDW currently hosts information for the following business processes:

  – Order Management
  – Finance

  Management wants to expand the EDW by consolidating several independent data marts into it. As a start, they decide to focus on the sales and inventory data marts.

  To begin the project, we need to understand the data currently in the EDW. The data in the EDW on DB2 is listed in Table 9-1.

Table 9-1 EDW Server information

Parameter	Value
Server	DB2EDW
Operating system	AIX 5. 2L
Database	EDWDB
User/Password	db2/db2
Schema for EDW	db2edw
Schema for staging of SQLServer1 tables	stageora1
Schema for staging of Oracle1 tables	stagesql1

Archived

- Store sales data mart on SQL Server 2000:

  The sales data mart contains data collected from the sales activity in the retail stores. Table 9-2 lists the sales data mart data.

Table 9-2 Store sales data mart information

Parameter	Value
Server	SQLServer1
Operating system	Windows NT Server
Database	StoreSalesDB
User/Password	sales_admin/admin
Schema	StoreSalesDB (same as database)

- Store inventory data mart (Independent) on Oracle 9i.

  The inventory data mart contains information about the inventory levels of the various products in the retail stores. Table 9-3 lists the inventory data mart data.

Table 9-3 Store inventory data mart information

Parameter	Value
Server	OracleServer1
Operating system	Windows NT Server
Database/Schema	StoreInventoryDB (same as schema)
User/Password	inventory_admin/admin
Schema	StoreInventoryDB

Archived

9.2.2 Issues with the present scenario

These are some issues the enterprise faces with the sales and inventory independent data marts, as was shown in Figure 9-2 on page 258:

- There is no data integration or consistency across the sales and inventory data marts. As we can see in Figure 9-3, the data from the two data marts is analyzed independently, and each of them generates their own reports. There is no integration of data at the data mart level, even though, from a business perspective, the two processes (sales and inventory) are tightly coupled and need information exchange so management can predict inventory needs — based on daily sales activity, for example.

Figure 9-3 Reports across independent data marts are disintegrated

- In the present scenario of sales and inventory information existing across two independent data marts, it is not possible to get a single report, such as the one shown in Figure 9-4, that shows sales quantity (from sales data mart) and quantity-in-inventory (from inventory data mart), on the same report.

  Even if we could generate such a report from the independent data marts, the quality would be unacceptable. This is because the data is disintegrated, inconsistently defined, and likely at differing levels of concurrency. That is, the data marts are likely on differing update schedules. Therefore, data from the two cannot be combined for any meaningful result.

Figure 9-4 A report combining sales and inventory data

Archived

In the present implementation, there are added costs for maintaining separate hardware and software environments for the two data marts:

- Additional expenses for training and the skilled resources required to maintain the two environments.
- Additional expenses for development and maintenance of (redundant) data extract, transform, and load (ETL) processes for the two environments
- Additional effort to develop and maintain the redundant data that exists on the two data marts because they are not integrated — in the form of product, store, and supplier data, as examples — as well as a high likelihood that the redundant data will also be inconsistent because of uncoordinated update maintenance cycles
- Lack of a common data model and common data definitions, which will result in inconsistent and inaccurate analyses and reports
- Inconsistent and inaccurate reports due also to the different levels of data concurrency and maintenance update cycles
- Additional resources required manage and maintain the two operating environments
- Implementation of multiple, differing security strategies that can result in data integrity issues as well as security breaches

9.2.3 Configuration objectives and proposed architecture

The primary goals to achieve in this proposed architecture are as follows:

- Integrate data for accurate analysis of sales and inventory, and enable reports with the combined results. For example, the merchandise manager is able to view sales quantity and inventory data in a single report, as shown in Figure 9-5 on page 263 (as well as in Figure 9-4 on page 261). With integrated data, management can now identify stores that are overstocked with specific articles and move some of those articles into stores that are under-stocked, thereby reducing potential markdowns and increasing sales for those articles.

Archived

Figure 9-5 Centralized reporting for sales and inventory businesses through the EDW

For example, management can now:

- Reduce hardware cost and software license costs by consolidating the data marts into a single operating environment on the EDW
- Reduce IT resources required to maintain multiple operating environments on multiple technologies
- Reduce application development and maintenance costs for ETL processing, data management, and data manipulation
- Standardize on a common data model, reducing maintenance costs and improving data quality and integrity
- Coordinate data update cycles to maintain data concurrency and data consistency, improving data and report quality and integrity

Archived

9.2.4 Hardware configuration

The hardware configuration for the EDW and two independent data marts is summarized in the tables that follow:

- **Enterprise data warehouse:** The EDW is hosted on DB2 UDB, running in the AIX 5. 2L operating environment. Table 9-4 lists the specifics of the configuration.

Table 9-4 EDW hardware configuration

Parameter	Value
Server	DB2EDW
Operating system	AIX 5. 2L
Database	EDWDB
Memory	10 GB
Processor	16 CPUs
Disk space	420 GB Disk
Network connection	TCP/IP

- **Store sales data mart:** This is hosted on SQL Server 2000, and running in the Windows NT operating environment. Table 9-5 lists the specifics of the sales data mart configuration.

Table 9-5 Store sales data mart hardware configuration

Parameter	Value
Server	SQLServer1
Operating system	Windows NT Server
Database	StoreSalesDB
Memory	512 MB
Processor	4 CPUs
Disk space	20 GB
Network connection	TCP/IP

Archived

- **Store inventory data mart:** This is hosted on Oracle 9i, and running in the Windows NT operating environment. Table 9-6 lists the specifics of the inventory data mart configuration.

Table 9-6 Store inventory data mart hardware configuration

Parameter	Value
Server	OracleServer1
Operating system	Windows NT Server
Database	StoreInventoryDB
Memory	512 MB
Processor	4 CPUs
Disk space	20 GB
Network connection	TCP/IP

9.2.5 Software configuration

We used a number of software products during this project, as listed in Table 9-7.

Table 9-7 Software used in the sample consolidation project

Software	Description
DB2 UDB V8. 2	DB2 UDB is used as the database for the EDW. Data from sales and inventory data marts is consolidated into the EDW.
DB2 Migration ToolKit V1. 3	Used to migrate the data from Oracle 9i and SQL Server 2000 (except the Store_Sales_Fact table) to DB2 UDB.
WebSphere Information Integrator 8. 2	To copy a huge fact table (Store_Sales_Fact) from the SQL Server 2000 database to the EDW.
SQL Server 2000	To host the independent data mart for the Store Sales data.
Oracle 9i	To host the independent data mart for the Store Inventory data.

Archived

The software was installed and configured in the server environment as depicted in Figure 9-6.

Figure 9-6 Software configuration setup

9.3 Data schemas

In this section we describe the data models used for the two data marts and the EDW. The sales and inventory data marts are independent and exist on separate hardware/software platforms. Both these independent data marts are built using dimensional modeling techniques. The EDW hosts data for the order management and finance business processes.

To begin the consolidation process, we studied in detail the existing data marts from business, technical content, and data quality perspectives.

9.3.1 Star schemas for the data marts

In the sample consolidation project, the two independent data marts were built on star schema data models. They are described as follows:

- **Store sales data mart:** This star schema is shown in Figure 9-7. This is a basic (and incomplete) data model developed solely for purposes of this consolidation exercise project.

Figure 9-7 Star Schema for Sales

Rather than describing the Store Sales data model in text, we decided to summarize the description in a tabular form for easier understanding. That detailed summary descriptive information is contained in Table 9-8.

Archived

Table 9-8 Store sales star schema details

Parameter	Description
Name of data mart	StoreSalesDB: (This data mart is hosted on SQLServer1 machine).
Business process	The business process for which this data mart is designed is the retail store sales. This data mart captures data about the product sales made by various employees to customers in different stores. The data relating to supplier of the product sold is also stored in this data mart. All this data is captured on a daily basis in an individual store. The retail business has several stores. The data captured in this data mart is at the individual line item level as is measured by a scanner device used by the clerk at the store.
Granularity	The grain of store sales star schema is an individual line item on the bill generated after a customer purchases goods at the retail store.
Dimensions	Calendar: The calendar dimension stores date at day level.
	Product: The product dimension stores information about product and the category to which it belongs.
	Supplier: The supplier dimension stores information about the suppliers who supply various products at the stores.
	Customer: This dimension stores information about customers who buy products from stores. For anonymous customers who are not known to the store, this customer table will have a row to identify *unknown* customers.
	Customer_Category: The customer_category table stores information about the segment to which customers belong. Examples of some segments are large, small, medium, and unknown.
	Employee: This table stores information about all employees working for the retail stores at different locations.
	Stores: This table stores information about all stores of the retail business.
	Store_Category: This table stores information about the type of store such as large, small or medium.
Facts	There are 3 facts in the Store_Sales_Fact table: - SalesQty: Quantity of sales for a line item (Additive Fact) - UnitPrice: Unit price for the line item (Non-Additive Fact) - Discount: Discount offered for the line item (Additive Fact assuming discount applies to individual line item per quantity)
Source system	The OLTP retail sales database is the source for this data mart.
Source owner	Retail Sales OLTP Group
Data mart owner	Retail Sales Business Group

Parameter	Description
Reports being generated	The following reports are being generated using the sales data marts:
	Daily sales report by product and supplier
	Weekly sales report
	Monthly sales report by store
	Quarterly sales report
	Yearly sales report by region
Data quality	During the assessment of this data mart, prior to consolidation, the following observations were made in regard to data quality: - Product Dimension: The product information needs to be conformed using the standard definition that exists in the EDW. Some attributes for the product may be needed to be added to the EDW tables if they are not present in the EDW. Also there is need to conform product names to a standardized convention used in the EDW. - Calendar Dimension: The calendar dimension stores data at the daily level. It does not have attributes at the financial and calendar hierarchy level. Using the EDW calendar dimension, the retail business would be able to analyze on certain interesting date attributes which are missing from the present schema. - Supplier Dimension: The supplier dimension needs to be conformed to a central EDW definition. - Surrogate Keys: All dimensions must use surrogate key generation procedures to generate surrogate keys for their dimensions. The EDW already has implemented certain standard guidelines to be followed to generate surrogate keys for various dimensions.

Note: Appendix B, “Data consolidation examples” on page 315, contains a description of the following tables for the sales data mart:

- Products (Dimension)
- Customer (Dimension)
- Customer_Category (Dimension)
- Supplier (Dimension)
- Employee (Dimension)
- Calendar (Dimension)
- Stores (Dimension)
- Store_Category (Dimension)
- Store_Sales_Fact (Fact Table)

Archived

- **Store inventory data mart:** This star schema is shown in Figure 9-8. This is a basic (and incomplete) data model developed solely for purposes of this consolidation exercise project.

Figure 9-8 Star Schema for Inventory

Rather than describing the Inventory data model in text, we decided to summarize the description in a tabular form for easier understanding. That detailed summary descriptive information is contained in Table 9-9.

Table 9-9 Store inventory star schema details

Parameter	Description
Name of data mart	StoreInventoryDB: (This data mart is hosted on OracleServer1 machine]\).
Business process	The business process for which this data mart is designed is the retail store inventory. This data mart captures data about the inventory levels for all products in a given store on a daily basis along with the supplier information for the product. On a daily basis one row is inserted into the fact table for the inventory level of each product in a given store.
Granularity	The grain of this data mart is the quantity_in_inventory (also called quantity-on-hand) per product at the end of the day in a particular store. In addition to this, the data mart also includes the information pertaining to supplier of the product.

Archived

Parameter	Description
Dimensions	Calendar: The calendar dimension stores date at day level.
	Product: The product dimension stores information about product and the category to which it belongs.
	Supplier: The supplier dimension stores information about the suppliers who supply various products at the stores.
	Stores: This table stores information about all stores of the retail business.
Facts	The single fact used in this star schema is: - Quantity_In_Inventory (Semi-Additive)
Source systems	The OLTP retail inventory database is the source for this data mart.
Source owners	Retail OLTP Group
data mart owner	Retail Inventory Business Group
Reports being generated	Weekly inventory by product and supplier.
	Month end inventory for products by supplier and store.
Data quality	During the assessment of this data mart, prior to consolidation, the following observations were made in regard to data quality: - Product Dimension: The product information needs to be conformed using the standard definition that exists in the EDW. some attributes for the product may be needed to be added to the EDW if they are not present in the EDW. Also there is need to conform product names to a standardized convention used in the EDW. - Calendar Dimension: The calendar dimension stores data at the daily level. It does not have attributes at the financial and calendar hierarchy level. Using the EDW calendar dimension, the retail business would be able to analyze on certain interesting date attributes which are missing from the present schema. - Supplier Dimension: The supplier dimension needs to be conformed to a central EDW definition. - Surrogate Keys: All dimensions must use surrogate key generation procedures to generate surrogate keys for their dimensions. The EDW already has implemented certain standard guidelines to be followed to generate surrogate keys for various dimensions.

Archived

Note: Appendix B, “Data consolidation examples” on page 315, contains a description of the following tables for the inventory data mart:

- Products (Dimension)
- Supplier (Dimension)
- Calendar (Dimension)
- Stores (Dimension)
- Store_Inventory_Fact (Fact Table)

9.3.2 EDW data model

The existing EDW data model used for this sample exercise project is shown in Figure 9-9.

Figure 9-9 Existing EDW data model

The existing EDW model consists of both normalized and denormalized tables. We used it as a base to develop the new expanded EDW model, which is designed to consolidate the sales and inventory data marts. The resulting EDW data model is shown in Figure 9-12 on page 282.

Rather than describing the EDW data model in text, we decided to summarize the description in a table form for easier understanding. That detailed summary descriptive information is contained in Table 9-10.

Archived

Table 9-10 EDW schema details

Parameter	Description
Name of EDW	EDWDB (This data warehouse is hosted on DB2EDW machine).
Business Process	The EDW currently contains information for the following two business processes: - Finance which includes accounts and ledgers. - Order Management which includes orders, quotes, shipments and invoicing. In our sample consolidation exercise, we consolidate the sales and inventory data marts into the EDW.
Granularity	The EDW consists of several fact tables with different grains for each of the processes such as orders, shipments, invoicing, quotes, and accounts.
Dimensions and other normalized tables (Total of about 47 tables)	Calendar: The calendar dimension stores date at day level.
	Product: The product dimension stores information about product. This dimension is conformed and is used by several business processes such as orders, shipments, invoice, account, and billing.
	Vendor: The vendor dimension stores information about the vendors who supply various products at the stores. This dimension is conformed and is used by several business processes such as orders, shipments, invoice, account, and billing.
	Concurrency Dimension: This dimension is used to identify the currency type associated with the local-currency facts.
	Customer_Shipping: This dimension stores information about shipping locations for a customer
	Class: This dimension defines description of class to which a vendor belongs.
	Merchant_Group: This defines predefined merchant groups with which the retailer does business.
	And more. . .
Facts	The EDW has several facts and measures relating to the Order management and financial side of the business. Some of the facts are: - Order Amount - Order Quantity - Order Discount - Invoice Amount - Invoice Quantity - Net Dollar Amount - Shipping Charges - Storage Cost - Retail Case Factor - Interest Charged - Interest Paid

Archived

Parameter	Description
Source systems	There are several source systems for the existing EDW. Some of them are: - Order Management OLTP - Shipment OLTP - Financial OLTP
Source owners	The owners of the above source systems are the order management and finance business groups.
EDW owner	EDW Group
Data quality	The data assessment was done for tables such as calendar, product and vendor because the existing data marts to be consolidated would need this information present inside the EDW. The information in the tables calendar, product and vendor tables was found up-to-date and correct.

9.4 The consolidation process

We have described the two independent data marts and the EDW, and detailed the contents of their data models. Now we choose a consolidation approach so we can integrate the data from the multiple sources in our example consolidation project.

9.4.1 Choose the consolidation approach

As discussed in 4.2, “Approaches to consolidation” on page 71, there are three approaches for consolidating independent data marts. Each of these approaches may be used depending on the size of the enterprise, the speed with which you need to deploy the EDW, and cost savings the enterprise may want to achieve.

The three approaches to consolidating the independent data marts are:

- **Simple migration:** We do not use this approach in our example consolidation project. With this approach, all data from the independent data marts now exists on a single hardware platform, but there is still disintegrated and redundant information in the consolidated platform after completion. This is a quicker approach to implement, but it does not provide the integration that we desire in the example.
- **Centralized consolidation:** With this approach, you can elect to redesign the EDW or to start with the primary data mart and merge the others with it. We elected to use the centralized consolidation with redesign in our example consolidation project. However, since we are consolidating two independent data marts into an existing EDW, we will need to use/enhance certain dimensions of the EDW, such as product and vendor.

- **Distributed consolidation:** With this approach, the data in the various independent data marts is consolidated without *physically* integrating the data marts. This is done by restructuring the dimensions in each data mart so that they *conform* with each other. We did not use this approach in our example because we wanted to demonstrate integration of the data.

9.4.2 Assess independent data marts

We need to assess the sales and inventory data marts on the following parameters:

- Business processes used
- Granularity
- Dimensions used
- Facts used (for example, we need to understand whether facts are additive, semi-additive, non-additive, or pseudo in nature)
- Source systems used
- Source system owner
- data mart owner
- Reports currently being generated
- Data quality

We assessed the sales and inventory data marts based on above parameters in 9.3, “Data schemas” on page 266.

After the assessment process, based on parameters stated above, we identify the common and redundant information between the two data marts. In order to do this, we list all dimensions of the two data marts separately in horizontal and vertical fashion as shown in Figure 9-10. Then we identify the data elements that have the same meaning from an enterprise perspective.

It may be that two tables have the same content, but different names. For example, one might have a supplier table and the other a vendor table. But they contain the same, or similar, data. It may also be that a product table is present in both data marts, but the information, in terms of the number of columns, is different. To help us understand these issues, we created the matrix shown in Figure 9-10, to help us to identify common elements of data on the two data marts. The common data elements (dimensions and facts) would then be compared with the EDW existing structure to determine what elements need to be conformed.

Archived

	Inventory Data Mart			
Sales Data Mart	**Product**	**Store**	**Suppliers**	**Calendar**
Product	X			
Customer				
Customer_Category				
Store		X		
Store_Category		X		
Calendar				X
Supplier			X	
Employee				

Figure 9-10 Identifying common elements

Using the information we have gained, we create Table 9-11 with the *common* and *uncommon* data elements from the sales and inventory data marts.

Table 9-11 Common and uncommon data elements

Common data elements (Level of granularity could differ)	**Uncommon data elements**
Product	Customer (in sales data mart only)
Stores	Employee (in sales data mart only)
Store Category	
Supplier	
Calendar	

To conform the data elements, we use the following procedure:

- **For consolidating the data marts into the existing EDW:** In this case, we look for any already conformed standard source of information available before we design a new dimension. In the sample consolidation project, the information pertaining to calendar, product and vendor is already present in the EDW.

Archived

Our next step then is to assess the calendar, product and vendor dimension tables to identify whether these existing tables have enough information (columns) to answer the queries relating to sales and inventory business processes. If data elements are missing, we add them to the EDW dimension tables.

- **For consolidating data marts into a new EDW:** In this case, there would be no existing source of data in the EDW. So, we design new conformed dimensions for *common data elements as* shown in Table 9-11. These new conformed dimensions should include attributes from both the sales and inventory data marts so that both business processes are able to get answers to their queries from a common conformed dimension.

  For *uncommon* data elements, such as customer and employee shown in Table 9-11, we would need to create new dimensions in the EDW.

9.4.3 Understand the data mart metadata definitions

The metadata of the data marts gives users and administrators easy access to standard definitions and rules which can enable them to better understand the data they need to analyze. The problem is that each independent data mart is often implemented with its own metadata repository. So, each independent data mart often has its own definition of commonly used terms within the enterprise such as sales, revenue, profit, loss, and margin. Even definition of very common terms such as product and customer may be different. It is important to understand these metadata differences in each independent data mart.

We analyze the following metadata for the sales and inventory data marts in our sample exercise:

- **Business metadata:** This includes business definition of common terms used in the enterprise. It also includes a business description of each report being used by each independent data mart.
- **Technical metadata:** This includes the technical aspects of data, such as table columns, data types, lengths, and lineage. It helps us understand the present structure and relationship of entities within the enterprise in the context of a particular independent data mart. Each of the EDW tables contain some metadata columns. This is shown in Appendix B, “Data consolidation examples” on page 315. Some of the columns are:
  - METADATA_CREATE_DATE
  - METADATA_UPDATE_DATE
  - METADATA_CREATE_BY
  - METADATA_UPDATE_BY
  - METADATA_EFFECTIVE_START_DATE
  - METADATA_EFFECTIVE_END_DATE

- **ETL metadata:** This includes data generated as a result of the ETL processes used to populate the independent data marts. ETL metadata includes data such as such number of rows loaded, number rejected, errors during execution, and time taken. This information helps use understand the quality of data present in the independent data marts.

9.4.4 Study existing EDW

The existing EDW covers the following business processes:

- Finance
- Order Management

We assessed the EDW in detail in 9.3.2, “EDW data model” on page 272.

Based on the assessment of the EDW and the independent data marts (see section 9.3.1, “Star schemas for the data marts” on page 266), we can construct a matrix, as shown in Figure 9-11.

	Sales Data Mart							Inventory Data Mart				
Existing EDW Tables	Product	Stores	Stores_Category	Customer	Customer_Category	Employee	Calendar	Supplier	Calendar	Product	Stores	Supplier
Product	X									X		
Vendor								X				X
Calendar							X		X			
Currency												
Customer_Shipping												
Merchant												
Merchant_Group												
Class												
And More ⋮												

Figure 9-11 Identifying common data elements

Archived

From Figure 9-11, we deduce that we can take into account the following existing tables of the EDW for consolidating the independent data marts:

- **Product:** The product table in the EDW is being used by business processes such as finance and order management. Upon detailed study of the attributes of the product table, it is found that this table is able to answer queries for both sales and inventory business processes with its present structure. Also the quality of data in the product table present in the EDW is good.

  Conclusion: The product table of the EDW can be used without any change.

- **Vendor:** The vendor table stores the information about suppliers of products. The independent data marts store the same information inside their respective tables named supplier. Upon detailed study of the attributes of the vendor table (EDW), it is found that this table is able to answer queries for both sales and inventory business processes with its present structure. Also the quality of data in the vendor table present in the EDW is good.

  Conclusion: The vendor table of the EDW can be used without any change.

- **Calendar:** The data stored in the calendar table of EDW is at daily level. This table has hierarchies for calendar and fiscal analysis. The table also has attributes for analyzing information based on weekdays, weekends, holidays and major events such as presidents day, super bowl, or labor day. These attributes help in analyzing enterprise performance across holidays, seasons, weekdays, weekends, fiscal and calendar hierarchies. The ability to perform such an analysis with the sales and inventory data marts, is currently not possible.

  Conclusion: The calendar table of the EDW can be used without any change.

Note: The product, vendor, and calendar information is present in the EDW. We analyze these tables in detail to see if they have enough information to answer question relating to the sales and inventory business processes. If these tables have all information to satisfy the needs of the sales and inventory business processes, then we use these tables.

If these tables of the EDW do not have some information, then they would have to be changed to accommodate more columns for the needs of sales/inventory business processes.

9.4.5 Set up the environment needed for consolidation

In 9.2, “Project environment” on page 257, we discussed in detail the environment setup for our sample consolidation project. This included the following topics:

- Overview of present test scenario architecture
- Issues with the present scenario
- Configuration objectives and proposed architecture
- Hardware configuration
- Software configuration

We set up the hardware and software for our sample consolidation project based on the above information.

9.4.6 Identify dimensions and facts to conform

A conformed dimension is one that means the same thing to each fact table to which it can be joined. In the following paragraphs we provide a more precise definition of a conformed dimension.

Two dimensions are said to be conformed if they share one, more, or all attributes that are drawn from the same domain. In other words, one dimension may be conformed even if contains a subset of attributes from the primary dimension.

For example, in our sample consolidation project, we observe that the calendar dimension in the sales data mart conforms to the calendar dimension in the EDW because the number of attributes and values in the calendar table of the sales data mart is a subset of calendar table in the EDW. As can be seen in the EDW Calendar table shown in Appendix B, “Data consolidation examples” on page 315, there are more columns present in this table than in the calendar table in the sales data mart.

Fact conformation means that if two facts exist in two separate locations in the EDW, then they must be the same to be called the same. As an example, revenue and profit are facts that must be conformed. However, in our sample consolidation project, we do not find any facts that need to be conformed.

Archived

After assessment of the independent data marts (9.4.2, “Assess independent data marts” on page 275) and the EDW (9.4.4, “Study existing EDW” on page 278), the dimensions that need to be conformed are summarized in Table 9-12.

Table 9-12 Dimensions in data marts that need to conform to the EDW

Dimensions in data marts	Corresponding EDW dimension
Calendar (Sales data mart)	Calendar
Calendar (Inventory data mart)	Calendar
Product (Sales data mart)	Product
Product (Inventory data mart)	Product
Supplier (Sales data mart)	Vendor
Supplier (Inventory data mart)	Vendor

Note: No additions or modifications are required for the EDW dimensions, because they all have the attributes needed to answer sales and inventory related business questions.

After assessing the requirement for conforming facts, we found, as shown in Table 9-13, that there are no facts to be conformed.

Table 9-13 Facts common between EDW and independent data marts

Facts in data marts	Corresponding EDW facts
SalesQty (Sales data mart)	None
UnitPrice (Sales data mart)	None
Discount (Sales data mart)	None
Quantity_In_Inventory (Inventory data mart)	None

Archived

9.4.7 Design target EDW schema

The target EDW schema designed to consolidate the two independent data marts is shown in Figure 9-12.

Figure 9-12 EDW Schema designed for consolidation

The schema designed in Figure 9-12 uses the following existing dimensions of the EDW:

- Product
- Calendar
- Vendor

The new tables added to the EDW schema are:

- Stores
- Customer
- Employee
- Store_Sales_Fact
- Store_Inventory_Fact

The EDW schema tables are explained in detail in Appendix B, “Data consolidation examples” on page 315.

9.4.8 Perform source/target mapping

The source to target data map details the specific fields where data is to be extracted and transformed, to populate the target database columns of the EDW schema. The source to target data map includes the following items:

- Target EDW table name
- Target EDW column name
- Target EDW data type
- Data mart involved in consolidation
- Table name of source data mart
- Column name of source data mart
- Data type of source data mart
- Transformation rules involved

The source to target data mapping for the sales and inventory data marts is shown in the Appendix C, “Data mapping matrix and code for EDW” on page 365.

9.4.9 ETL design to load the EDW from data marts

The entire ETL process of consolidating data marts into an EDW is broadly divided into two steps, as shown in Figure 9-13.

Archived

Figure 9-13 Consolidating the two independent data marts

- In Step 1, the ETL process is designed to transfer data from the two data marts (sales and inventory) into the EDW.
- In Step 2, the ETL process is designed to feed the EDW directly from the sources for the sales and inventory data marts. As shown in Figure 9-13 on page 284, the sales and inventory data marts can be eliminated after this step.

Note: In the sample consolidation project, we only describe the ETL for the consolidation depicted in Step 1 of Figure 9-13.

The ETL design to consolidate data from sales and inventory data marts into the EDW is broadly divided into the following two phases:

- Source to staging process
- Staging to publish process

Source to staging process

In this phase, we migrate data from the sales and inventory data marts into the EDW staging area as shown in Figure 9-14.

Archived

Figure 9-14 Source to staging process

The data is migrated into staging areas stagesql1 and stageora1 using the following tools:

- Migration ToolKit V1. 3 (MTK)
- WebSphere Information Integrator V8. 2 (WebSphere II)

Table 9-14 explains in detail the objects extracted and populated into the staging areas and the particular tool used.

Table 9-14 Objects transferred from source to staging area

Data mart name	Object extracted	Software used	Staging area
Sales	Employee	MTK	stagesql1
Sales	Calendar	MTK	stagesql1
Sale	Product	MTK	stagesql1
Sales	Stores	MTK	stagesql1
Sales	Store_Category	MTK	stagesql1
Sales	Customer	MTK	stagesql1
Sale	Customer_Category	MTK	stagesql1
Sales	Supplier	MTK	stagesql1
Sales	Store_Sales_Fact	WebSphere II	stagesql1
Inventory	Calendar	MTK	stageora1
Inventory	Stores	MTK	stageora1

Archived

Data mart name	Object extracted	Software used	Staging area
Inventory	Supplier	MTK	stageora1
Inventory	Product	MTK	stageora1
Inventory	Store_Inventory_Fact	MTK	stageora1

Note: As shown in Table 9-14, we use WebSphere Information Integrator 8. 2 software only for referring to a single fact large table in the sales data mart.

Some of the important activities done in the staging area are as follows:

- The data types of the source data elements must be converted to match the data types of the target columns in the staging area of the EDW. The MTK and WebSphere II do the conversion of the data types. Also it is important that data lengths of the target columns must be adequate to allow the source data elements to be moved, expanded, or truncated.
- We analyze the data to validate against business data domain rules such as:
  - One customer having several primary keys or unique IDs. This is a major problem faced when consolidating independent data marts. As an example, the same customer “Cust-1” may have separate primary keys in independent data marts. The same customer may also have several names or addresses represented wrongly in several independent data marts. Such problem are solved by cleansing data and using surrogate keys. One such example, using products, is shown in Table 9-16.
  - Data elements should not have un-handled NULL values in columns for columns that logically cannot contain NULLs. NULL values cause loss of data when two or more tables are joined based on a column that has NULL values. NULL values should generally be represented as “N/A” or “Do Not Know”.
  - Data elements that can have NULL values should be identified.
  - Data elements should not have decoded columns which represent some meaning such as “PX88V121234” where “PX88” means chocolate products and “V1” means from “Nuts only”. That is, there should be no embedded logic inside and codes. All descriptions of the code belong as a column inside the dimension table and not inside a cryptic element.
  - Data elements which have date present in them should be handled carefully. As an example, a date such as “19-MAR-2005” could be stored separately in several data marts as “March19, 2005”, ”03-19-2005”, ”19-03-2005”, ”20050319”, or ”20051903”. Also it could be that the date is stored in a textual field column instead of the normal date field column.

- Data should be consistently represented. For example, the US city of San Jose should be expressed in data as SJ, SJE or SJSE. But only one naming convention should be used for all such data.
- Data elements (columns) should not be concatenated around free-form text fields. For example address line1, address line2, address line3, and so on. The correct representation must be to break each element and to store it into a separate column in the dimension table.
- Data rows should not be duplicated. Basically what this means is that column sets that should be unique should be identified.
- Data domain values should not be arbitrary, such as age column of customer age or employee age, having a value 888. Another example could be a “date of first purchase” being earlier than the customer’s “date of birth”.
- Data elements that hold numeric values should contain only acceptable ranges of numeric fields.
- Data elements that hold character values should contain only acceptable ranges of character fields.
- A data domain should not contain intelligent default values. For example, a social security number 66-66-66 might indicate that it represents a person with “Illegal immigrant” status.
- Data elements should be determined that can explicitly only contain a set of values. Such set of values should also be determined.
- Within a given data domain, a single code value should not be used to represent multiple entities. For example, code “1” and “2” to represent a customer, ”3” and “4” to represent a product.

► Data cleansing and quality checks:

This involves cleaning incorrect attribute vales, name and address parsing, missing decodes, incorrect data, missing data and inconsistent values of data. In our sample consolidation project, we faced a problem with incorrect data, as shown in Table 9-15.

Table 9-15 Customer table sample data

CustomerKey (Surrogate)	CustomerID_ Natural	Enterprise Name	Address	City
1	1792-ZS	Cottonwood	13 Brwn Str	San Jose
2	1792-ZS	Cottonwool	13 Bwnr Str	San Jose
3	1792-ZS	Cottonwoode	13 Brwn Str	San Jose
4	1792-ZS	Cottonwod	13 Bwnr Str	San Jose

The correct customer name is “Cottonwood”. All the rows shown above actually belong to the same customer “Cottonwood” but appear as three different customers to someone who sends a mailing list. The outcome is that when the enterprise sends some mail to all its customers, the “Cottonwood” office gets the same mailing four times.

- Data transformations:

  The data is transformed based on the source to target data mapping table shown in Appendix C, “Data mapping matrix and code for EDW” on page 365.

- Manage surrogate key assignments and lookup table creation:

  All conformed dimensions within the data warehouse use a dimension mapping process to derive surrogate keys and enforce dimensional conformity. The benefit of this approach is the added ability of the data warehouse to handle multiple source systems as well as multiple independent data marts without causing physical changes to the dimension table.

  The conformed dimension mapping example in Table 9-16 shows three Product Name values as they have been extracted for the first time, from the Sales data mart into the data warehouse, in the data mart (sales) Product Table. There are also three Product Name values that have been extracted from the Inventory data mart out of which one Product Name is the same as extracted by the sales data mart.

Table 9-16 Sales and Inventory data mart Product table values

Primary Key (SS_Key or data mart_Key)	SS_ID or data mart_ID	Product Name
66	Sales	Chocolate-Brand-1
67	Sales	Chocolate-Brand-2
68	Sales	Chocolate-Brand-3
1000	Inventory	Chocolate-Brand-66
1001	Inventory	Chocolate-Brand-1
1002	Inventory	Chocolate-Brand-99

The process populating the data warehouse dimension mapping table (Table 9-17) and corresponding enterprise data warehouse product dimension table (Table 9-18), has extracted the unique primary key from the data mart and generated surrogate keys (EDW_Key) in the data warehouse. As we go downstream, the fact table ETL processes will use the dimension mapping table for looking up appropriate surrogate keys, as they are relevant to each row of the fact table.

*SS_Key or data mart_Key: Source system primary key or primary key of the customer data mart dimension.

*SS_ID or data mart_ID: Name or ID given to the source system or data mart as a whole.

Table 9-16 shows a very common problem faced when consolidating several independent data marts into the EDW. It is observed that product “Chocolate-Brand-1” has different primary keys in the sales and inventory data marts.

Table 9-17 Data warehouse dimension mapping table

SS_ID or data mart_ID	Entity_Name	SS_Key or data mart_Key	EDW_Key
Sales	Product (EDW)	66	1
Sales	Product (EDW)	67	2
Sales	Product (EDW)	68	3
Inventory	Product (EDW)	1000	4
Inventory	Product (EDW)	1001	1
Inventory	Product (EDW)	1002	5

*SS_ID or data mart_ID: Name or ID given to the data mart as a whole.

*Entity_Name: Name of the dimension table in the EDW.

*SS_Key or data mart_Key: Source system primary key or data mart primary key.

The following logic is employed when processing each product into the data warehouse within inventory data mart:

– If the unique natural key of a product from the inventory data mart is equal to the unique natural key of a product from the sales data mart, then the already assigned surrogate key is applied when inserting the new row into the mapping table. This is shown for product named “Chocolate-Brand-1“.

– If the unique natural key of a product from inventory data mart is not equal to the unique natural key of any product from the sales data mart, then a new surrogate key is generated and that key is applied when inserting new records into the dimension table and the dimension mapping table.

Archived

Table 9-18 Product_DW dimension of the EDW

EDW_Key	Product Name
1	Chocolate-Brand-1
2	Chocolate-Brand-2
3	Chocolate-Brand-3
4	Chocolate-Brand-66
5	Chocolate-Brand-99

Staging to publish process

In this process we load the data from staging area to the EDW schema shown in Figure 9-15. We use the source to target data map created in section 9.4.8. We load each dimensions primary key with a surrogate key.

Figure 9-15 Staging to publish process

The ETL code involves the following functions:

- Dimension table loading
- Fact table loading

The ETL code to load the EDW from the staging area is described in Appendix C, “Data mapping matrix and code for EDW” on page 365.

Note: The ETL code referenced is only a sample. Providing the code for a complete ETL process is outside the scope of this book. At this date, ETL coding is a well understood process, and tools, such as IBM WebSphere DataStage, are available to provide that service.

9.4.10 Metadata standardization and management

Metadata is very important, and needs to be standardized across the enterprise. To do so would require creation of a standardized common metadata repository which includes all of the applications, data, processes, hardware, software, technical metadata, and business knowledge (business metadata) possessed by an enterprise.

Metadata management includes the following aspects:

- **Business metadata:** Provides a roadmap for users to access the data warehouse. It hides technological constraints by mapping business language to the technical systems. Business metadata includes:
  - Glossary of terms
  - Terms and definitions for tables and columns.
  - Definition of all reports
  - Definition of data in the data warehouse
- **Technical metadata:** Includes the technical aspects of data such as table columns, data types, lengths, and lineage. Some examples include:
  - Physical table and column names
  - Data mapping and transformation logic
  - Source system details
  - Foreign keys and indexes
  - Security
  - lineage analysis: Helps track data from a report back to the source, including any transformations involved.
- **ETL execution metadata:** Includes the data produced as a result of ETL processes, such as number of rows loaded, rejected, errors during execution, and time taken. Some of the columns that can be used as ETL process metadata are:
  - Create Data: Date the row was created in the data warehouse.
  - Update Date: Date the row was updated in the data warehouse.
  - Create By: User name that created the record.
  - Update By: User name that updated the record.

Archived

- Active in Operational system flag: Used to indicate whether the production keys of the dimensional record are still active in the operational source.
- Confidence level indicator: Helps user identify potential problems in the operational source system data.
- Current Flag indicator: Flag used to identify the latest version of a row.
- OLTP System Identifier: Used to track origination source of a data row in the data warehouse for auditing and maintenance purposes.

Table 9-19 shows some sample metadata columns in a dimension table.

Table 9-19 Employee table with sample metadata columns

EmployeeID (Surrogate)	EmployeeID OLTP Key	Employee Name	City	Current Flag Indicator	OLTP System Identifier
1	RD-18998	John Smith	San Jose	Y	1
2	RD-18999	Mark Waugh	New York	N	2
3	RD-18999	Mark Waugh	San Diego	Y	2
4	RD-18675	Sachin Tendulkar	Bombay	Y	3
5	RD-12212	Tom Williams	Dayton	Y	3

As shown in Table 9-19, the employees data is populated by OLTP systems as shown in Table 9-20. The OLTP System identifier helps in tracking the data from the EDW back to the original OLTP system. Also a “Current Flag Indicator” shows the most current record in the EDW along with all previous old records.

Table 9-20 describes the various OLTP source systems that feed data to the EDW.

Table 9-20 Operational System Identifier metadata table

OLTP System Identifier	Description of the Source System
1	Store Sales-North Region
2	Store Sales-South Region
3	Store Sales-West Region
4	Store Sales-East Region

Archived

9.4.11 Consolidating the reporting environment

Typically when consolidating independent data marts into an EDW, it is also a good practice to identify the reporting tools or reporting environments that are being used in the enterprise to query the independent data marts. The reporting tools may be client-server, Web-based, or a mix of the two.

Figure 9-16 shows that each independent data mart generally has its own reporting environment. This means that each data mart has its own report server, security, templates, metadata, backup procedure, print server, development tools and other costs associated with the reporting environment.

Figure 9-16 Reporting environment of independent data marts

Some of the disadvantages of have diverse reporting tools and reporting environments within the same enterprise are:

- High cost in IT infrastructure both in terms of software and hardware needed to support diverse reporting needs. Multiple Web servers are used in the enterprise to support reporting needs of each independent data mart.
- No common reporting standards. Without any common standards, it is difficult to analyze information effectively.
- Several duplicate and competing reporting systems present.
- Multiple backup strategies for the various reporting systems.
- Multiple repositories for each reporting tool.

Archived

- No common strategy for security for data access: In scenarios where there are multiple reporting tools to query independent data marts, there is no common enterprise wide security strategy for data access. Each reporting tool builds its own security domain to secure the data of its data mart. Such multiple security strategies often jeopardize the quality and reliability of organizational information.
- High cost of training for the multiple business users on the various reporting tools.
- High cost of training developers in learning diverse reporting solutions.
- Cost of development of each report is high in case of diverse reporting tools.

The advantages of standardizing the reporting environment are:

- Reduced cost of IT infrastructure in terms of software and hardware.
- Reduced cost of report development.
- Single and integrated security strategy for the entire enterprise.
- Single reporting repository for the single reporting solution.
- Elimination of duplicate, competing report systems.
- Reduced training costs of developers in learning a single reporting solution in comparison to multiple tools.
- Reduced training cost for business users in learning various reporting tools.
- A common standardization in reports is introduced to achieve consistency across enterprise.
- Reduced number of defects in comparison to multiple reporting tools accessing multiple independent data marts.

9.4.12 Testing the populated EDW data with reports

In order to test the consolidation process, we analyze reports from the consolidated and non-consolidated data mart environments. This is to validate that we can still get the same reports after consolidation as with the independent data marts. It is also to demonstrate that users can create new and expanded reports by having access to additional data sources (in the EDW) and by the data quality and data consistency work that was performed during consolidation.

Archived

Independent data mart reports

Figure 9-17 shows the reports we developed from the independent data marts for a product code “PX391-BR”. The sales data mart shows the $(Revenue), whereas the inventory data mart shows the inventory on hand.

We look at some sample data in those reports and observe that:

- There is no data integration, these are independent data marts. They are managed separately and maintained separately. Therefore, there is no consistency checking between them.
- The Product Name for Product Code “PX391-BR” is spelled incorrectly in the sales and inventory data marts. In the sales data mart it is called “Bread-Weat”, whereas in the inventory data mart it is called “Bread-Wheat”. This is a result of the lack of consistency checking.
- It is not clear from the reports whether or not there is a difference in the metadata definitions. For example, we cannot tell whether the definition of inventory and definition of inventory on hand are the same because there is no sharing of the data between the two organizations.

Figure 9-17 Testing - individual reports from the independent data marts

Individual Sales and Inventory reports from the EDW

In this section we discuss the reporting capabilities from the two individual organizations after the data marts have been consolidated into the EDW.

First we show that each organization can still create the same reports after consolidation as they had prior to consolidation. This is to demonstrate to each organization that the consolidation was successful. You can see, in Figure 9-18, that the reports agree with those in Figure 9-17.

Then, in addition, some enhancements were achieved even in this first phase. Those enhancements are to the data quality and consistency. For example, there is a change in a Product Name. That is, in both reports the Product Name for Product Code PX391-BR is now the same. It is Bread-Wheat (not Bread-Weat as prior to consolidation).

Date	Product Code	Product Name	Revenue($)
01/01/05	PX391-BR	Bread-Wheat	1000
01/01/05	PX392-BR	Bread-Maize	1000
01/03/05	PX393-BR	Meat-Fish	796
01/04/05	PX394-BR	Meat-Chicken	8980

Date	Product Code	Product Name	Inventory on Hand
01/01/05	PX391-BR	Bread-Wheat	68890
01/01/05	PX392-BR	Bread-Maize	68890
01/03/05	PX398-BR	Meat-Fish	9252213
01/04/05	PX399-BR	Meat-Chicken	5421542

Figure 9-18 Testing - validating the consolidation

Now we can proceed to the next phase and demonstrate how new reports can be generated because additional data is now available from the EDW. The reports are still individual reports by organization, but with additional information.

Figure 9-19 shows the individual business reports we developed for product code “PX391-BR”. Note, however, that the sales report still shows the $(Revenue) and the inventory report still shows the inventory on hand, but we have added new information to the report.

Figure 9-19 Individual sales and inventory reports from EDW.

Integrated Sales and Inventory reports from the EDW

In this next phase we have integrated the data marts into the EDW. That is, the dimensions and facts are conformed, the metadata is consistent, and the ETL processing is coordinated so we have the same level of concurrence in the data. That is, the data for both sales and inventory have been updated during the same cycle, on the same date.

Now we can have one integrated report to satisfy enterprise management, rather than reports that can only satisfy the individual organizations.

In Figure 9-20 the report shows both sales and inventory data. This can enable management to perform significantly better decision making because they now have integrated information. For example, this simple report shows sales quantity in addition to quantity on hand - and it is accurate! Now management can perform such analyses as:

- Sales by store and sales by region
- Sales by product and by time and by season
- Sales by supplier and by discount
- Better plan product deliveries to the stores based on accurate inventory
- Better plan production levels and resource usage

Date	Product	Sales Quantity	$ Revenue	Quantity on Hand
01/10/05	P1	100	400	1000
02/11/05	P1	98	392	868
03/12/05	P1	10	40	796
04/13/05	P1	10	40	786

Figure 9-20 Integrated sales and inventory report from EDW

In addition to these types of reports, having the consolidated information adds significantly to their business intelligence capabilities. Now they can start to really manage the business. For example, based on sales trends and their ability to better manage inventory and deliveries management can focus on their key performance indicators. By proactively managing the business, they will be better able to meet their business goals and objectives. This fits right in with the current industry focus on business performance management.

With performance objectives and shorter measurement cycles, management will better be able to deliver to their stakeholders. It is all part of the goal of managing costs, meeting sales objectives, and beating the competition. It is another milestone.

9.5 Reaping the benefits of consolidation

There are numerous benefits to be realized from consolidation of data marts into the EDW. These benefits are not only tangible, but also intangible ones. Many of these benefits are difficult to quantify in terms of monetary value, but in reality do provide significant competitive advantage. In order to understand the intangible benefits, the enterprise must compare the decisions that cannot be made with independent data marts with the decisions that can be made with the EDW.

Below we list some of the benefits that enterprise gains from our simple consolidation exercise:

- Making integrated data available for analysis:

  The consolidation effort in our simple example helps us show the benefit of integrating the sales and inventory business. For instance, we can extract a sales report from our EDW which shows that a $2500 lawn tractor will sell, on average, twenty per week. Using the inventory data in our EDW, we can identify that a slow-selling, high-priced article such as a lawn tractor may likely have an issue of running out of stock, as stores generally stock only twenty items.

  By integrating the sales and inventory data we are able to more quickly identify when items sell and their corresponding inventory levels in stores. Having the integrated sales and inventory data in the EDW helps us to increase the stock of such items thereby reducing lost sales due to being out of stock. This enables timely order placement which helps maintain that safety stock. In other words it simply helps the enterprise with achieving profits.

- Non-conformed dimensions associated with independent data marts:

  In our sample consolidation project, we observed that the product dimension was non-conformed between the two data marts. The result was that in many cases the same product was represented under misspelled names in different marts. This lead the inventory to order the same product twice several times on a regular basis which caused an imbalance between the sales and inventory movement for this product leading to excessive overstocking. The overstocking led to huge carrying costs for the inventory.

  Also, in addition to this the management spent huge amounts of time tracking the problem to incompatible data coming from the independent data marts.

  Such problems are eliminated with the integration of information in the EDW.

- Standardization of metadata definitions from a business standpoint:

  In our sample consolidation exercise we were able to standardize metadata definitions for the sales and inventory business processes.

- Elimination of redundant data:

  We were able to remove redundant data for product, vendor and dates related information which was duplicated and inconsistently defined in the sales and inventory data marts.

Archived

- Cost savings:

  These are some cost savings that came from consolidating independent data marts into the EDW:

  - Reduction in hardware and software costs
  - Reduction in software licenses
  - Reduction in long term technical training costs associated with maintaining diverse hardware/software platforms
  - Reduction in long term end user training costs that is associated with training users to use the diverse independent data marts and also in most cases diverse reporting tools
  - Reduction in number of third party software licences associated with any add-ons associated with the independent data marts
  - Elimination of on-going maintenance and support fees for independent data marts.
  - Elimination of on-going system administration costs associated with maintaining multiple data marts.
  - Space occupied by several independent data marts can be saved and used by enterprise for other purposes.
  - Security costs involved in storing multiple servers in several places can be reduced by consolidation into the EDW.
  - Expenditures for evaluating and selecting several data mart software can also be reduced.
  - Elimination of the operating cost for the independent data marts

Archived

A

Consolidation project example: Table descriptions

In this appendix we provide a description of the tables used in the independent data marts on Oracle 9i and SQL Server 2000, and in our enterprise data warehouse on DB2. In addition, there are examples of the DDL statements used to create those tables.

We started with an EDW schema built on DB2 UDB, and data mart schemas built on Oracle ad SQL Server. The objective was then to modify the EDW schema to accept the data from the Oracle and SQL Server data marts. The tables that comprise those data schemas are described in the remainder of this appendix.

© Copyright IBM Corp. 2005. All rights reserved.

Data schemas on the EDW

In this section we cover the tables contained on the EDW. Table A-1 describes the contents of the calendar table on the EDW.

Table A-1 EDW.CALENDAR table

Column name	Data type	Description
C_DATEID_SURROGATE	INTEGER	SURROGATE KEY
C_DATE	DATE	DATE IN MM-DD-YYYY FORMAT
C_YEAR	SMALLINT	YEAR
C_QUARTER	CHAR(50)	QUARTER OF MONTH AS Q1,Q2,Q3,Q4
C_MONTH	VARCHAR(100)	MONTH NAME
C_DAY	SMALLINT	DAY
CALENDAR_DATE	DATE	CALENDAR DATE
CALENDAR_DAY	CHAR(10)	CALENDAR DAY
CALENDAR_WEEK	CHAR(10)	CALENDAR WEEK
CALENDAR_MONTH	CHAR(10)	CALENDAR MONTH
CALENDAR_QUARTER	CHAR(10)	CALENDAR QUARTER
CALENDAR_YEAR	CHAR(10)	CALENDAR YEAR
FISCAL_DATE	DATE	FISCAL DATE (SUCH AS DATE STARTING FROM MARCH 01 IN SOME COUNTRIES)
FISCAL_DAY	CHAR(10)	FISCAL DAY
FISCAL_WEEK	CHAR(10)	FISCAL WEEK
FISCAL_MONTH	CHAR(10)	FISCAL MONTH
FISCAL_QUARTER	CHAR(10)	FISCAL QUARTER
FISCAL_YEAR	CHAR(10)	FISCAL YEAR
SEASON_NAME	CHAR(10)	NAME OF SEASON
HOLIDAY_INDICATOR	CHAR(10)	Y/N FOR WHETHER HOLIDAY OR NOT
WEEKDAY_INDICATOR	CHAR(10)	Y/N FOR WHETHER WEEKDAY OR NOT
WEEKEND_INDICATOR	CHAR(10)	Y/N FOR WHETHER WEEKEND OR NOT
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)

Archived

Column name	Data type	Description
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTICE_ START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTICE_ END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Table A-2 describes the contents of the product table on the EDW.

Table A-2 EDW.PRODUCT table

Column name	Data type	Description
PRODUCTKEY	INTEGER	SURROGATE KEY
PRODUCTID_NATURAL	VARCHAR(100)	NATURAL ID FOR THE PRODUCT
PRODUCTNAME	VARCHAR(100)	NAME OF PRODUCT
CATERGORYNAME	VARCHAR(100)	CATEGORY NAME TO WHICH PRODUCT BELONGS
CATEGORYDESC	VARCHAR(400)	CATEGROY DESCRIPTION TO WHICH PRODUCT BELONGS
P_ITEM_STATUS	CHAR(10)	ITEM STATUS OF PRODUCT
P_POS_DES	CHAR(10)	POINT OF SALES DESCRIPTION OF PRODUCT
P_ORDER_STAT_FLAG	CHAR(10)	ORDER STATUS FLAG OF PRODUCT
P_HAZARD_CODE	CHAR(10)	HAZARDOUS CODE OF PRODUCT
P_HAZARD_STATUS	CHAR(10)	HAZARDOUS STATUS OF PRODUCT
P_TYPE_DIET	CHAR(10)	TYPE OF DIET TO WHICH PRODUCT BELONGS
P_WEIGHT	CHAR(10)	WEIGHT OF PRODUCT
P_WIDTH	CHAR(10)	WIDTH OF PRODUCT (IF ANY)
P_PACKAGE_SIZE	CHAR(10)	PACKING SIZE OF PRODUCT
P_PACKAGE_TYPE	CHAR(10)	PACKAGE TYPE OF PRODUCT
P_STOREAGE_TYPE	CHAR(10)	STORAGE TYPE USED BY PRODUCT
P_PRODUCT_MARKET	CHAR(10)	TYPE OF MARKET SEGMENT TO WHICH PRODUCT BELONGS
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE

Archived

Column name	Data type	Description
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTIVE _START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTIVE_ END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Table A-3 describes the contents of the vendor table on the EDW.

Table A-3 EDW.VENDOR table

Column name	Data type	Description
SUPPLIERKEY	INTEGER	SURROGATE KEY
SUPPLIERID_NATURAL	INTEGER	NATURAL KEY OF THE SUPPLIER
COMPANYNAME	VARCHAR(100)	COMPANY NAME OF SUPPLIER
CONTACTNAME	VARCHAR(100)	CONTACT NAME OF SUPPLIER
CONTACTTITLE	VARCHAR(100)	CONTACT TITLE OF SUPPLIER
ADDRESS	VARCHAR(100)	CONTACT ADDRESS OF SUPPLIER
CITY	VARCHAR(100)	CONTACT CITY OF SUPPLIER
REGION	VARCHAR(100)	CONTACT REGION OF SUPPLIER
POSTALCODE	VARCHAR(100)	CONTACT POSTALCODE OF SUPPLIER
COUNTRY	VARCHAR(100)	CONTACT COUNTRY OF SUPPLIER
PHONE	VARCHAR(100)	CONTACT PHONE OF SUPPLIER
FAX	VARCHAR(100)	CONTACT FAX OF SUPPLIER
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTIVE _START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTIVE _END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Archived

Table A-4 describes the contents of the stores table on the EDW.

Table A-4 EDW.STORES table

Column name	Data type	Description
STOR_ID	INT	SURROGATE KEY
STOR_NAME	CHAR(40)	NATURAL KEY OF THE STORE
STOR_ADDRESS	CHAR(40)	STORE ADDRESS
CITY	CHAR(20)	CITY NAME
STATE	CHAR(2)	STATE TO WHICH STORE BELONGS
ZIP	CHAR(5)	ZIP CODE
STORE_CATEGORY	VARCHAR(100)	CATEGORY
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTIVE _START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTIVE _END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Table A-5 describes the contents of the store inventory fact table on the EDW.

Table A-5 EDW.EDW_INVENTORY_FACT table

Column name	Data type	Description
STORE_ID	INTEGER	SURROGATE KEY OF STORES TABLE
PRODUCT_ID	INTEGER	SURROGATE KEY OF PRODUCT TABLE
DATE_ID	INTEGER	SURROGATE KEY OF CALENDAR TABLE
SUPPLIER_ID	INTEGER	SURROGATE KEY OF SUPPLIER TABLE
QUANTITY_IN_INVENTORY	INTEGER	TOTAL INVENTORY OF THE PRODUCT AT END OF DAY

Archived

Table A-6 describes the contents of the employee table on the EDW.

Table A-6 EDW.EMPLOYEE table

Column name	Data type	Description
EMPLOYEEKEY	INTEGER	SURROGATE KEY
EMPLOYEEID_NATURAL	INTEGER	NATURAL ID FOR THE EMPLOYEE
REPORTS_TO_ID	INTEGER	REPORTING MANAGERS ID(SURROGATE KEY)
FULLNAME	VARCHAR(100)	FULL NAME OF EMPLOYEE
LASTNAME	VARCHAR(100)	LASTNAME OF EMPLOYEE
FIRSTNAME	VARCHAR(100)	FIRSTNAME OF EMPLOYEE
MANAGERNAME	VARCHAR(100)	MANAGER NAME OF EMPLOYEE
DOB	DATE	DATE OF BIRTH
HIREDATE	DATE	HIRING DATE
ADDRESS	VARCHAR(100)	MAILING ADDRESS OF EMPLOYEE
CITY	VARCHAR(80)	CITY
REGION	VARCHAR(80)	REGION
POSTALCODE	VARCHAR(80)	POSTALCODE OF EMPLOYEE
COUNTRY	VARCHAR(90)	COUNTRY OF CITIZENSHIP
HOMEPHONE	VARCHAR(90)	RESIDENCE PHONE
EXTENSION	VARCHAR(90)	OFFICE PHONE AND EXTENSION
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTIVE _START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTIVE _END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Archived

Table A-7describes the contents of the customer table on the EDW.

Table A-7 Customer table

Column name	Data type	Description
CUSTOMERKEY	INTEGER	SURROGATE KEY
CUSTOMERID_NATURAL	VARCHAR(100)	CUSTOMER NATURAL ID
CUSTOMER_CATEGORY	VARCHAR(100)	CATEGORY TO WHICH CUSTOMER BELONGS
COMPANYNAME	VARCHAR(100)	COMPANY NAME OF THE CUSTOMER
CONTACTNAME	VARCHAR(100)	CONTACT NAME OF THE CUSTOMER
ADDRESS	VARCHAR(100)	ADDRESS OF CUSTOMER
CITY	VARCHAR(100)	CITY OF CUSTOMER
REGION	VARCHAR(100)	REGION OF CUSTOMER
POSTALCODE	VARCHAR(100)	POSTALCODE OF CUSTOMER
COUNTRY	VARCHAR(100)	COUNTRY OF CUSTOMER
PHONE	VARCHAR(100)	PHONE OF CUSTOMER
FAX	VARCHAR(100)	FAX OF CUSTOMER
METADATA_CREATE_DATE	DATE	RECORD CREATED DATE
METADATA_UPDATE_DATE	DATE	RECORD UPDATED DATE
METADATA_CREATE_BY	CHAR(10)	RECORD CREATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_UPDATE_BY	CHAR(10)	RECORD UPDATED BY(GENERALLY USER ID FOR DATABASE)
METADATA_EFFECTIVE _START_DATE	DATE	EFFECTIVE START DATE OF THE RECORD
METADATA_EFFECTIVE _END_DATE	DATE	EFFECTIVE END DATE OF THE RECORD

Table A-8 describes the contents of the store sales fact table on the EDW.

Table A-8 EDW.EDW_SALES_FACT table

Column name	Data type	Description
PRODUCTKEY	INTEGER	SURROGATE KEY OF PRODUCT
EMPLOYEEKEY	INTEGER	SURROGATE KEY OF EMPLOYEE
CUSTOMERKEY	INTEGER	SURROGATE KEY OF CUSTOMER
SUPPLIERKEY	INTEGER	SURROGATE KEY OF SUPPLIER

Archived

Column name	Data type	Description
STOREID	INTEGER	SURROGATE KEY OF STORE
DATEID	INTEGER	SURROGATE KEY OF CALENDAR
POSTRANSNO	INTEGER	POINT OF SALES TRANSACTION NUMBER
SALESQTY	INTEGER	SALES QUANTITY
UNITPRICE	DECIMAL(19,4)	UNIT PRICE OF PRODUCT
SALESPRICE	DECIMAL(19,4)	SELLING PRICE OF PRODUCT
DISCOUNT	DECIMAL(19,4)	DISCOUNT OFFERED ON A PRODUCT

Data schemas on the ORACLE data mart

In this section we cover the tables contained in the inventory data mart, built on Oracle 9i. As part of the project, this schema, and the tables defined by it, were consolidated with the EDW schema. *SCOTT* is the user name for this data mart, and is thus part of each table name.

Table A-9 describes the contents of the store inventory fact table on the inventory data mart.

Table A-9 TSCOTT.STORE_INVENTORY_FACT table

Column name	Data type	Description
STORE_ID	NUMBER(10)	SURROGATE KEY OF STORES TABLE
PRODUCT_ID	NUMBER(10)	SURROGATE KEY OF PRODUCT TABLE
DATE_ID	NUMBER(10)	SURROGATE KEY OF CALENDAR TABLE
SUPPLIER_ID	NUMBER(10)	SURROGATE KEY OF SUPPLIER TABLE
QUANTITY_IN_INVENTORY	NUMBER(10)	TOTAL INVENTORY OF THE PRODUCT AT END OF DAY

Table A-10 describes the contents of the calendar table on the inventory data mart.

*Table A-10 SCOTT.*CALENDAR *table*

Column name	Data type	Description
CALENDAR_ID	NUMBER(10)	SURROGATE KEY OF INVENTORY DATAMART
C_DATE	DATE	DATE IN MM-DD-YYYY FORMAT

Column name	Data type	Description
C_YEAR	NUMBER(5)	YEAR
C_QUARTER	CHAR(10 BYTE)	QUARTER OF MONTH AS Q1,Q2,Q3,Q4
C_MONTH	VARCHAR2(100BYTE)	MONTH NAME
C_DAY	NUMBER(3)	DAY

Table A-11 describes the contents of the product table on the inventory data mart.

Table A-11 SCOTT.PRODUCT table

Column name	Data type	Description
PRODUCTKEY	NUMBER(10)	SURROGATE KEY OF INVENTORY DATAMART
PRODUCTID_NATURAL	VARCHAR2 (50 BYTE)	NATURAL ID FOR THE PRODUCT
PRODUCTNAME	VARCHAR2 (50 BYTE)	NAME OF PRODUCT
CATERGORYNAME	VARCHAR2 (50 BYTE)	CATEGORY NAME TO WHICH PRODUCT BELONGS
CATEGORYDESC	VARCHAR2 (100 BYTE)	CATEGROY DESCRIPTION TO WHICH PRODUCT BELONGS

Table A-12 describes the contents of the supplier table on the inventory data mart.

Table A-12 SCOTT.SUPPLIER table

Column name	Data type	Description
SUPPLIERKEY	NUMBER(10)	SURROGATE KEY OF INVENTORY DATAMART
SUPPLIERID_NATURAL	NUMBER(10)	NATURAL KEY OF THE SUPPLIER
COMPANYNAME	VARCHAR2(50BYTE)	COMPANY NAME OF SUPPLIER
CONTACTNAME	VARCHAR2(50BYTE)	CONTACT NAME OF SUPPLIER
CONTACTTITLE	VARCHAR2(50BYTE)	CONTACT TITLE OF SUPPLIER
ADDRESS	VARCHAR2(50BYTE)	CONTACT ADDRESS OF SUPPLIER
CITY	VARCHAR2(50BYTE)	CONTACT CITY OF SUPPLIER
REGION	VARCHAR2(50BYTE)	CONTACT REGION OF SUPPLIER
POSTALCODE	VARCHAR2(50BYTE)	CONTACT POSTALCODE OF SUPPLIER

Archived

Column name	Data type	Description
COUNTRY	VARCHAR2(50BYTE)	CONTACT COUNTRY OF SUPPLIER
PHONE	VARCHAR2(50BYTE)	CONTACT PHONE OF SUPPLIER
FAX	VARCHAR2(50BYTE)	CONTACT FAX OF SUPPLIER

Table A-13 describes the contents of the stores table on the inventory data mart.

Table A-13 SCOTT.STORES table

Column name	Data type	Description
STOR_ID	NUMBER(10)	SURROGATE KEY OF INVENTORY DATAMART
STOR_NAME	VARCHAR2(40BYTE)	NATURAL KEY OF THE STORE
STOR_ADDRESS	VARCHAR2(40BYTE)	STORE ADDRESS
CITY	VARCHAR2(40BYTE)	CITY NAME
STATE	VARCHAR2(40BYTE)	STATE TO WHICH STORE BELONGS
ZIP	VARCHAR2(50BYTE)	ZIP CODE
STORE_CATALOG_ID	NUMBER(10)	CATALOG TO WHICH STORE BELONGS

Data schemas on the SQL Server 2000 data mart

In this section we cover the tables contained in the sales data mart, built on SQL Server 2000. As part of the project, this schema, and the tables defined by it, were consolidated with the EDW schema. *DBO* is the user name for this data mart, and is thus part of each table name.

Table A-14 describes the contents of the calendar table on the sales data mart.

Table A-14 DBO.CALENDAR table

Column name	Data type	Description
C_DATEID_SURROGATE	INT	SURROGATE KEY OF SALES DATAMART
C_DATE	SMALLDATETIME	DATE IN MM-DD-YYYY FORMAT
C_YEAR	SMALLINT	YEAR
C_QUARTER	VARCHAR(50)	QUARTER OF MONTH AS Q1,Q2,Q3,Q4
C_MONTH	VARCHAR(50)	MONTH NAME
C_DAY	TINYINT	DAY

Table A-15 describes the contents of the product table on the sales data mart.

Table A-15 DBO.PRODUCT table

Column name	Data type	Description
PRODUCTKEY	INT	SURROGATE KEY OF SALES DATAMART
PRODUCTID_NATURAL	VARCHAR(100)	NATURAL ID FOR THE PRODUCT
PRODUCTNAME	VARCHAR(50)	NAME OF PRODUCT
CATERGORYNAME	VARCHAR(50)	CATEGORY NAME TO WHICH PRODUCT BELONGS
CATEGORYDESC	VARCHAR(100)	CATEGROY DESCRIPTION TO WHICH PRODUCT BELONGS

Table A-16 describes the contents of the supplier table on the sales data mart.

Table A-16 DBO.SUPPLIER table

Column name	Data type	Description
SUPPLIERKEY	INT	SURROGATE KEY OF SALES DATAMART
SUPPLIERID_NATURAL	INT	NATURAL KEY OF THE SUPPLIER
COMPANYNAME	VARCHAR(50)	COMPANY NAME OF SUPPLIER
CONTACTNAME	VARCHAR(50)	CONTACT NAME OF SUPPLIER
CONTACTTITLE	VARCHAR(50)	CONTACT TITLE OF SUPPLIER
ADDRESS	VARCHAR(50)	CONTACT ADDRESS OF SUPPLIER
CITY	VARCHAR(50)	CONTACT CITY OF SUPPLIER
REGION	VARCHAR(50)	CONTACT REGION OF SUPPLIER
POSTALCODE	VARCHAR(50)	CONTACT POSTALCODE OF SUPPLIER
COUNTRY	VARCHAR(50)	CONTACT COUNTRY OF SUPPLIER
PHONE	VARCHAR(50)	CONTACT PHONE OF SUPPLIER
FAX	VARCHAR(50)	CONTACT FAX OF SUPPLIER

Table A-17 describes the contents of the stores table on the sales data mart.

Table A-17 DBO.STORES table

Column name	Data type	Description
STOR_ID	INT	SURROGATE KEY OF SALES DATAMART
STOR_NAME	VARCHAR(50)	NATURAL KEY OF THE STORE

Archived

Column name	Data type	Description
STOR_ADDRESS	VARCHAR(100)	STORE ADDRESS
CITY	VARCHAR(50)	CITY NAME
STATE	VARCHAR(20)	STATE TO WHICH STORE BELONGS
ZIP	VARCHAR(50)	ZIP CODE
STORE_CATEG_ID	INT	CATEGORY

Table A-18 describes the contents of the store category table on the sales data mart.

Table A-18 DBO.STORE_CATEGORY table

Column name	Data type	Description
STORE_CATEG_ID	INT	SURROGATE KEY OF SALES DATAMART
STORE_CATEGORY	CHAR(50)	CATEGORY TO WHICH STORE BELONGS

Table A-19 describes the contents of the customer table on the sales data mart.

Table A-19 DBO.CUSTOMER table

Column name	Data type	Description
CUSTOMERKEY	INT	SURROGATE KEY OF SALES DATAMART
CUSTOMER_NATURALID	VARCHAR(100)	NATURAL ID OF THE CUSTOMER
COMPANY NAME	VARCHAR(100)	NAME OF COMPANY
CONTACT NAME	VARCHAR(100)	CONTACT NAME OF CUSTOMER
ADDRESS	VARCHAR(100)	CUSTOMER MAILING ADDRESS
CITY	VARCHAR(100)	CITY OF CUSTOMER
REGION	VARCHAR(100)	REGION OF CUSTOMER
POSTALCODE	VARCHAR(100)	POSTALCODE OF CUSTOMER
COUNTRY	VARCHAR(100)	COUNTRY OF CUSTOMER
PHONE	VARCHAR(100)	PHONE OF CUSTOMER
FAX	VARCHAR(100)	FAX OF CUSTOMER
CUSTOMER_CATG_ID	INT	ID OF THE CUSTOMERS CATEGORY

Archived

Table A-20 describes the contents of the employee table on the sales data mart.

Table A-20 DBO.EMPLOYEE table

Column name	Data type	Description
EMPLOYEEKEY	INT	SURROGATE KEY OF SALES DATAMART
EMPLOYEEID_NATURAL	INT	NATURAL ID FOR THE EMPLOYEE
REPORTS_TO_ID	INT	REPORTING MANAGERS ID(SURROGATE KEY)
FULLNAME	VARCHAR(50)	FULL NAME OF EMPLOYEE
LASTNAME	VARCHAR(50)	LASTNAME OF EMPLOYEE
FIRSTNAME	VARCHAR(50)	FIRSTNAME OF EMPLOYEE
MANAGERNAME	VARCHAR(50)	MANAGER NAME OF EMPLOYEE
DOB	DATETIME	DATE OF BIRTH
HIREDATE	DATETIME	HIRING DATE
ADDRESS	VARCHAR(60)	MAILING ADDRESS OF EMPLOYEE
CITY	VARCHAR(50)	CITY
REGION	VARCHAR(50)	REGION
POSTALCODE	VARCHAR(50)	POSTALCODE OF EMPLOYEE
COUNTRY	VARCHAR(50)	COUNTRY OF CITIZENSHIP
HOMEPHONE	VARCHAR(50)	RESIDENCE PHONE
EXTENSION	VARCHAR(50)	OFFICE PHONE AND EXTENSION

Table A-21 describes the contents of the store sales fact table on the sales data mart.

Table A-21 DBO.STORE_SALES_FACT table

Column name	Data type	Description
PRODUCTKEY	INT	SURROGATE KEY OF PRODUCT
EMPLOYEEKEY	INT	SURROGATE KEY OF EMPLOYEE
CUSTOMERKEY	INT	SURROGATE KEY OF CUSTOMER
SUPPLIERKEY	INT	SURROGATE KEY OF SUPPLIER
DATEID	INT	SURROGATE KEY OF CALENDAR
POSTRANSNO	INT	POINT OF SALES TRANSACTION NUMBER
SALESQTY	INT	SALES QUANTITY

Archived

Column name	Data type	Description
UNITPRICE	MONEY	UNIT PRICE OF PRODUCT
SALESPRICE	MONEY	SELLING PRICE OF PRODUCT
DISCOUNT	MONEY	DISCOUNT OFFERED ON A PRODUCT
STOREID	INT	SURROGATE KEY OF STORE

Archived

B

Data consolidation examples

In this appendix we show examples of one of the stages in a consolidation project. In particular, we show how we migrated the data from the Oracle and SQL Server data sources to the DB2 EDW staging area, in preparation for consolidation into the DB2 EDW. In each example, we used the objects that are depicted in Table B-1.

To do this, we used two IBM products:

- BM DB2 Migration ToolKit V1.3 (MTK)
- IBM WebSphere Information Integrator V8.2 (WebSphere II)

Table B-1 Objects transferred from source to staging area

Data mart	Object extracted	Software used	Staging area
Sales	Employee	Migration ToolKit 1.3	stagesql1
Sales	Calendar	Migration ToolKit 1.3	stagesql1
Sale	Product	Migration ToolKit 1.3	stagesql1
Sales	Vendor	Migration ToolKit 1.3	stagesql1
Sales	Stores	Migration ToolKit 1.3	stagesql1

© Copyright IBM Corp. 2005. All rights reserved.

Data mart	Object extracted	Software used	Staging area
Sales	Store_Category	Migration ToolKit 1.3	stagesql1
Sales	Customer	Migration ToolKit 1.3	stagesql1
Sale	Customer_Category	Migration ToolKit 1.3	stagesql1
Sales	Supplier	Migration ToolKit 1.3	stagesql1
Inventory	Calendar	Migration ToolKit 1.3	stageora1
Inventory	Stores	Migration ToolKit 1.3	stageora1
Inventory	Supplier	Migration ToolKit 1.3	stageora1
Inventory	Product	Migration ToolKit 1.3	stageora1
Inventory	Store_Inventory_Fact	Migration ToolKit 1.3	stageora1

DB2 Migration ToolKit

In this section we provide a brief overview of DB2 Migration ToolKit (MTK). We also demonstrate how to migrate data from existing data marts, in our project residing on Oracle 9i and SQL Server 2000, to DB2 UDB enterprise data warehouse (EDW).

The MTK is a free tool that can simplify and shorten the migration project. With MTK, you can automatically convert database objects such as tables, views, and data types, into equivalent DB2 database objects. It provides the tools needed to automate previously costly migration tasks.

MTK features

For all RDBMS source platforms, MTK converts:

- DDL
- SQL statements
- Triggers
- Procedures
- Functions

Archived

MTK enables the following tasks:

- Obtaining source database metadata (DDL) by EXTRACTING information from the source database system catalogs through (JDBC/ODBC).
- Obtaining source database metadata (DDL) by IMPORTING DDL scripts created by SQL*Plus or third-party tools.
- Automating the conversion of database object definitions, including stored procedures, triggers, packages, tables, views, indexes, and sequences.
- Deploying SQL and Java compatibility functions that permit the converted code to “behave” functionally similar to the source code.
- Conversion of PL/SQL statements using SQL Translator tool.
- Viewing conversion information and messages.
- Deployment of the converted objects into a new or existing DB2 UDB database.
- Generating and running data movement (unload/load) scripts or performing the data movement on-line.
- Tracking the status of object conversions and data movement, including error messages, error location, and DDL change reports using the detailed migration log file and report.

MTK GUI interface

The MTK GUI interface, depicted in Figure B-1, presents five tabs, each of which represents a specific task in the conversion process. The tabs are organized from left to right and are entitled:

- Specify Source
- Convert
- Refine
- Generate Data Transfer Scripts
- Deploy to DB2

The menu bar contains Application, Project, Tools, and Help:

- **Application:** This allows you to set up your preferences, such as an editor.
- **Project:** You can start a new project, open or modify an existing project, import SQL source file, or perform backup restore functions through this.
- **Tools:** You can launch to SQL Translator, reports, and the log.
- **Help:** This is the MTK help text.

Archived

Figure B-1 MTK GUI interface

Consolidating with the MTK

In this section we provide a brief overview of the migration tasks, as represented in the MTK. You are guided through the execution of these tasks by the MTK Graphical User Interface (GUI). We describe those tasks here, and demonstrate them in two examples shown later in this section.

Five basic tasks are defined for the migration process; each represented by Tabs in the MTK GUI, as depicted in Figure B-1. Here is a brief overview of the tasks:

- **Task 1: Specify source**

  The SPECIFY SOURCE task (Figure B-2) focuses on Extracting or Importing database metadata (DDL) into the tool. The database objects defined in this DDL will then be used as the source code for conversion to DB2 UDB equivalent objects. Extraction requires a connection to the source database through ODBC or JDBC. Once the ODBC/JDBC connection is established, MTK will “read” the system catalogs of the source database and extract the definitions for use in the conversion process.

Archived

IMPORTING, on the other hand, requires an existing file, or files, which contain database object DDL. The Import task copies the existing DDL from the file system into MTK project directory for use in the database structure conversion process. Using MTK to perform data movement will be limited if IMPORTING is chosen.

Figure B-2 Specify source

- **Task 2: Convert**

  During the CONVERT task (Figure B-3), the user may complete several optional tasks before the actual conversion of the source code. These are:

  – Selecting format options for the converted code. Examples of options are: including the source code as comments in the converted code; including DROP before create object statements, among others.

  – Making changes to the default mapping between a source data type and its target DB2 data type.

  Once the optional tasks are completed, the user can click the **Convert** button and the source DDL statement is converted into DB2 DDL.

  Each conversion generates two files:

  – The **db2** file contains all of the source code converted to DB2 UDB target code.

  – The **.rpt** file can be opened and viewed from this pane, but it is best to examine it during the next task, which is Refine.

Archived

Figure B-3 Convert

- **Task 3: Refine**

  During the REFINE task (Figure B-4) the user may:

  – Examine the results of the conversion

  – View various types of messages generated by the tool and, if necessary specify changes to be made to the converted DDL

  If the user makes any changes to the converted DDL, they must return to the Convert step to apply the changes.

  You can use other tools such as the SQL Translator, Log, and Reports to help you refine the conversion. After you have refined the DB2 DDL statements to your satisfaction, you can move on to the Generate data transfer scripts step to prepare the data transfer scripts, or the Deploy to DB2 step to execute the DB2 DDL statements.

Archived

Figure B-4 Refine

- **Task 4: Generate data transfer scripts**

  In the GENERATE DATA TRANSFER task (Figure B-5), scripts are generated that will be used to:

  – Unload data from the source environment
  – Load or Import data into DB2 UDB

  Before creating the scripts, you may choose some advanced options that will affect how the IMPORT or LOAD utility operates. This will allow the user to refine the Load or Import specifications to correspond with the requirements of their data and environment.

Figure B-5 Generate Data Transfer script

Archived

- **Task 5: Deploy to DB2**

  The DEPLOY task (Figure B-6) is used to install database objects and Import/Load data into the target DB2 database. In this task, you can:

  – Choose to create the database or install the objects in an existing database.

  – Execute the DDL to create the database objects.

  – Extract data from the source database.

  – Load/import the source data into the target DB2 tables or choose any combination of the above three.

Figure B-6 Deploy to DB2

Archived

An overview of all the tasks in the MTK conversion process is shown in Figure B-7.

Figure B-7 MTK conversion tasks overview

Archived

Example: Oracle 9i to DB2 UDB

In this section we demonstrate using the MTK to transfer data from Oracle to DB2. In Figure B-8 we depict the test environment used, and highlight the activity of transferring data from Oracle 9i to DB2 UDB Version 8.2. The MTK will need to be configured to enable transfer of data from any source to DB2.

Figure B-8 Test environment - Oracle to DB2

Archived

Configuring MTK for data transfer

After installing the MTK, you are prompted to create a project. However, you can also create a new project at any time. The Project Management screen prompts you to enter the configuration parameters for the project. In our example we use the values in Figure B-9 to create the project.

By then clicking **OK**, we proceed to the MTK screen to specify the data source.

Figure B-9 Creating a new project

Specify source

You are prompted to choose the database to which you would like to connect. The MTK screen to specify the database name is depicted in Figure B-10. You must have installed your client for ORACLE to access the target database. You can also use an ODBC or a Service Name to connect to Oracle. Fill in the user name and password, and click **OK** to continue.

Figure B-10 Connect to Database

This task now focuses on extracting or importing the source metadata (DDL) into the MTK. If a database connection does not exist for this project, click the **Connect to Database** button depicted in Figure B-11.

Otherwise, specify which object you need to extract. You can also extract views procedures, and triggers to the target format. Select the tables to include, and click the **Extract** tab.

Archived

Figure B-11 Extract DDL from the source database

Extract

For the extract, you have the following options:

- **Create one file per stored procedure:** Specifies to have each stored procedure listed as a separate file in the project subdirectory. If items are specified in the *Include other needed objects?* section, the necessary tables, views, and data types will be placed in the root extraction file. Procedure specifications will be listed above the place from which they are called.
- **Include other needed objects:** Specifies whether all object dependencies should be included in the extraction. For example, if you select procedure *p* for extraction and it references table *t*, then table *t* will also be extracted. It is possible that some required objects might not be included even though this control option is selected. For example, system tables are never extracted. In some instances, source catalog tables are not always accurately maintained by the source database system.

  This option is designed to allow you to target specific objects, for example, to test migration scenarios. If you are migrating a large database with many objects, you will most likely want to break the migration into separate manageable files, converting the tables first, followed by triggers, procedures, and other objects.

Unless you are keenly aware of every reference to each object, do not use the *Include other needed objects?* option in a full migration such as just described. If you do, then the same object will likely be redefined many times, in which case MTK will post an error during conversion each time it encounters a duplicate definition.

- **Make context file:** Select this option to have any other needed objects put into a file with a context extension. The context file is put at the top of the list in the window on the extractor panel since these objects are depended upon by statements in the *.src file*.
- **Connect to database:** Used for multiple extractions. If you want to connect to a different database server while in the Extract window, click this button. The *Connect to Database* window opens, where you can specify a different alias, user ID, and password for the new connection.
- **Refresh available objects:** Click this tab to update the list of available objects from the current source database. You must refresh the objects:
  - To update any changes that have occurred to the database after you initially connected.
  - After making a new database connection.
- **Set quoted_identifier on:** Select this option if any of the selected objects include spaces in the names, or if the objects were created in a database session with QUOTED_IDENTIFIER ON. This option should only be used to extract individual objects that require it. Then click the **Extract** button to create a DLL file and continue to the next step.

Convert

The purpose of the Convert step is to convert source metadata to DB2 UDB metadata. In Figure B-12 we changed the default source schema option to *specify_schema_for_all_objects*. This is used to change from ORACLE owner of the objects, SCOTT, to the DB2 schema STAGEORA1.

Archived

Figure B-12 Converting Data

In the following list we describe the Convert Options in Figure B-12:

- **Source date format:** Select or type the format for the date constants when they are converted. The format must match that used in the source. Search data in your database to see all of contents and use the appropriate format, and test the conversion data in each step.

  The value specified depends upon the DBDATE environment variable. If DBDATE is not specified for the source database a default will be taken. As an example, the default date format for Informix is MDY4/ - as depicted in Table B-2.

Table B-2 Source data formats

Form	Example
MDY4/	12/03/2004
DMY2	03-12-2004
MDY4	12/03/2004
Y2DM	04.03.12
MDY20	120304
Y4MD	2004/12/03

- **Set DELIMIDENT:** Select this to indicate that the source SQL contains object names that use delimited identifiers (case-sensitive names within quotation marks, which can contain white space and other special characters). This setting must match the setting of the Informix DELIMIDENT environment variable setting for the source SQL.
- **DB2 UDB variable prefix:** This option is only available for conversions from Sybase or SQL Server. If a source variable begins with a prefix of @, the prefix must be changed to an acceptable DB2 UDB prefix. The default prefix chosen is v_, as indicated in the DB2 UDB variable prefix field. For example, @obj becomes v_obj after conversion to DB2 UDB.

  If you want to chose your own prefix, type the prefix into the DB2 UDB variable prefix field. For example, typing *my* into the field, results in *@obj* becoming *myobj* after conversion to DB2 UDB.
- **Default source schema:** You can specify the object name qualifier that you want to be used as the default schema in DB2 UDB. If a source database is extracted, then the list is populated with the name qualifiers that are available in the source file. The name you choose specifies those objects that you want to belong to the default schema in DB2 UDB, and will therefore have no schema name assigned. If you choose *from_first_object*, then the name qualifier of the first object encountered in the source file will be used as the default.

  Objects that have no qualifying name will be assigned a default schema name, as depicted in Table B-3.

Table B-3 Default schema names

informix
dbo
dba

  You can force every object to be given a schema name by selecting *specify_schema_for_all_objects* or by entering an unused qualifier as the default source schema.

  You can force a particular schema name for a set of objects by including a CONNECT SCHEMA_NAME statement in the source file. All objects that follow the connect statement are assigned the specified schema name.

  Select the *Input file contains DBCS characters* (incompatible with UTF-8) check box if the object names contain DBCS characters.
- **View source:** Click to display the source SQL file in an external file editor (defined in the Preference window). Note that MTK does not modify the source file during conversion, but you can make changes using the editor and reconvert.

- **View output:** Click to display the output DB2 UDB SQL file in an external file editor (defined in the Preferences window). However, you can take advantage of many more features if you use the Refine page to view the conversion results.

Refine

The Refine step gives you the opportunity to view the results of the conversion and to make changes.

The recommended strategy for addressing messages for a clean deployment is to first alter the source SQL and re-convert. When you can no longer address any problems by changing the source, alter the final DB2 UDB output before deploying to DB2 UDB.

To refine the conversion:

Change the names of objects using the tools provided on the Refine page. MTK keeps track of the name mapping each time you re-convert.

To edit the body of procedures, functions, and triggers, use either the editor on the Refine page or edit the original source,

> **Important:** Do not use both methods. If you have a need to edit the source for other reasons, you should edit procedures, functions, and triggers in the original source as well. Mixing the source editing methods can produce unpredictable results.

To apply the changes, you must go back to the Convert page and click **Convert**. Upon re-conversion, the translator merges the changes with the original extracted source metadata to produce updated target DB2 UDB and XML metadata. The original metadata is not changed (unless you edited it directly).

Repeat the refine-convert process to achieve as clean a result as possible.

When you have exhausted making all possible changes to the source metadata, you can modify the resulting DB2 UDB SQL file as necessary for a successful deployment. Be sure to make a backup copy first. Do not return to the Convert step after making any manual DB2 UDB SQL changes. Conversion of the source metadata replaces the existing DB2 UDB file, destroying any manual changes.

Tools other than those on the refine page exist to help you while you refine the conversion. They are the SQL Translator, Log, and Reports.

Archived

Once you have the DB2 UDB source tuned to your satisfaction, you can either go to the Generate Data Transfer Scripts page to prepare the scripts for data transfer or go directly to the Deploy page to deploy the DB2 UDB metadata.

As shown in Figure B-13, for example, set the DB2 name (SCHEMA) to STAGEORA1 and go to step, “Convert” on page 328 and redo this action. All of yours scripts will be converted.

Figure B-13 Refine Data

Generate data transfer scripts

If you plan to modify the load or import options, you should have an understanding of the DB2 UDB load and import options. For more information on the LOAD and IMPORT commands, refer to the DB2 UDB Command Reference (SC09-4828). For more information on the DB2 UDB data movement and other administrative tasks, see the DB2 UDB Data Movement Utilities Guide and Reference (SC09-4830) and the DB2 UDB Administration Guide: Implementation (SC09-4820).

In this step you set any data transfer options and generate both the deployment and data transfer scripts.

Important: Deployment scripts and data transfer scripts are created in this step. Even if you are not transferring data, this step must be completed to obtain the deployment scripts.

Restriction: The data scripts are written specifically for the target DB2 UDB database that will be deployed in the next step. Do not attempt to load the data into a database created by other means. Also ensure that you are completely satisfied with the conversion results before you use any data transfer scripts.

After defining all the methods, click **Create Scripts**, as depicted in Figure B-14.

Figure B-14 Generate Data to Transfer

Deploy to DB2

MTK can deploy the database to a local or remote system. You can deploy the converted objects and data to DB2 UDB at the same time or separately. For example, you might want to load the metadata during the day along with some sample data to test your procedures, and later load the data at night when the database has been tested and when network usage is low. When MTK deploys data, it extracts the data onto the system running MTK before loading it into the database.

Choose the name of your target database that has already been created on the Data Mart Consolidation Environment, type your user and password, and click **Deploy**. As you see in the Figure B-15 and the data will be transferred to DB2. One report will be generated by DB2 after the conversion. Some error messages you can bypass because you have reviewed and understand the differences between the original code and the how the MTK makes its conversion. Click **Deploy** and your database and scripts will be generated automatically by MTK transferring the data to the Stagging area.

Figure B-15 Deploy to DB2

For more information on this subject, please refer to the IBM Redbook, *Oracle to DB2 UDB Conversion Guide*, SG24-7048.

Example: SQL Server 2000 to DB2 UDB

In this section we demonstrate also how to use MTK to transfer data from SQL Server to DB2. Figure B-16 depicts the environment. The data resides on the Sales data mart, and we want to move it to the EDW on DB2 Version 8.2. To transfer the data, you must install the DB2 client and configure it to access DB2 on AIX.

The tasks are now basically the same as in “Example: Oracle 9i to DB2 UDB” on page 324, except we are using SQL Server.

Figure B-16 Migration Diagram from SQL Server to DB2

Archived

Specify source

First you create a project. In our example we use the values depicted in Figure B-17. Click **OK**.

Figure B-17 New Project

Archived

Now you are prompted to choose and connect to the database, as depicted in Figure B-18. You must have first installed your SQL Server client to access the target database. Then enter the user and password, and click OK. The SPECIFY SOURCE task focuses on Extracting or Importing database metadata (DDL) into the tool. Click Extract. If a database connection does not exist for this project, the Connect to Database window opens. You can use a ODBC or DSN Name to connect to SQL Server. Fill the fields with the appropriate data.

Figure B-18 Connect to Database

Archived

After clicking **OK**, you will see Figure B-19; and then you specify which objects you need to extract. You can also extract views, procedures, and triggers to the target format. To see more details on this option, go to “Extract” on page 327.

Figure B-19 Extract DDL from the source database

Convert

Now you can convert source metadata to DB2 UDB metadata. This is also used to change from SQL Server owner *dbo* of the objects to the DB2 schema STAGESQL1. The screen to specify these options is depicted in Figure B-20.

Figure B-20 Converting data

For additional details on these options, see the section, “Convert” on page 328.

Refine

The Refine step gives you the opportunity to view the results of the conversion and to make changes.

The recommended strategy for addressing messages for a clean deployment is to first alter the source SQL and re-convert. When you can no longer address any problems by changing the source, alter the final DB2 UDB output before deploying to DB2 UDB.

To refine the conversion, change the names of objects using the tools provided on the Refine page. MTK keeps track of the name mapping each time you re-convert.

In Figure B-21, we show an example to set the new DB2 name (SCHEMA) to STAGESQL1. You can then go back to the Convert task and redo it. All of the scripts will be converted.

To edit the body of procedures, functions, and triggers, use either the editor on the Refine page or edit the original source.

> **Important:** Do not use both methods. If you have a need to edit the source for other reasons, you should edit procedures, functions, and triggers in the original source as well. Mixing the source editing methods can produce unpredictable results.

To apply the changes, you must go back to the Convert page and click Convert. Upon re-conversion, the translator merges the changes with the original extracted source metadata to produce updated target DB2 UDB and XML metadata. The original metadata is not changed (unless you edited it directly).

Repeat the refine-convert process to achieve as clean a result as possible.

When you have exhausted making all possible changes to the source metadata, you can modify the resulting DB2 UDB SQL file as necessary for a successful deployment. Be sure to make a backup copy first. Do not return to the Convert step after making any manual DB2 UDB SQL changes. Conversion of the source metadata replaces the existing DB2 UDB file, destroying any manual changes.

Tools other than those on the refine page exist to help you while you refine the conversion. They are the SQL Translator, Log, and Reports.

Once you have the DB2 UDB source tuned to your satisfaction, you can either go to the Generate Data Transfer Scripts page to prepare the scripts for data transfer or go directly to the Deploy page to deploy the DB2 UDB metadata.

Archived

Figure B-21 Refine Data

Generate data transfer scripts

In this step you set any data transfer options and generate both the deployment and data transfer scripts.

Important: Deployment scripts and data transfer scripts are created in this step. Even if you are not transferring data, this step must be completed to obtain the deployment scripts

Restriction: The data scripts are written specifically for the target DB2 UDB database that will be deployed in the next step. Do not attempt to load the data into a database created by other means. Also ensure that you are completely satisfied with the conversion results before you use any data transfer scripts.

After defining all the methods, click the **Create Scripts** button as depicted in Figure B-22.

Figure B-22 Generate Data to Transfer

Deploy to DB2

MTK can deploy the database to a local or remote system.

You can deploy the converted objects and data to DB2 UDB at the same time or separately. For example, you might want to load the metadata during the day along with some sample data to test your procedures, and later load the data at night when the database has been tested and when network usage is low.

When MTK deploys data, it extracts the data onto the system running MTK before loading it into the database.

Click **Deploy** and the data will be transferred to the DB2 as depicted in Figure B-23.

Figure B-23 Deploy to DB2

For more information on this subject, please refer to the IBM Redbook, *Microsoft SQL Server to DB2 UDB Conversion Guide*, SG24-6672.

Consolidating with WebSphere II

In this sample scenario we have migrated the fact tables from the two data marts in Oracle and SQL Server to the staging area in the DB2 EDW using WebSphere II. The connections to the fact tables in the two data marts are established by creating Nicknames in the DB2 database using WebSphere II.

Table B-4shows the details of the table migrated from SQL Server to DB2.

Table B-4 Table migrated from SQL Server DB2

Data mart	Table name	Staging Area
Sales	Store_Sales_Fact	stagesql1

Table B-5 shows the details of the table migrated from Oracle to DB2.

Table B-5 Table migrated from Oracle to DB2

Data mart	Table name	Staging Area
Inventory	Store_Inventory_Fact	stageora1

Example - Oracle 9i to DB2 UDB

The following steps describe the configurations involved in setting up WebSphere II for establishing a link between the table located on Oracle 9i and your staging area in DB2.

- Configuration information for Oracle 9i wrapper
- Creating the Oracle wrapper
- Creating the Oracle server
- Creating Oracle user mappings
- Creating Oracle nicknames

We use the Control Center on a Windows platform to perform the necessary administration steps.

Configuration information for Oracle 9i wrapper

The information in Table B-6 is necessary to integrate the Oracle 9i data source:

Archived

Table B-6 Oracle information

Parameter	Value
ORACLE_HOME	`/home/oradba/orahome1`
Port	1521
User/Password	system/oraserv
Schema	scott

Table B-8 displays the DB2 server information.

Note: Make sure you have installed the Oracle Client on the federated server, and that you have successfully configured and tested the connection to your Oracle server.

The steps to configure connection to Oracle from DB2 on AIX are as follows.

- Configure Oracle tnsnames.ora
- Update the db2dj.ini file

Configuring Oracle tnsnames.ora

The Oracle tnsnames.ora contains information the Oracle Client uses to connect to the Oracle server. The file is usually located in the /network/admin sub-directory of the Oracle Client.

There needs to be an entry for the Oracle server that enables federated access. The name at the beginning of the entry is called the Oracle Network Service Name value, and is the value that will be used as the NODE setting of our WebSphere II server definition to the Oracle server.

Example B-1 shows our tnsnames.ora file. The Network Service Name that we will use as the NODE setting is highlighted.

Example: B-1 The tnsnames.ora file

```
NILE.ALMADEN.IBM.COM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = nile)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ITSOSJ)
    )
  )
```

Archived

Update the db2dj.ini file

The DB2 instance owner /sqllib/cfg/db2dj.ini file must contain the variable ORACLE_HOME. Here we discuss the Oracle variables in db2dj.ini:

- ORACLE_HOME - required. It indicates the Oracle Client base directory. In our sample scenario, The Oracle_Home variable is assigned the value as shown in Example B-2.
- TNS_ADMIN - optional. It indicates directory containing tnsnames.ora file. Only required if the tnsname.ora file is not in the default location, which is the Oracle Client's /network/admin sub-directory.
- ORACLE_NLS - optional.
- ORACLE_BASE - optional.

Example: B-2 Sample db2dj.ini file entry

```
ORACLE_HOME=/home/oradba/OraHome1
TNS_ADMIN=/home/oradba/OraHome1/network/admin
```

Creating the Oracle wrapper

Here are the steps to create the Oracle wrapper:

1. Open the DB2 Control Center.
2. Expand your instance and your database to **EDWDB**.
3. Right-click **Federated Database Objects** for the database **EDWDB** and click **Create Wrapper**.
4. Choose the data source type and enter a unique wrapper name like shown in Figure B-24. If your Oracle data source has Version 8i or higher, select **Oracle using OCI 8**, if not choose **Oracle using OCI 7**.
5. Click **OK.**

Archived

Figure B-24 Oracle - Create Wrapper

Example B-3 shows the command line version of creating the wrapper for your Oracle instance. Additionally, you may check your wrapper definition in the DB2 system catalogue with the two select statements included in Example B-3.

Example: B-3 Oracle - Create wrapper statement

```
CONNECT TO EDWDB;
CREATE WRAPPER "ORACLE" LIBRARY 'libdb2net8.a';

SELECT * from SYSCAT.WRAPPERS;
SELECT * from SYSCAT.WRAPOPTIONS;
```

Creating the Oracle server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the Oracle wrapper, we need to specify the Oracle server, from which you want to access data. For your wrapper defined, you can create several server definitions.

Here are the steps to create an Oracle server:

1. Select the wrapper you created in the previous step - **ORACLE.**
2. Right-click **Servers** for wrapper **ORACLE** and click **Create.**

The server definition demands the following inputs, visualized in Figure B-25:

- **Name**: The name of the server must be unique over all server definitions available on this federated database.
- **Type**: Select **ORACLE**
- **Version**: Select **8** or **9.** If you use Oracle wrapper using OCI 7, select the correct version of your Oracle data source server.

Figure B-25 Oracle - Create Server dialog

Switch to the **Settings** menu to complete your server edition. For server settings visualized in Figure B-26, some input fields require definition, and some are optional. The first two fields, Node and Password, are required.

Figure B-26 Oracle - Create Server - Settings

Example B-4shows the command line version of creating the server for your Oracle instance. Additionally, you may check your server definition in the DB2 federated system catalog with the two select statements listed below.

Example: B-4 Oracle - Create server statement

```
CONNECT TO FEDDB;
CREATE SERVER ORASRC TYPE ORACLE VERSION '9' WRAPPER "ORACLE" OPTIONS( ADD NODE
'NILE.ALMADEN.IBM.COM', PASSWORD 'Y');

SELECT * from SYSCAT.SERVERS;
SELECT * from SYSCAT.SERVEROPTIONS;
```

Creating Oracle user mappings

When the federated server needs to access the data source server, it first needs to establish the connection. With the user mapping, you define an association from a federated server user ID to an Oracle user ID and password that WebSphere II uses in connections to the Oracle server on behalf

of the federated user. An association must be created for each user who will be using the federated system. In our case, we only use the user ID of db2mart.

Here are the steps to create an Oracle server:

1. Select the server we created in the previous step - **ORASRC**.
2. Right-click **User Mappings** for server **ORASRC** and click **Create**.

Figure B-27 lists all the users IDs available on your federated system. Select the user who will be the sender of your distributed requests to the Oracle data source. We selected the owner of our federated server instance, **db2mart.**

Figure B-27 Oracle - Create User Mappings

Switch to settings as shown in Figure B-28, to complete the user mapping. You need to identify the username and password to enable the federated system to connect to our Oracle data source.

Figure B-28 Oracle - Create User Mappings - Settings

Example B-5 shows the command line version of creating the user mapping for your Oracle instance. Additionally, you may check your user mapping definition in the DB2 federated system catalog with the select statement listed at the end.

Example: B-5 Oracle - Create user mapping statement

```
CONNECT TO EDWDB;
CREATE USER MAPPING FOR "DB2MART" SERVER "ORASRC" OPTIONS ( ADD REMOTE_AUTHID
'system', ADD  REMOTE_PASSWORD '*****') ;

SELECT * from SYSCAT.USEROPTIONS;
```

Creating Oracle nicknames

After having set up the Oracle wrapper, the server definition and the user mapping to our Oracle database, we finally need to create the actual link to a table located on our remote database as a nickname.

When you create a nickname for an Oracle table, catalog data from the remote server is retrieved and stored in the federated global catalog.

Steps to create a DB2 Oracle nickname:

1. Select the server **ORASRC**.
2. Right-click **Nicknames** for the server **ORASRC** and click **Create**.

You will then see a dialog. You have two possibilities to add a nickname. Either click **Add** to manually add a nickname by specifying local and remote schema and table identification, or use the **Discover** functionality. In our sample scenario,

we click the **Add** button to add a nickname to the database. Figure B-29 shows the Add Nickname dialog box where the Nickname details are specified for the remote table ‘Store_Inventory_Fact’.

Figure B-29 Oracle - Add nickname

Once the details for the remote table is specified, click the **OK** button to add the Nickname to the selection list shown in Figure B-30.

Figure B-30 Oracle - Selection list for creating nicknames

If the **Discover** filter is used to add entries to the Create Nickname window, the default schema will be the user ID that is creating the nicknames. Use the **Schema** button to change the local schema for your Oracle server nicknames.

> **Tip:** We recommend that you use the same schema name for all Oracle nicknames in your federated DB2 Database.

Example B-6 shows the command line version of creating the nicknames for your Oracle instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements listed in the example.

Example: B-6 Oracle - Create nickname statements

```
CONNECT TO EDWDB;
CREATE NICKNAME ORACLE.STORE_INVENTORY_FACT FOR
ORASRC.SCOTT.STORE_INVENTORY_FACT;

SELECT * from SYSCAT.TABLES WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA='ORACLE';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA='ORACLE';
```

> **Note:** For further information on WebSphere II configuration and installation please refer to the redbook, Data Federation with IBM DB2 Information Integrator V8.1, SG24-7052. Also note that since publication of that redbook, the name has changed from *DB2 Information Integrator* to *WebSphere Information Integrator*.

Example - SQL Server to DB2 UDB

The following steps describe the configurations involved in setting up the WebSphere II. It is basically the same as described for Oracle in “Example - SQL Server to DB2 UDB” on page 353.

Archived

Microsoft SQL Server client configuration in AIX

Table B-7 contains SQL Server parameters used in configuring data transfer from SQL Server.

Table B-7 MIcrosoft SQL Server Information

Parameter	Value
Server	**nile.almaden.ibm.com**
DB Name	**Store_Sales_DB**
Port	**1433**
User/Password	**sa/sqlserv**

Table B-8 contains the parameters necessary for configuring data transfer to DB2.

Table B-8 DB2 Server Information

Parameter	Value
Server	**clyde.almaden.ibm.com**
Instance Name	**db2mart**
DB Name	**EDWDB**
Port	**3900**
User/Password	**db2mart/db2serv**

In order to access the SQL Server database from DB2/AIX using WebSphere II, we have to install DataDirect ODBC driver on the federated server.

The Installation guide for DataDirect ODBC driver can be found in the Web site

`http://www.datadirect.com/index.ssp`

In our environment, the DataDirect ODBC driver is installed under /opt. The directory created by the installation of DataDirect driver under /opt is odbc32v50.

These are the steps to configure connection to SQL Server from AIX:

1. Update the .odbc.ini files.
2. Update the .profile file of user.
3. Update the db2dj.ini file.
4. Update the DB2 environment variables.
5. Test load the SQL Server library.

Archived

Update the .odbc.ini files

On AIX, the .odbc.ini file contains information the DataDirect Connect ODBC driver for Microsoft SQL Server uses to connect to the SQL Server server. The file can be anywhere on the system. The ODBC_INI variable in the db2dj.ini file will tell Information Integrator to find the .odbc.ini file it is to use. It is recommended that a copy of the .odbc.ini file containing an entry for the SQL Server server be placed in the DB2 instance owner's home directory. There needs to be an entry for the SQL Server server to which we will define federated access. The name of the entry is called the Data Source Name value, and is the value that will be used as the NODE setting of our Information Integrator server definition to the SQL Server server.

Example B-7 shows the .odbc.ini entry for SQL Server. The Data Source Name that we will use as the NODE setting in our Information Integrator server definition is SQL2000.

Example: B-7 The .odbc.ini file entry for SQL Server

```
SQL2000=SQL Server
[SQL2000]
Driver=/opt/odbc32v50/lib/ivmsss20.so
Address=nile,1433
```

Note: Other parameters can be included in the .odbc.ini entry for the SQL Server server. The example shows the minimum required for Information Integrator to use the entry to connect to the SQL Server server. For Information Integrator on Windows, the entry for the SQL Server server needs to be in Windows ODBC Data Source Administration; the entry needs to be a system DSN.

Update the .profile file of user

In our scenario we update the .profile file of the DB2 Instance owner userid, which is db2mart. The following entries have to be updated in the .profile file:

```
ODBCINI=/opt/odbc32v50/odbc.ini; export ODBCINI
export LIBPATH=/opt/odbc32v50/lib:$LIBPATH
```

Where /opt/odbc32v50 is the installation path of the DataDirect ODBC driver.

Update the db2dj.ini file

The DB2 instance owner's /sqllib/cfg/db2dj.ini file must contain the variable ODBC_INI and DJX_LOAD_LIBRAY_PATH. Here is discussion of the variables for SQL Server in db2dj.ini.

Archived

Entries are only required on UNIX. No entries are required in db2dj.ini on Windows for Information Integrator to connect a SQL Server server:

- DJX_ODBC_LIBRARY_PATH - required. Indicates location of the DataDirect Connect Driver Manager (libodbc.a) and SQL Server ODBC driver. In our case the value is set to:

  ```
  DJX_LOAD_LIBRARY_PATH=/opt/odbc32v50/lib
  ```

- ODBC_INI - required.

  Indicates the full path to the .odbc.ini file that Information Integrator is to use. In our case the value is set to:

  ```
  ODBC_INI=/home/db2mart/.odbc.ini
  ```

  Where /home/dbemart is the home directory of the db2 Instance owner db2mart.

Important: If the .odbc.ini file is in the DB2 instance owner's home directory (like in the example), you may be tempted to use $HOME in the specification. Do not do this, as it will cause errors (perhaps even bring DB2 down).

Update the DB2 environment variables

On AIX, two DB2 environment variables must be set when using the WebSphere II Relational Wrapper for SQL Server. These variables are not required on Windows:

- db2set DB2ENVLIST=LIBPATH.
- db2set DB2LIBPATH=<path of the directory containing the DataDirect Connect driver manager (libodbc.a) and SQL Server ODBC driver>

Example B-8 shows the values for the environment variables in our test scenario.

Example: B-8 DB2 Environment variables

```
DB2LIBPATH=/opt/odbc32v50/lib
DB2ENVLIST=LIBPATH
```

Test load the ODBC library file for SQL Server

You can test to determine if the configuration you have done is correct by loading the ODBC library file 'ivmsss20.so' for SQL Server under the 'lib' directory of the DataDirect installation path.

The DataDirect installation path in our test scenario is: /opt/odbc32v50.

Example B-9 shows the command that we executed in our test server to load the ODBC library file for SQL Server and the output message.

Example: B-9 Command to test load the ODBC library file for SQL Server

```
$ ivtestlib ivmsss20.so
Load of ivmsss20.so successful, qehandle is 0x2
File version: 05.00.0059 (B0043, U0029)
```

If you receive an error message on executing the command, you will have to check to see that all the related configuration settings are correct.

Creating the Microsoft SQL Server wrapper

Here are the steps (see Figure B-31):

1. Expand your instance and database.
2. Right-click **Federated Database Objects** for database **EDWDB** and click **Create Wrapper.**
3. Choose data source type **Microsoft SQL Server** and enter a unique wrapper name**.**
4. Click **OK**.

Figure B-31 Microsoft SQL Server - Create Wrapper dialog

Archived

Example B-10 shows the command line version of creating the wrapper for your Microsoft SQL Server instance. Please check the wrapper definition with the two DB2 system catalog tables listed.

Example: B-10 Microsoft SQL Server - Create wrapper statement

```
CONNECT TO EDWDB;
CREATE WRAPPER "MSSQL" LIBRARY 'libdb2mssql3.a';
SELECT * from SYSCAT.WRAPPERS;
SELECT * from SYSCAT.WRAPOPTIONS;
```

Creating the Microsoft SQL Server server

A server definition identifies a data source to the federated database. A server definition consists of a local name and other information about that data source server. Since we have just created the Microsoft SQL wrapper, we need to specify the SQL Server, from which you want to access data. For your wrapper defined, you can create several server definitions.

Here are the steps to create a Microsoft SQL Server:

1. Select the wrapper you created in the previous step - **MSSQL**.
2. Right-click **Servers** for wrapper **MSSQL** and click **Create**.

Archived

The server definition requires the following inputs, shown in Figure B-32:

- **Name**: The name of the server must be unique over all server definitions available on this federated database.
- **Type**: Select MSSQLSERVER
- **Version**: Select 6.5, 7.0 or 2000

Figure B-32 Microsoft SQL Server - Create Server

Switch to the Settings menu to complete your server definitions, as displayed in Figure B-33.

Figure B-33 Microsoft SQL Server - Create Server - Settings

For a Microsoft SQL Server, you need to specify the first three options, Node, DBName and Password, in order to entirely define the connection. All other server options are optional. Server options are used to describe a data source server. You can set these options at server creation time, or modify these settings afterwards.

Example B-11 shows the command line version of creating the wrapper for your Microsoft SQL instance. Additionally, you may check your server definition in the DB2 system catalogue with the two select statements listed below.

Example: B-11 Microsoft SQL Server - Create server statement

```
CONNECT TO EDWDB;
CREATE SERVER MSSQL2000 TYPE MSSQLSERVER VERSION '2000' WRAPPER "MSSQL"
OPTIONS( ADD NODE 'SQL2000', DBNAME 'store_sales_db', PASSWORD 'Y');

SELECT * from SYSCAT.SERVERS;
SELECT * from SYSCAT.SERVEROPTIONS;
```

Creating Microsoft SQL Server user mappings

When the federated server needs to access the data source server, it needs to establish the connection with it first. With the user mapping you define an association from a federated server user ID and password to an SQL Server user ID and password. An association must be created for each user that will be using the federated system. In our case, we only use one user ID, db2mart.

Here are the steps to create a user mapping definition:

1. Select the server we created in the previous step - **MSSQL2000**.
2. Right-click **User Mappings** for server **MSSQL2000** and click **Create**.

Figure B-34 lists all users IDs available on your federated system. Select the user who will be the sender of your distributed requests to the Microsoft SQL data source. We select the owner of our federated server instance, **db2mart**.

Figure B-34 Microsoft SQL Server - Create user mappings

Switch to **Settings**, as shown in Figure B-35, to complete the user mapping. You need to identify the username and password to enable the federated system to connect to our DB2 Microsoft SQL data source.

Figure B-35 Microsoft SQL Server - Create user mappings - Settings

Example B-12 shows the command line version to create the user mapping for your SQL Server instance. Additionally, you may check your user mapping definition in the DB2 system catalog with the SELECT statements listed below.

Example: B-12 Microsoft SQL Server - Create user mapping statement

```
CONNECT TO EDWDB;
CREATE USER MAPPING FOR "DB2MART" SERVER "MSSQL2000"
OPTIONS ( ADD REMOTE_AUTHID 'sa', ADD  REMOTE_PASSWORD '*****') ;

SELECT * from SYSCAT.USEROPTIONS;
```

Creating Microsoft SQL Server nicknames

After setting up the Microsoft SQL wrapper, the server definition, and the user mapping to our Microsoft SQL Server database, we finally need to create the actual link to a table located on our remote database as a nickname.

When you create a nickname for a Microsoft SQL Server table, catalog data from the remote server is retrieved and stored in the federated global catalog.

Here are the steps to create a Microsoft SQL nickname:

1. Select the server **MSSQL2000**.
2. Right-click **Nicknames** for server **MSQL2000** and click **Create**.

A dialog is displayed. You have two possibilities to add a nickname. Either you click **Add** to manually add a nickname by specifying local and remote schema and table identification, or you can use the **Discover** functionality. Figure B-36 shows the Add Nickname dialog box where the Nickname details are specified for the remote table 'Store_Sales_Fact'.

Figure B-36 Microsoft SQL Server - Add Nickname

Once the details for the remote table are specified, click the **OK** button to add the Nickname to the selection list shown in Figure B-37.

Figure B-37 Microsoft SQL Server - Selection list for creating nicknames

If the **Discover** filter is used to add entries to the Create Nickname window, the default schema will be the user ID that is creating the nicknames. Use the **Schema** button to change the local schema for your Microsoft SQL Server nicknames.

> **Tip:** We recommend that you use the same schema name for all Microsoft SQL nicknames in your federated DB2 database.

Example B-13 shows the command line version of creating the nicknames for your Microsoft SQL instance. Additionally, you may check the nickname definition in the DB2 system catalog with the select statements listed in the example.

Example: B-13 Microsoft SQL Server - Create nickname statements

```
CONNECT TO EDWDB;
CREATE NICKNAME L.NATION FOR MSSQL2000."sqlserv".NATION;
CREATE NICKNAME SQL2000.STORE_SALES_FACT FOR MSSQL2000.DBO.STORE_SALES_FACT;

SELECT * from SYSCAT.TABLES WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.TABOPTIONS WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.COLUMNS WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.COLOPTIONS WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.INDEXES WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.INDEXOPTIONS WHERE TABSCHEMA='SQL2000';
SELECT * from SYSCAT.KEYCOLUSE WHERE TABSCHEMA='SQL2000';
```

Archived

C

Data mapping matrix and code for EDW

This appendix provides the data mapping matrix used to populate the EDW from the staging area.

Archived

© Copyright IBM Corp. 2005. All rights reserved.

Source to target data mapping matrix

Table C-1 shows the source to target data mapping matrix used to consolidate the Oracle and SQL Server data marts into DB2.

Table C-1 Source to Target Data Mapping Matirx

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Calendar	C_dateid_surrogate	Integer				Int	Surrogate key generated by edw
Calendar	C_date	Date	Sales	Calendar	C_date	Small Datetime	Natural key for employee
Calendar	C_year	Smallint	Sales	Calendar	C_year	Smallint	Data type conversion
Calendar	C_quarter	Char(50)	Sales	Calendar	C_quarter	Varchar(50)	Data type conversion
Calendar	C_month	Varchar(100)	Sales	Calendar	C_month	Varchar(50)	Data type conversion
Calendar	C_day	Smallint	Sales	Calendar	C_day	Tinyint	Data type conversion
Calendar	Calendar_date	Date	Sales	Calendar			EDW Table Column
Calendar	Calendar_day	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Calendar_week	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Calendar_month	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Calendar_quarter	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Calendar_year	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Fiscal_date	Date	Sales	Calendar			EDW Table Column
Calendar	Fiscal_day	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Fiscal_week	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Fiscal_month	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Fiscal_quarter	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Fiscal_year	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Season_name	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Holiday_indicator	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Weekday_indicator	Char(10)	Sales	Calendar			EDW Table Column
Calendar	Weekend_indicator	Char(10)	Sales	Calendar			EDW Table Column

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Calendar	Metadata_create_ date	Date	Sales	Calendar			Metadata of edw table
Calendar	Metadata_update_ date	Date	Sales	Calendar			Metadata of edw table
Calendar	Metadata_create_by	Char(10)	Sales	Calendar			Metadata of edw table
Calendar	Metadata_update_by	Char(10)	Sales	Calendar			Metadata of edw table
Calendar	Metadata_effectice_ start_date	Date	Sales	Calendar			Metadata of edw table
Calendar	Metadata_effectice_ end_date	Date	Sales	Calendar			Metadata of edw table
Product	Productkey	Integer				Int	Surrogate key generated by edw
Product	Productid_natural	Varchar(100)	Sales	Product	Productid_ natural	Varchar(100)	Data type conversion
Product	Product name	Varchar(100)	Sales	Product	Productname	Varchar(50)	Data type conversion
Product	Category name	Varchar(100)	Sales	Product	Catergory name	Varchar(50)	Data type conversion
Product	Category desc	Varchar(400)	Sales	Product	Category desc	Varchar(100)	Data type conversion
Product	P_item_status	Char(10)	Sales	Product			EDW Table Column
Product	P_pos_des	Char(10)	Sales	Product			EDW Table Column
Product	P_order_stat_flag	Char(10)	Sales	Product			EDW Table Column
Product	P_hazard_code	Char(10)	Sales	Product			EDW Table Column
Product	P_hazard_status	Char(10)	Sales	Product			EDW Table Column
Product	P_type_diet	Char 10	Sales	Product			EDW Table Column
Product	P_weight	Char(10)	Sales	Product			EDW Table Column
Product	P_width	Char(10)	Sales	Product			EDW Table Column
Product	P_package_size	Char(10)	Sales	Product			EDW Table Column
Product	P_package_type	Char(10)	Sales	Product			EDW Table Column
Product	P_storeage_type	Char(10)	Sales	Product			EDW Table Column
Product	P_product_market	Char(10)	Sales	Product			EDW Table Column
Product	Metadata_create_ date	Date	Sales	Product			Metadata of edw table

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Product	Metadata_update_ date	Date	Sales	Product			Metadata of edw table
Product	Metadata_create_by	Char(10)	Sales	Product			Metadata of edw table
Product	Metadata_update_by	Char(10)	Sales	Product			Metadata of edw table
Product	Metadata_effectice_ start_date	Date	Sales	Product			Metadata of edw table
Product	Metadata_effectice_ end_date	Date	Sales	Product			Metadata of edw table
Vendor	Supplierkey	Integer				Int	Surrogate key generated by edw
Vendor	Supplierid_natural	Integer	Sales	Supplier	Supplierid_ natural	Int	Supplier natural key
Vendor	Companyname	Varchar(100)	Sales	Supplier	Company name	Varchar(50)	Data type conversion
Vendor	Contactname	Varchar(100)	Sales	Supplier	Contactname	Varchar(50)	Data type conversion
Vendor	Contacttitle	Varchar(100)	Sales	Supplier	Contacttitle	Varchar(50)	Data type conversion
Vendor	Address	Varchar(100)	Sales	Supplier	Address	Varchar(50)	Data type conversion
Vendor	City	Varchar(100)	Sales	Supplier	City	Varchar(50)	Data type conversion
Vendor	Region	Varchar(100)	Sales	Supplier	Region	Varchar(50)	Data type conversion
Vendor	Postalcode	Varchar(100)	Sales	Supplier	Postalcode	Varchar(50)	Data type conversion
Vendor	Country	Varchar(100)	Sales	Supplier	Country	Varchar(50)	Data type conversion
Vendor	Phone	Varchar(100)	Sales	Supplier	Phone	Varchar(50)	Data type conversion
Vendor	Fax	Varchar(100)	Sales	Supplier	Fax	Varchar(50)	Data type conversion
Vendor	Metadata_create_ date	Date	Sales				Metadata of edw table
Vendor	Metadata_update_ date	Date	Sales				Metadata of edw table
Vendor	Metadata_create_by	Char(10)	Sales				Metadata of edw table
Vendor	Metadata_update_by	Char(10)	Sales				Metadata of edw table
Vendor	Metadata_effectice_ start_date	Date	Sales				Metadata of edw table
Vendor	Metadata_effectice_ end_date	Date	Sales				Metadata of edw table
Employee	Employeekey	Integer				Int	Surrogate key generated by edw

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Employee	Employeeid_natural	Integer	Sales	Employee	Employeeid_ natural	Int	Natural key for employee
Employee	Reports_to_id	Integer	Sales	Employee	Reports_to_id	Int	Data type conversion
Employee	Fullname	Varchar(100)	Sales	Employee	Lastname	Varchar(50)	Data type conversion
Employee	Lastname	Varchar(100)	Sales	Employee	Lastname	Varchar(50)	Data type conversion
Employee	Firstname	Varchar(100)	Sales	Employee	Firstname	Varchar(50)	Data type conversion
Employee	Managername	Varchar(100)	Sales	Employee	Managername	Varchar(50)	Data type conversion
Employee	Dob	Date	Sales	Employee	Dob	Datetime	Data type conversion
Employee	Hiredate	Date	Sales	Employee	Hiredate	Datetime	Data type conversion
Employee	Address	Varchar(100)	Sales	Employee	Address	Varchar(60)	Data type conversion
Employee	City	Varchar(80)	Sales	Employee	City	Varchar(50)	Data type conversion
Employee	Region	Varchar(80)	Sales	Employee	Region	Varchar(50)	Data type conversion
Employee	Postalcode	Varchar(80)	Sales	Employee	Postalcode	Varchar(50)	Data type conversion
Employee	Country	Varchar(90)	Sales	Employee	Country	Varchar(50)	Data type conversion
Employee	Homephone	Varchar(90)	Sales	Employee	Homephone	Varchar(50)	Data type conversion
Employee	Extension	Varchar(90)	Sales	Employee	Extension	Varchar(50)	Data type conversion
Employee	Metadata_create_ date	Date	Sales				Metadata of edw table
Employee	Metadata_update_ date	Date	Sales				Metadata of edw table
Employee	Metadata_create_by	Char(10)	Sales				Metadata of edw table
Employee	Metadata_update_by	Char(10)	Sales				Metadata of edw table
Employee	Metadata_effective_s tart_date	Date	Sales				Metadata of edw table
Employee	Metadata_effective_e nd_date	Date	Sales				Metadata of edw table
Customer	Customerkey	Integer				Int	Surrogate key generated by edw
Customer	Customerid_natural	Varchar(100)	Sales	Customer	Customerid_ natural	Varchar(100)	Customer natural id
Customer	Customer_category	Varchar(100)	Sales	Customer_ category	Customer_ category	Varchar(100)	Two snowflaked, customer and customer_category are merged into 1 table

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Customer	Companyname	Varchar(100)	Sales	Customer	Companyname	Varchar(100)	Data type conversion
Customer	Contactname	Varchar(100)	Sales	Customer	Contactname	Varchar(100)	Data type conversion
Customer	Address	Varchar(100)	Sales	Customer	Address	Varchar(100)	Data type conversion
Customer	City	Varchar(100)	Sales	Customer	City	Varchar(100)	Data type conversion
Customer	Region	Varchar(100)	Sales	Customer	Region	Varchar(100)	Data type conversion
Customer	Postalcode	Varchar(100)	Sales	Customer	Postalcode	Varchar(100)	Data type conversion
Customer	Country	Varchar(100)	Sales	Customer	Country	Varchar(100)	Data type conversion
Customer	Phone	Varchar(100)	Sales	Customer	Phone	Varchar(100)	Data type conversion
Customer	Fax	Varchar(100)		Customer	Fax		Data type conversion
Customer	Metadata_create_date	Date	Sales				Metadata of edw table
Customer	Metadata_update_date	Date	Sales				Metadata of edw table
Customer	Metadata_create_by	Char(10)	Sales				Metadata of edw table
Customer	Metadata_update_by	Char(10)	Sales				Metadata of edw table
Customer	Metadata_effective_start_date	Date	Sales				Metadata of edw table
Customer	Metadata_effective_start_date	Date	Sales				Metadata of edw table
Stores	Stor_id	Int				Int	Surrogate key generated by edw
Stores	Stor_name	Char(40)	Sales	Stores	Store_name	Varchar(50)	Data type conversion
Stores	Stor_address	Char(40)	Sales	Stores	Store_address	Varchar(100)	Data type conversion
Stores	City	Char(20)	Sales	Stores	City	Varchar(50)	Data type conversion
Stores	State	Char(2)	Sales	Stores	State	Char(20)	Data type conversion
Stores	Zip	Char(5)	Sales	Stores	Zip	Char(50)	Data type conversion
Stores	Store_category	Varchar(100)	Sales	Store_category	Store_category	Char(50)	Two snowflaked, store and store_category are merged into 1 table
Stores	Metadata_create_date	Date	Sales				Metadata of edw table
Stores	Metadata_update_date	Date	Sales				Metadata of edw table

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Stores	Metadata_create_by	Char(10)	Sales				Metadata of edw table
Stores	Metadata_update_by	Char(10)	Sales				Metadata of edw table
Stores	Metadata_effective_ start_date	Date	Sales				Metadata of edw table
Stores	Metadata_effective_ start_date	Date	Sales				Metadata of edw table
Edw_Sales_ fact	Productkey	Integer	Sales	Store_sales_ fact	Productkey	Int	Surrogate key generated by edw
Edw_Sales_ fact	Employeekey	Integer	Sales	Store_ sales_fact	Employeekey	Int	Surrogate key generated by edw
Edw_Sales_ fact	Customerkey	Integer	Sales	Store_ sales_fact	Customerkey	Int	Surrogate key generated by edw
Edw_Sales_ fact	Supplierkey	Integer	Sales	Store_ sales_fact	Supplierkey	Int	Surrogate key generated by edw
Edw_Sales_ fact	Dateid	Integer	Sales	Store_ sales_fact	Calendar_id	Int	Surrogate key generated by edw
Edw_Sales_ fact	Postransno	Integer	Sales	Store_ sales_fact	Postransno	Int	Point of sales transaction number
Edw_Sales_ fact	Salesqty	Integer	Sales	Store_ sales_fact	Salesqty	Int	Quantity of sale of product for a postransno
Edw_Sales_ fact	Unitprice	Decimal (19,4)	Sales	Store_ sales_fact	Unitprice	Money	Unit price of product for a postransno
Edw_Sales_ fact	Salesprice	Decimal (19,4)	Sales	Store_ sales_fact	Salesprice	Money	Sales price of product for a postransno
Edw_Sales_ fact	Discount	Decimal (19,4)	Sales	Store_ sales_fact	Discount	Money	Discount price of product for a postransno
Edw_Sales_ fact	Storeid	Integer	Sales	Store_ sales_fact	Storeid	Int	Surrogate key generated by edw
Calendar	C_dateid_surrogate	Integer					Surrogate key generated by edw
Calendar	C_date	Date	Inventory	Calendar	C_date	Date	Natural key for employee
Calendar	C_year	Smallint	Inventory	Calendar	C_year	Number(5)	Simple oracle to db2 data type conversion
Calendar	C_quarter	Char(50)	Inventory	Calendar	C_quarter	Char (10 byte)	Simple oracle to db2 data type conversion
Calendar	C_month	Varchar(100)	Inventory	Calendar	C_month	Varchar2 (100 byte)	Simple oracle to db2 data type conversion

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Calendar	C_day	Smallint	Inventory	Calendar	C_day	Number(3)	Simple oracle to db2 data type conversion
Calendar	Calendar_date	Date	Inventory	Calendar			EDW Table Column
Calendar	Calendar_day	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Calendar_week	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Calendar_month	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Calendar_quarter	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Calendar_year	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_date	Date	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_day	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_week	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_month	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_quarter	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Fiscal_year	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Season_name	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Holiday_indicator	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Weekday_indicator	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Weekend_indicator	Char(10)	Inventory	Calendar			EDW Table Column
Calendar	Metadata_create_ date	Date	Inventory	Calendar			Metadata of edw table
Calendar	Metadata_update_ date	Date	Inventory	Calendar			Metadata of edw table
Calendar	Metadata_create_by	Char(10)	Inventory	Calendar			Metadata of edw table
Calendar	Metadata_update_by	Char(10)	Inventory	Calendar			Metadata of edw table
Calendar	Metadata_effectice_ start_date	Date	Inventory	Calendar			Metadata of edw table
Calendar	Metadata_effectice_ end_date	Date	Inventory	Calendar			Metadata of edw table
Product	Productkey	Integer	Inventory				Surrogate key generated by edw
Product	Productid_natural	Varchar(100)	Inventory	Product	Productid_ natural	Varchar2 (50 byte)	Simple oracle to db2 data type conversion

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Product	Productname	Varchar(100)	Inventory	Product	Productname	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Product	Catergoryname	Varchar(100)	Inventory	Product	Catergory name	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Product	Categorydesc	Varchar(400)	Inventory	Product	Categorydesc	Varchar2 (100 byte)	Simple oracle to db2 data type conversion
Product	P_item_status	Char(10)	Inventory	Product			EDW Table Column
Product	P_pos_des	Char(10)	Inventory	Product			EDW Table Column
Product	P_order_stat_flag	Char(10)	Inventory	Product			EDW Table Column
Product	P_hazard_code	Char(10)	Inventory	Product			EDW Table Column
Product	P_hazard_status	Char(10)	Inventory	Product			EDW Table Column
Product	P_type_diet	Char(10)	Inventory	Product			EDW Table Column
Product	P_weight	Char(10)	Inventory	Product			EDW Table Column
Product	P_width	Char(10)	Inventory	Product			EDW Table Column
Product	P_package_size	Char(10)	Inventory	Product			EDW Table Column
Product	P_package_type	Char(10)	Inventory	Product			EDW Table Column
Product	P_storeage_type	Char(10)	Inventory	Product			EDW Table Column
Product	P_product_market	Char(10)	Inventory	Product			EDW Table Column
Product	Metadata_create_ date	Date	Inventory	Product			Metadata of edw table
Product	Metadata_update_ date	Date	Inventory	Product			Metadata of edw table
Product	Metadata_create_by	Char(10)	Inventory	Product			Metadata of edw table
Product	Metadata_update_by	Char(10)	Inventory	Product			Metadata of edw table
Product	Metadata_effectice_ start_date	Date	Inventory	Product			Metadata of edw table
Product	Metadata_effectice_ end_date	Date	Inventory	Product			Metadata of edw table
Vendor	Supplierkey	Integer	Inventory				Surrogate key generated by edw
Vendor	Supplierid_natural	Integer	Inventory	Supplier	Supplierid_ natural	Number(10)	Supplier natural key
Vendor	Companyname	Varchar(100)	Inventory	Supplier	Company name	Varchar2 (50 byte)	Simple oracle to db2 data type conversion

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Vendor	Contactname	Varchar(100)	Inventory	Supplier	Contactname	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Contacttitle	Varchar(100)	Inventory	Supplier	Contacttitle	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Address	Varchar(100)	Inventory	Supplier	Address	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	City	Varchar(100)	Inventory	Supplier	City	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Region	Varchar(100)	Inventory	Supplier	Region	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Postalcode	Varchar(100)	Inventory	Supplier	Postalcode	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Country	Varchar(100)	Inventory	Supplier	Country	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Phone	Varchar(100)	Inventory	Supplier	Phone	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Fax	Varchar(100)	Inventory	Supplier	Fax	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Vendor	Metadata_create_ date	Date	Inventory				Metadata of edw table
Vendor	Metadata_update_ date	Date	Inventory				Metadata of edw table
Vendor	Metadata_create_by	Char(10)	Inventory				Metadata of edw table
Vendor	Metadata_update_by	Char(10)	Inventory				Metadata of edw table
Vendor	Metadata_effectice_ start_date	Date	Inventory				Metadata of edw table
Vendor	Metadata_effectice_ end_date	Date	Inventory				Metadata of edw table
Stores	Stor_id	Int	Inventory				Surrogate key generated by edw
Stores	Stor_name	Char(40)	Inventory	Stores	Store_name	Varchar2 (40 byte)	Simple oracle to db2 data type conversion
Stores	Stor_address	Char(40)	Inventory	Stores	Store_ address	Varchar2 (40 byte)	Simple oracle to db2 data type conversion
Stores	City	Char(20)	Inventory	Stores	City	Varchar2 (40 byte)	Simple oracle to db2 data type conversion
Stores	State	Char(2)	Inventory	Stores	State	Varchar2 (40 byte)	Simple oracle to db2 data type conversion

Archived

Table name	Column name	Data type	Data Mart Name	Table name	Column	Data type	(conversion and metadata)
Stores	Zip	Char(5)	Inventory	Stores	Zip	Varchar2 (50 byte)	Simple oracle to db2 data type conversion
Stores	Store_category	Varchar (100)	Inventory	Store_ category	Store_ category	Number(10)	Two snowflaked, store and store_category are merged into 1 table
Stores	Metadata_create_ date	Date	Inventory				Metadata of edw table
Stores	Metadata_update_ date	Date	Inventory				Metadata of edw table
Stores	Metadata_create_by	Char(10)	Inventory				Metadata of edw table
Stores	Metadata_update_by	Char(10)	Inventory				Metadata of edw table
Stores	Metadata_effective_ start_date	Date	Inventory				Metadata of edw table
Stores	Metadata_effective_ start_date	Date	Inventory				Metadata of edw table
Edw_inventory _fact	Store_id	Integer	Inventory	Store_ inventory_ fact	Storeid	Number(10)	Matches surrogate key generated by edw for store_id
Edw_inventory _fact	Product_id	Integer	Inventory	Store_ inventory_ fact	Product_id	Number(10)	Matches surrogate key generated by edw for product_id
Edw_inventory _fact	Date_id	Integer	Inventory	Store_ inventory_ fact	Calendar_id	Number(10)	Matches surrogate key generated by edw for calendar_id
Edw_inventory _fact	Supplier_id	Integer	Inventory	Store_ inventory_ fact	Supplier_id	Number(10)	Matches surrogate key generated by edw for supplier_id

Archived

SQL ETL Code to populate the EDW

We use DB2 SQL code for the ETL process to populate to EDW from the staging area. Sample ETL code is depicted in Example C-1.

Example: C-1 DB2 SQL ETL code

```
INSERT INTO EDW.CALENDAR
(
C_DATEID_SURROGATE,
C_DATE,
C_YEAR,
C_QUARTER,
C_MONTH,C_DAY
)
SELECT
CALENDAR_ID,
DATE(C_DATE),
C_YEAR,
C_QUARTER,
C_MONTH,C_DAY
FROM STAGESQL1.CALENDAR
ORDER BY CALENDAR_ID
;
INSERT INTO EDW.CUSTOMER
(
CUSTOMERKEY,
CUSTOMER_CATEGORY,
CUSTOMERID_NATURAL,
COMPANYNAME,
CONTACTNAME,
ADDRESS,
CITY,
REGION,
POSTALCODE,
COUNTRY,
PHONE,
FAX
)
SELECT
CUSTOMERKEY,
CUSTOMER_CATEGORY,
CUSTOMERID_NATURAL,
COMPANYNAME,
CONTACTNAME,
ADDRESS,
CITY,
REGION,
POSTALCODE,
```

Archived

```
COUNTRY,
PHONE,
FAX
FROM
STAGESQL1.CUSTOMER A,
STAGESQL1.CUSTOMER_CATEGORY B
WHERE
A.CUSTOMER_CATG_ID=B.CUSTOMER_CATG_ID
ORDER BY CUSTOMERKEY
;
INSERT INTO EDW.EMPLOYEE
(
EMPLOYEEKEY,
EMPLOYEEID_NATURAL,
REPORTS_TO_ID,FULLNAME,
LASTNAME,
FIRSTNAME,
MANAGERNAME,
DOB,
HIREDATE,
ADDRESS,
CITY,
REGION,
POSTALCODE,
COUNTRY,
HOMEPHONE,
EXTENSION
)
SELECT
EMPLOYEEKEY,
EMPLOYEEID_NATURAL,
REPORTS_TO_ID,
FULL_NAME,
LASTNAME,
FIRSTNAME,
MANAGER_NAME,
DATE(DOB),
DATE(HIREDATE),
ADDRESS,
CITY,
REGION,
POSTALCODE,
COUNTRY,
HOMEPHONE,
EXTENSION
FROM STAGESQL1.EMPLOYEE
ORDER BY EMPLOYEEKEY
;
INSERT INTO EDW.PRODUCT
```

Archived

```
(
PRODUCTKEY,
PRODUCTID_NATURAL,
PRODUCTNAME,
CATERGORYNAME,
CATEGORYDESC
)
SELECT
PRODUCTKEY,
PRODUCTID_NATURAL,
PRODUCTNAME,
CATERGORYNAME,
CATEGORYDESC
FROM STAGESQL1.PRODUCT
ORDER BY PRODUCTKEY
;
INSERT INTO EDW.STORES
(
STOR_ID,
STOR_NAME,
STOR_ADDRESS,
CITY,
STATE,
ZIP,
STORE_CATEGORY
)
SELECT
STORE_ID,
STORE_NAME,
STORE_ADDRESS,
SUBSTR(CITY,1,20),
SUBSTR(STATE,1,2),
SUBSTR(ZIP,1,5),
CHAR(INT(STORE_CATALOG_ID))
FROM STAGEORA1.STORES
ORDER BY STORE_ID;
INSERT INTO EDW.VENDOR
(
SUPPLIERKEY,
SUPPLIERID_NATURAL,
COMPANYNAME,
CONTACTNAME,
CONTACTTITLE,
ADDRESS,
CITY,
REGION,
POSTALCODE,
COUNTRY,
PHONE,
```

Archived

```
FAX
)
SELECT
SUPPLIERKEY,
SUPPLIERID_NATURAL,
COMPANY_NAME,
CONTACT_NAME,
CONTACT_TITLE,
ADDRESS,
CITY,
REGION,
POSTALCODE,
COUNTRY,
PHONE,
FAX
FROM STAGESQL1.SUPPLIER
ORDER BY SUPPLIERKEY
;
INSERT INTO EDW.EDW_INVENTORY_FACT
(
STORE_ID,
PRODUCT_ID,
DATE_ID,
SUPPLIER_ID,
QUANTITY_IN_INVENTORY
)
SELECT
A.STORE_ID,
B.PRODUCTKEY,
C.CALENDAR_ID,
D.SUPPLIERKEY,
QUANTITY_IN_INVENTORY
FROM STAGEORA1.STORES A,
STAGEORA1.PRODUCT B,
STAGEORA1.CALENDAR C,
STAGEORA1.SUPPLIER D ,
STAGEORA1.STORE_INVENTORY_FACT E
WHERE A.STORE_ID=E.STORE_ID
AND B.PRODUCTKEY=E.PRODUCT_ID
AND C.CALENDAR_ID=E.CALENDAR_ID
AND D.SUPPLIERKEY=E.SUPPLIER_ID
;
INSERT INTO EDW.EDW_SALES_FACT
(
DATEID,
CUSTOMERKEY,
EMPLOYEEKEY,
PRODUCTKEY,
STOREID,
```

Archived

```
SUPPLIERKEY,
POSTRANSNO,
SALESQTY,
UNITPRICE,
SALESPRICE,
DISCOUNT
)
SELECT
B.CALENDAR_ID,
C.CUSTOMERKEY,
D.EMPLOYEEKEY,
E.PRODUCTKEY,
F.STORE_ID,
G.SUPPLIERKEY,
A.POS_TRANSNO,
A.SALESQTY,
A.UNITPRICE,
A.SALESPRICE,
A.DISCOUNT
FROM
STAGESQL1.STORE_SALES_FACT A,
STAGESQL1.CALENDAR B,
STAGESQL1.CUSTOMER C,
STAGESQL1.EMPLOYEE D,
STAGESQL1.PRODUCT E,
STAGESQL1.STORES F,
STAGESQL1.SUPPLIER G
WHERE
B.CALENDAR_ID=A.DATEID
AND C.CUSTOMERKEY=A.CUSTOMERKEY
AND D.EMPLOYEEKEY=A.EMPLOYEEKEY
AND E.PRODUCTKEY=A.PRODUCTKEY
AND F.STORE_ID=A.STOREID
AND G.SUPPLIERKEY=A.SUPPLIERKEY
;
```

Archived

D

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246653

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246653.

© Copyright IBM Corp. 2005. All rights reserved.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
DMC-SAMPLE.ZIP	Zipped DDL and CSV to re-create our DMC Sample Scenario
DMC-README.TXT	Text instructions for implementing the Sample Scenario

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Instructions for implementing the Sample Environment are contained in the DMC-README.TXT file.

You will required the following DBMSs to implement the Sample Scenario:

1. Oracle 9i Database Server
2. Microsoft SQL2000 Database Server
3. DB2 UDB ESE Version 8.2

Archived

Abbreviations and acronyms

ACS	access control system
ADK	Archive Development Kit
AIX	Advanced Interactive eXecutive from IBM
API	Application Programming Interface
AQR	automatic query re-write
AR	access register
ARM	automatic restart manager
ART	access register translation
ASCII	American Standard Code for Information Interchange
AST	Application Summary Table
BLOB	Binary Large OBject
BW	Business Information Warehouse (SAP)
CCMS	Computing Center Management System
CFG	Configuration
CLI	Call Level Interface
CLOB	Character Large OBject
CLP	Command Line Processor
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CS	Cursor Stability
DAS	DB2 Administration Server
DB	Database
DB2	Database 2™
DB2 UDB	DB2 Universal DataBase
DBA	Database Administrator
DBM	DataBase Manager
DBMS	DataBase Management System
DCE	Distributed Computing Environment
DCM	Dynamic Coserver Management
DCOM	Distributed Component Object Model
DDL	Data Definition Language - a SQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE.
DES	Data Encryption Standard
DIMID	Dimension Identifier
DLL	Dynamically Linked Library
DML	Data Manipulation Language - an INSERT, UPDATE, DELETE, or SELECT SQL statement.
DMS	Database Managed Space
DPF	Data Partitioning Facility
DRDA®	Distributed Relational Database Architecture™
DSA	Dynamic Scalable Architecture
DSN	Data Source Name
DSS	Decision Support System
EAI	Enterprise Application Integration
EBCDIC	Extended Binary Coded Decimal Interchange Code
EDA	Enterprise Data Architecture
EDU	Engine Dispatchable Unit
EDW	Enterprise Data Warehouse
EGM	Enterprise Gateway Manager
EJB	Enterprise Java Beans

Archived

© Copyright IBM Corp. 2005. All rights reserved.

ER	Enterprise Replication
ERP	Enterprise Resource Planning
ESE	Enterprise Server Edition
ETL	Extract, Transform, and Load
ETTL	Extract, Transform/Transport, and Load
FP	Fix Pack
FTP	File Transfer Protocol
Gb	Giga bits
GB	Giga Bytes
GUI	Graphical User Interface
HADR	High Availability Disaster Recovery
HDR	High availability Data Replication
HPL	High Performance Loader
I/O	Input/Output
IBM	International Business Machines Corporation
ID	Identifier
IDE	Integrated Development Environment
IDS	Informix Dynamic Server
II	Information Integrator
IMG	Integrated Implementation Guide (for SAP)
IMS	Information Management System
ISAM	Indexed Sequential Access Method
ISM	Informix Storage Manager
ISV	Independent Software Vendor
IT	Information Technology
ITR	Internal Throughput Rate
ITSO	International Technical Support Organization
IX	Index
J2EE	Java 2 Platform Enterprise Edition
JAR	Java Archive
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JE	Java Edition
JMS	Java Message Service
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KB	Kilobyte (1024 bytes)
LDAP	Lightweight Directory Access Protocol
LPAR	Logical Partition
LV	Logical Volume
Mb	Mega bits
MB	Mega Bytes
MDC	Multidimensional Clustering
MPP	Massively Parallel Processing
MQI	Message Queuing Interface
MQT	Materialized Query Table
MRM	Message Repository Manager
MTK	DB2 Migration ToolKit for Informix
NPI	Non-Partitioning Index
ODBC	Open DataBase Connectivity
ODS	Operational Data Store
OLAP	OnLine Analytical Processing
OLE	Object Linking and Embedding
OLTP	OnLine Transaction Processing
ORDBMS	Object Relational DataBase Management System
OS	Operating System
O/S	Operating System
PDS	Partitioned Data Set

Archived

PIB	Parallel Index Build
PSA	Persistent Staging Area
RBA	Relative Byte Address
RBW	Red Brick™ Warehouse
RDBMS	Relational DataBase Management System
RID	Record Identifier
RR	Repeatable Read
RS	Read Stability
SCB	Session Control Block
SDK	Software Developers Kit
SID	Surrogage Identifier
SMIT	Systems Management Interface Tool
SMP	Symmetric MultiProcessing
SMS	System Managed Space
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPL	Stored Procedure Language
SQL	Structured Query
TCB	Thread Control Block
TMU	Table Management Utility
TS	Tablespace
UDB	Universal DataBase
UDF	User Defined Function
UDR	User Defined Routine
URL	Uniform Resource Locator
VG	Volume Group (Raid disk terminology).
VLDB	Very Large DataBase
VP	Virtual Processor
VSAM	Virtual Sequential Access Method
VTI	Virtual Table Interface
WSDL	Web Services Definition Language
WWW	World Wide Web
XBSA	X-Open Backup and Restore APIs
XML	eXtensible Markup Language
XPS	Informix eXtended Parallel Server

Archived

Archived

Glossary

Access Control List (ACL). The list of principals that have explicit permission (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

Aggregate. Pre-calculated and pre-stored summaries, kept in the data warehouse to improve query performance

Aggregation. An attribute level transformation that reduces the level of detail of available data. For example, having a Total Quantity by Category of Items rather than the individual quantity of each item in the category.

Analytic. An application or capability that performs some analysis on a set of data.

Application Programming Interface. An interface provided by a software product that enables programs to request services.

Asynchronous Messaging. A method of communication between programs in which a program places a message on a message queue, then proceeds with its own processing without waiting for a reply to its message.

Attribute. A field in a dimension table/

BLOB. Binary Large Object, a block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted.

Commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins.

Compensation. The ability of DB2 to process SQL that is not supported by a data source on the data from that data source.

Composite Key. A key in a fact table that is the concatenation of the foreign keys in the dimension tables.

Computer. A device that accepts information (in the form of digitalized data) and manipulates it for some result based on a program or sequence of instructions on how the data is to be processed.

Configuration. The collection of brokers, their execution groups, the message flows and sets that are assigned to them, and the topics and associated access control specifications.

Connector. See Message processing node connector.

DDL (Data Definition Language)**.** a SQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE DATABASE.

DML (Data Manipulation Language)**.** an INSERT, UPDATE, DELETE, or SELECT SQL statement.

Data Append. A data loading technique where new data is added to the database leaving the existing data unaltered.

Data Append. A data loading technique where new data is added to the database leaving the existing data unaltered.

Data Cleansing. A process of data manipulation and transformation to eliminate variations and inconsistencies in data content. This is typically to improve the quality, consistency, and usability of the data.

© Copyright IBM Corp. 2005. All rights reserved.

Data Federation. The process of enabling data from multiple heterogeneous data sources to appear as if it is contained in a single relational database. Can also be referred to “distributed access”.

Data mart. An implementation of a data warehouse, typically with a smaller and more tightly restricted scope - such as for a department, workgroup, or subject area. It could be independent, or derived from another data warehouse environment (dependent).

Data mart - Dependent. A data mart that is consistent with, and extracts its data from, a data warehouse.

Data mart - Independent. A data mart that is standalone, and does not conform with any other data mart or data warehouse.

Data Mining. A mode of data analysis that has a focus on the discovery of new information, such as unknown facts, data relationships, or data patterns.

Data Partition. A segment of a database that can be accessed and operated on independently even though it is part of a larger data structure.

Data Refresh. A data loading technique where all the data in a database is completely replaced with a new set of data.

Data silo. A standalone set of data in a particular department or organization used for analysis, but typically not shared with other departments or organizations in the enterprise.

Data Warehouse. A specialized data environment developed, structured, shared, and used specifically for decision support and informational (analytic) applications. It is subject oriented rather than application oriented, and is integrated, non-volatile, and time variant.

Database Instance. A specific independent implementation of a DBMS in a specific environment. For example, there might be an independent DB2 DBMS implementation on a Linux server in Boston supporting the Eastern offices, and another separate and independent DB2 DBMS on the same Linux server supporting the western offices. They would represent two instances of DB2.

Database Partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

DataBlades. These are program modules that provide extended capabilities for Informix databases, and are tightly integrated with the DBMS.

DB Connect. Enables connection to several relational database systems and the transfer of data from these database systems into the SAP Business Information Warehouse.

Debugger. A facility on the Message Flows view in the Control Center that enables message flows to be visually debugged.

Deploy. Make operational the configuration and topology of the broker domain.

Dimension. Data that further qualifies and/or describes a measure, such as amounts or durations.

Distributed Application In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

Drill-down. Iterative analysis, exploring facts at more detailed levels of the dimension hierarchies.

Dynamic SQL. SQL that is interpreted during execution of the statement.

Engine. A program that performs a core or essential function for other programs. A database engine performs database functions on behalf of the database user programs.

Archived

Enrichment. The creation of derived data. An attribute level transformation performed by some type of algorithm to create one or more new (derived) attributes.

Extenders. These are program modules that provide extended capabilities for DB2, and are tightly integrated with DB2.

FACTS. A collection of measures, and the information to interpret those measures in a given context.

Federated data. A set of physically separate data structures that are logically linked together by some mechanism, for analysis, but which remain physically in place.

Federated Server. Any DB2 server where the WebSphere Information Integrator is installed.

Federation. Providing a unified interface to diverse data.

Gateway. A means to access a heterogeneous data source. It can use native access or ODBC technology.

Grain. The fundamental lowest level of data represented in a dimensional fact table.

Instance. A particular realization of a computer process. Relative to database, the realization of a complete database environment.

Java Database Connectivity. An application programming interface that has the same characteristics as ODBC but is specifically designed for use by Java database applications.

Java Development Kit. Software package used to write, compile, debug and run Java applets and applications.

Java Message Service. An application programming interface that provides Java language functions for handling messages.

Java Runtime Environment. A subset of the Java Development Kit that allows you to run Java applets and applications.

Materialized Query Table. A table where the results of a query are stored, for later reuse.

Measure. A data item that measures the performance or behavior of business processes.

Message domain. The value that determines how the message is interpreted (parsed).

Message flow. A directed graph that represents the set of activities performed on a message or event as it passes through a broker. A message flow consists of a set of message processing nodes and message processing connectors.

Message parser. A program that interprets the bit stream of an incoming message and creates an internal representation of the message in a tree structure. A parser is also responsible to generate a bit stream for an outgoing message from the internal representation.

Meta Data. Typically called data (or information) about data. It describes or defines data elements.

MOLAP. Multi-dimensional OLAP. Can be called MD-OLAP. It is OLAP that uses a multi-dimensional database as the underlying data structure.

Multi-dimensional analysis. Analysis of data along several dimensions. For example, analyzing revenue by product, store, and date.

Multi-Tasking. Operating system capability which allows multiple tasks to run concurrently, taking turns using the resources of the computer.

Multi-Threading. Operating system capability that enables multiple concurrent users to use the same program. This saves the overhead of initiating the program multiple times.

Nickname. An identifier that is used to reference the object located at the data source that you want to access.

Node Group. Group of one or more database partitions.

Node. See Message processing node and Plug-in node.

ODS. (1) Operational data store: A relational table for holding clean data to load into InfoCubes, and can support some query activity. (2) Online Dynamic Server - an older name for IDS.

OLAP. OnLine Analytical Processing. Multi-dimensional data analysis, performed in real-time. Not dependent on underlying data schema.

Open Database Connectivity. A standard application programming interface for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group.

Optimization. The capability to enable a process to execute and perform in such a way as to maximize performance, minimize resource utilization, and minimize the process execution response time delivered to the end user.

Partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

Pass-through. The act of passing the SQL for an operation directly to the data source without being changed by the federation server.

Pivoting. Analysis operation where user takes a different viewpoint of the results. For example, by changing the way the dimensions are arranged.

Primary Key. Field in a table that is uniquely different for each record in the table.

Process. An instance of a program running in a computer.

Program. A specific set of ordered operations for a computer to perform.

Pushdown. The act of optimizing a data operation by pushing the SQL down to the lowest point in the federated architecture where that operation can be executed. More simply, a pushdown operation is one that is executed at a remote server.

ROLAP. Relational OLAP. Multi-dimensional analysis using a multi-dimensional view of relational data. A relational database is used as the underlying data structure.

Roll-up. Iterative analysis, exploring facts at a higher level of summarization.

Server. A computer program that provides services to other computer programs (and their users) in the same or other computers. However, the computer that a server program runs in is also frequently referred to as a server.

Shared nothing. A data management architecture where nothing is shared between processes. Each process has its own processor, memory, and disk space.

Spreadmart. A standalone, non-conforming, non-integrated set of data, such as a spreadsheet, used for analysis by a particular person, department, or organization.

Static SQL. SQL that has been compiled prior to execution. Typically provides best performance.

Static SQL. SQL that has been compiled prior to execution. Typically provides best performance.

Subject Area. A logical grouping of data by categories, such as customers or items.

Archived

Synchronous Messaging. A method of communication between programs in which a program places a message on a message queue and then waits for a reply before resuming its own processing.

Task. The basic unit of programming that an operating system controls. Also see Multi-Tasking.

Thread. The placeholder information associated with a single use of a program that can handle multiple concurrent users. Also see Multi-Threading.

Type Mapping. The mapping of a specific data source type to a DB2 UDB data type

Unit of Work. A recoverable sequence of operations performed by an application between two points of consistency.

User Mapping. An association made between the federated server user ID and password and the data source (to be accessed) used ID and password.

Virtual Database. A federation of multiple heterogeneous relational databases.

Warehouse Catalog. A subsystem that stores and manages all the system metadata.

Wrapper. The means by which a data federation engine interacts with heterogeneous sources of data. Wrappers take the SQL that the federation engine uses and maps it to the API of the data source to be accessed. For example, they take DB2 SQL and transform it to the language understood by the data source to be accessed.

xtree. A query-tree tool that allows you to monitor the query plan execution of individual queries in a graphical environment.

Archived

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 394. Note that some of the documents referenced here may be available in softcopy only.

- *Oracle to DB2 UDB Conversion Guide,* SG24-7048.
- *DB2 UDB ESE V8 non-DPF Performance Guide for High Performance OLTP and BI,* SG24-6432
- *DB2 UDB’s High Function Business Intelligence in e-business*, SG24-6546.
- *Moving Data Across the DB2 Family,* SG24-6905.
- *Preparing for DB2 Near-Realtime Business Intelligence*, SG24-6071.
- *DB2 Cube Views: A Primer,* SG24-7002
- *Up and Running with DB2 UDB ESE: Partitioning for Performance in an e-Business Intelligence World*, SG24-6917.
- *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012.
- *Virtualization and the On Demand Business*, REDP-9115.
- XML for DB2 Information Integration, SG24- 6994.

Other publications

These publications are also relevant as further information sources:

- *DB2 UDB Administration Guide: Performance, SC09- 4821.*
- *IBM DB2 High Performance Unload for Multiplatforms and Workgroup - User’s Guide, SC27-1623*.

Archived

© Copyright IBM Corp. 2005. All rights reserved.

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

A
abstraction layer 34
administration procedures 196
AitinSoft 206
AIX 205
analysis - multi-dimensional 27
Analytic Services 132
analytic structures 4, 23, 93, 151, 168, 179
  data quality 154
  evaluating 152
  redundant data 160
  spreadsheets 29
  tools 166
Analytical Intelligence 168
API 36
application conversion 214
Application development 55
application development costs 263
application logic 214
application programming interfaces 217
application testing 215
application tuning 216
approaches to consolidation 72
architecting a data mart 22
architecture profile 214
ASCII formats 254
assessment phase 151, 168, 257
attribute change-handling strategies 156
automatic commits 248

B
backup and recovery 1, 195
backup procedure 293
Basel II 5, 53
Batch/Script processing 196
benefits of consolidation 5
BI 235, 239
BI Reference Architecture 17
block index 239, 252
block-based indexes 26
BM WebSphere Information Integrator V8.2 315
Borland Together 207
buffer pool objects 230
buffer pools 229
Build SQL 146
business case for consolidation 54
business definitions and rules 185
Business Intelligence xi, 11, 59
  also see BI xi
business metadata 74, 162, 186, 277, 291
Business Objects 126
business performance management 17
Business Project Leader 182
business rules 155
Business Subject Area Specialist 182

C
CA AllFusion ERwin Data Modeler 207
Calendar Dimension 269
candidate dimension keys 239
candidate dimensions 239
candidates for DMC 65
cell density 237, 239
cell utilization 239
Centralized consolidation 72, 179, 274
  advantages 81
  merge with primary 76, 184
  redesign 184
  using redesign 76
CLP 252
clustering dimensions 26
clustering index 234
Cognos 126
Column Mapping 146
command line processor 247, 252
common data elements 275
common data model 263
composite block index 239
conformed dimension - definition 188
conformed dimension mapping 288
conformed dimensions 77, 188, 277, 288
conformed facts 77, 188
conforming 187
consolidated reporting environment 98
consolidating data marts 276
Consolidating spreadsheet data 117

© Copyright IBM Corp. 2005. All rights reserved.

consolidating spreadsheets 121
techniques 121
consolidation approach 274
consolidation areas 180
consolidation benefits 5
consolidation lifecycle 69, 149
consolidation process 274
consolidation source systems 161
conversion 317
Convert 319
Convert task 317
Converting Java applications 216
converting spreadsheet data 122
converting the data 200
converting the database structure 221
Cost Analysis worksheet template 62
Cost savings 1, 300
Cube Views 28
customized conversions 225

D

Dashboard 166
data capture 1
Data cleansing 287
data concurrency 263
data conformance 31
data consistency 155, 160, 196, 263
data consolidation 199
data conversion process 200
data conversion steps 220
data conversion time plan 201
Data Definition Language 207, 221
also see DDL
data definitions 2
data elements - common 275
data elements - uncommon 276
data federation 5, 30, 39, 106
data identification 31
data integration 5, 76
data integrity 158, 196, 263
data integrity - foundation 158
Data mapping 187
data mart xi, 2, 20, 50
impact analysis 245
Data mart architecture 11
Data Mart Business Loss Analysis 63
data mart consolidation xi, 1, 52
business case 54
project activities 150
data mart consolidation - also see DMC
Data Mart Consolidation Lifecycle 69, 149, 256
Data Mart Cost Analysis 62
data mart proliferation 51, 168
data mart total cost 177
data mart usage growth 176
data mining 164
data model 197
Data Modeling tools 112, 166
data models 11, 42, 266
dimensions 43
facts 43
grain 45
measures 43
data quality 151, 154, 160, 196, 263, 275
data quality and integrity 2
data quality methodology 158
data redundancy 55, 85, 160
data refresh - availability impact 245
data refresh - disk capacity 245
data refresh considerations 244
data refresh types 244
data replication 30, 41
data silos xi, 2
data timeliness 155
data transfer scripts 332
data transformation 1, 75, 106, 288
data transformation logic 187
Data Transformation Services 211
data types 75, 155
conversion 286
data update cycles 263
data warehouse 12, 16, 152
architecture 11, 16
Data Warehouse Edition (see DB2 Data Warehouse Edition)
data warehousing 8, 50
data warehousing implementations 18
data warehousing mplementations
Centralized 18
Distributed 19
Hub and Spoke 18
Virtual 19
data warehousing techniques 30
database configuration parameter
logbufsz 252
util_heap_sz 252
database partitioning 242

database structure conversion 221
DataJunction 207
DataStage (see IBM WebSphere Data Stage)
DB2 Alphablox 59, 104, 121
DB2 Call Level Interface 218
DB2 CLI driver 219
DB2 Connect 121, 248
DB2 Control Center 133, 224, 252
DB2 Cube Views 29, 60, 105
DB2 Data Propagator 205, 252
DB2 Data Warehouse Center 141
DB2 Data Warehouse Edition 58, 104
DB2 Data Warehouse Enterprise Edition 59
DB2 DDL statements 320
DB2 Design Advisor 233
DB2 Entity Analytics 111
DB2 ESE 241
DB2 Export utility 246
DB2 for z/OS 124
DB2 High Performance Unload 246, 253
DB2 Import utility 209, 246
DB2 Intelligent Miner 59, 105
DB2 iSeries 202
DB2 Load utility 250
DB2 Migration ToolKit 10, 108, 200, 209, 222, 265, 315
  also see MTK
DB2 native load 209
DB2 Office Connect Enterprise Web Edition 59, 105
DB2 OLAP Server 121, 132
DB2 optimizer 230
DB2 parameters
  ALLOW READ ACCESS 251
  LOCK WITH FORCE 251
  SAVECOUNT 251
  USE 251
DB2 Query Patroller 59, 106, 237
DB2 Relationship Resolution 111
DB2 SQL/XML 128
DB2 UDB 3, 59, 104, 221, 258, 265
DB2 UDB Command Line Processor 224
DB2 UDB Database Partitioning Feature 105
DB2 UDB Enterprise Server Edition 104
DB2 UDB scripts 224
DB2 Universal Database (also see DB2 UDB)
DB2 Warehouse Manager 59, 106, 131, 139, 200, 205, 209, 211
  spreadsheet scenario 140
  Transferring Excel data 117
DB2 XML Extender 124
db2batch command 249
db2batch utility 249
db2move utility 253
DDL 317–318, 322, 326, 337
decision support databases 4
deferred refresh 24, 240
DEL 246
denormalized tables 272
dependent data marts 21, 38, 52, 68, 152
Deploy 320, 322
Deploy to DB2 task 317
deployment scripts 332
Design Advisor 232
design phase 183, 257
DETAILED clause 231
DETAILED option 231
development cost 58, 99
dimension granularity 239
dimension mapping table 288
dimension metadata 155
dimension table attributes 155
Dimension table loading 290
Dimension versioning 155
dimensional data model 43, 91
dimensional modeling techniques 266
dimensions 43, 268
Direct server connection 129
directory of reports 197
Distributed consolidation 72, 82, 179, 275
  key features 83
DMC - ETL design differences 191
DMC assessment findings 168
DMC assessment phase 150–151
DMC implementation phase 151, 195
DMC planning phase 150, 178
DMC project plan 181
DMC project team 181
DMC without an EDW 190
DPF 241
drill-down 27
DriverManager 220
drop 319
duplicate support costs 55
Dynamic SQL 235

Archived

E
education 216
EDW 12, 76, 257, 264, 316
EDW architecture design 183
  Centralized Consolidation Approach 184
  Distributed Consolidation Approach 185
  Simple Migration Approach 183
EDW data model 272
EDW schema 78, 282
  design 183
EDW staging area 284
Embarcadero Technologies ER/Studio 207
Enterprise Data Warehouse (see EDW)
enterprise information integration 37, 40
Enterprise JavaBeans 218
Entity-Relationship model - see ER model
ER model 91, 221
ERWin 112
Essbase application manager 132
Essbase XTD Spreadsheet 132
ESSCMD command-line interface 132
ETL 40, 51, 73, 79, 205, 262
ETL code to populate the EDW 376
ETL construction 195
ETL design 183
ETL execution metadata 187, 291
ETL metadata 278
ETL process for consolidation 193
ETL processes 101, 151, 283
ETL tool 166
ETL transformation specifications 191
evaluating analytic structures 152
Event Monitor 237
Excel wrapper 133
executive sponsorship 114
expiration date 173
Export notebook 247
Export utility 246, 250
extensible markup language (see XML)
extent size 237
extract, transform, and load (see ETL)
Extracting 318, 326, 337

F
Fact conformation 188, 280
fact table 43
Fact table loading 290
facts 43, 268
fallback 202
fast sort operations 234
federated data access 37
federated server 34, 133
flat files 152, 209
flush package cache 236
Foreign keys and indexes 187
Frequency 31
full refresh 241, 244

G
Generate Data Transfer 321
Generate Data Transfer Scripts task 317
geo-spatial 225
getting data in 16
getting data out 16
glossary of terms 197
Granularity 239, 275
GUI 317

H
hardware cost 263
Hardware costs example 177
heterogeneous federation 34
hierarchies 155
high cost of data marts 11, 54
high frequency update 31
HOLAP 28, 100
Hybrid OLAP (see HOLAP)

I
IBM Content Manager 36
IBM DB2 Migration ToolKit (see DB2 Migration Tool-Kit)
IBM iSeries 205
IBM WebSphere DataStage 111, 191, 204, 225
IBM WebSphere MQ 37
IBM z/OS servers 205
immediate refresh 24, 240
Impact analysis 245
implementation phase 195, 257
Implementation recommendation report 182
Import 321
Import utility 247
importing 318, 326, 337
inconsistent data 74
inconsistent data definitions 2

Archived

incremental refresh 244
independent data marts 21, 38, 50, 61, 68, 74, 152
index defragmentation 234
index scans 233
index statistics 236
Index table space 230
indexes 232, 317
  block-based 26
  free space 234
  NLEVELS 235
  PCTFREE 234
  record-based 26
  Type 1 232
  Type 2 232
indexing best practices 233
indexing modes 250
  incremental 250
  rebuild 250
index-only access 233
information integrity 159
information integrity framework 159
information pyramid 13
informational data 12
Informix IDS 202
Informix XPS 202
isolation level
  exclusive lock 251
Ispirer Systems 207
IXF 246, 253

J

J2EE 59, 206
J2EE Application Servers 218
Java 317
Java access methods 217
Java applications 216
Java Server Pages 218
Java Servlets 218
JDBC 217, 318
JDBC drivers 218
join performance 234

K

Kumaran 206

L

Large objects 229
latency 24
load 321
  Loading an MDC table 252
Load notebook 252
Load/Unload 208
locking contention 240
logbufsz 252
logging 240
long fields 229
Loss Analysis template worksheet 63

M

maintenance costs 263
Manual methods 221
mapping Oracle data types 209
mapping XML schema to DB2 128
massively parallel processor (see MPP)
materialized query table (see MQT)
MDC 23, 236
MDC - cell density 237
MDC - composite block index 239
MDC - dimensions 236
MDC - extent sizes 237
MDC best practices 238
MDC load operations 252
MDC table 252
MDC table - performance 236
Message Oriented Middleware 204
metadata 61, 73, 80, 277, 317
metadata management 114, 291
metadata repository 197, 291
Metadata specialist 182
metadata standardization 61, 180, 186, 291
Metadata transport 221
Microsoft 132
  Bulk Copy Program 211
  Bulk Insert Utilities 211
  Data Access Components 220
  Data Transformation Services 211
  Excel 132
  Open Database Connectivity 218
  SQL Server 202
Microsoft SQL Server 203
migration tools 221
mkfifo 206
Modeling 59
Modifying/Constructing end user reports 195
MOLAP 28, 100

Archived

monotonic 239
MPP 242
MQ Series 128
MQT 23, 240
MAINTAINED BY SYSTEM 25
MAINTAINED BY USER 25
MQT - best practices 240
MQT - incremental refresh 241
MTK 202–203, 315–316
MTK - Features 203
MTK Graphical User Interface 318
multi-dimensional analysis 27
Multidimensional Clustering (see MDC)
Multidimensional OLAP (see MOLAP)
multi-layer architecture 17

N

named pipes 206, 247
natural key 289
network load 245
Normalization 42
normalized tables 272

O

object-relational mapping 124
ODBC 220, 318
ODBC application development 220
ODBC/JDBC 318
ODS 30, 152
ODS characteristics 31
OLAP 12, 27
OLE DB driver 132
OLTP 12, 47, 235–236
OLTP databases 152
Online analytical processing (see OLAP)
Open Database Connectivity - see ODBC
Openlink 130
Openlink multi-tier ODBC 130
operational data 12
Operational data store (see ODS)
operational procedures 196
operational transaction systems 17
Oracle 107, 203, 257
Oracle 9i 265
Oracle Data Dictionary 215
Oracle server 347
Oracle Warehouse Builder 208

P

package cache 236
parallel data export 249
parallel export function 249
parameter marker 235
partitioned database 249
PeopleSoft 204
performance - best practices 230
Performance - buffer pools 229
performance - database partitioning 241
performance - import/export utilities 248
performance - indexing 232
performance - MDC best practices 238
performance - memory 229
performance - MQT 240
performance and consolidation 227
performance and scalability 20
performance demands 228
Performance management 227
performance objectives 228
performance techniques 229
Physical column names 187
Physical table names 187
planning phase 178, 257
platform considerations 62
primary data mart 76
Product Dimension 269
productivity - developers and users 2
ProfileStage 191
Project management tools 166

Q

quality checks 287
quality of data 74, 114
Queue (Q) Replication 41

R

range of cells 136
Rational Rose Professional Data Modeler Edition 207
reach-through 28
real-time analytics 30
real-time business intelligence 14, 17
real-time data 14
real-time data integration 204
real-time data warehousing 17
recommendation report 182
record-based indexes 26

recovery plans 196
Redbooks Web site 394
  Contact us xiv
redundant data 262
referential integrity
  informational 240
Refine 317, 320
Refresh considerations 227
REFRESH DEFERRED 25, 240
REFRESH IMMEDIATE 240
Registering nicknames for Excel data 135
Registering the server for spreadsheet data 135
Registering the spreadsheet wrapper 134
Regular Fragmentations package 126
Relational OLAP (see ROLAP)
replication 205
report development 196
report specifications 196
reporting - maintenance costs 164
reporting environment 80, 97, 163, 180, 293
Reporting server configuration 195
Reporting tool client configuration 195
reporting tools 151, 178, 293
reports 320
  requirements 194
resource standardization 2
Risks, constraints, and concerns 181
ROLAP 28, 100
roll-out procedure 216
runstats utility 230
  DETAILED clause 231
  LIKE STATISTICS clause 231
  WITH DISTRIBUTION clause 230

S

SAMPLED option 231
SAP 204
Sarbanes Oxley 5, 53
SAS 204
SAX applications 125
schedules 196
schema changes 88
schemas 91
Scope definition 181
Scoring 59
SDK 220
Sequences 316
service level agreements 228
shredding 128
Simple migration 72, 179, 274
single version of the truth 5, 75
single view of the enterprise 3, 5
slicing 27
slowly changing dimensions 155
SMP (see symmetric multi-processor)
Snowflake schema 42
software license costs 263
software licenses 1
software updates 1
Solaris Operating Environment 205
Source to target data mapping 183, 195
space requirements 75
Specify Source 318, 326, 337
  task 317
sponsor for consolidation 179
Spreadsheet Add-in 132
spreadsheet consolidation example 137
spreadsheet data 117
spreadsheet data conversion 122
Spreadsheet data marts 68
spreadsheet document conversion 122
spreadsheet documents to XML 122
Spreadsheet integration 59
spreadsheet standards 117
spreadsheets 29, 117
  consolidation using DB2 OLAP Server 132
  consolidation using WebSphere II 137
  data transfer to DB2 117
  techniques for consolidating 121
  transfer with no conversion 129
  transferring data 117
SQL - efficient 235
SQL Replication 41
SQL Server 2000 73, 257, 265
SQL Translator 320
SQL*Plus 317
SQL/XML 124
SQLJ API 217
SQLWays 207
staging area 290
staging tables 240, 249
standardization 2
Star schema 42, 241
Star schema data models 266
Static SQL 235
static SQL statements 217
Statistical analysis 166

Archived

statistics 236
Store inventory data mart 265, 270
Store sales data mart 264, 266
stored procedure 72, 183, 203, 216, 222
summary tables 23
Supplier Dimension 269
support procedures 197
surrogate key assignments 288
surrogate keys 74, 79, 156, 269, 290
Sybase 107, 203
Sybase ASE 202
symmetric multi-processor 242

T

Table design
  normalization 46
target environment 215
Techne Knowledge Systems 207
technical metadata 74, 162, 187, 277, 291
Temporary table spaces 230
terabytes 241
Teradata 107
Testing phase 196
thin MQT 241
Third normal form 47
time series 225
tools for consolidation 103
total cost of ownership 55
training 99
training and skills development 55
transforming data from DB2 to spreadsheet 129
Translator 317
trend analysis 27
triggers 72, 183, 203, 222
Type 1 index 232
Type 2 index 232

U

uncommon data elements 276
unique indexes 233
unique natural key 289
Universal JDBC Driver 217
Unload 321
User defined data types 183
User education 216
user interface transactions 196
User reports 183
user-defined functions 203, 223
util_heap_sz 252

V

Velocity 31
Views 183
Visualization 59

W

WebSphere Data Integration Suite 191
WebSphere DataStage 191, 204, 209
WebSphere DataStage (also see IBM WebSphere DataStage)
WebSphere II 10, 34–35, 41, 59, 106, 121, 133, 200, 205, 209, 212, 265, 286, 315, 344
  creating the Oracle server 347
  creating the Oracle wrapper 346
  wrappers 35
  XML wrapper 124
WebSphere II for Content 36
WebSphere Information Integrator (see WebSphere II)
WebSphere MQ 204
WebSphere ProfileStage 191
why consolidate data marts 1
Windows 2000 205
Windows NT 205
wrappers 35
WSF 246

X

XML 33, 122
XML Extender for DB2 124
XML schema - global 125
XML schema - local 125
XML schema to a spreadsheet 129
XML shredding 123
XML Wrapper 124
XSLT - to transform XML 125

Z

z-lock 241

Archived

Data Mart Consolidation: Getting Control of Your Enterprise Information

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Archived

Archived

Archived

Data Mart Consolidation: Getting Control of Your Enterprise Information

Managing your information assets and minimizing operational costs

Enabling a single view of your business environment

Minimizing or eliminating those data silos

This IBM Redbook is primarily intended for use by IBM Clients and IBM Business Partners. The current direction in the Business Intelligence marketplace is towards data mart consolidation. Originally data marts were built for many different reasons, such as departmental or organizational control, faster query response times, easier and faster to design and build, and fast payback.

However, data marts did not always provide the best solution when it came to viewing the business enterprise as a whole. They provide benefits to the department or organization to whom they belong, but typically do not give management the information they need to efficiently and effectively run the business.

In many cases the data marts led to the creation of departmental or organizational data silos (non-integrated sources of data). That is, information was available to the particular department or organization, but was not integrated across all the department's or organizations. Worse yet, many data marts were built without concern for the others. This led to inconsistent definitions of the data, inconsistent collection of data, inconsistent collection times for the data, and so on. The result was an inconsistent picture of the business for management, and an inability for good business performance management. The solution is to consolidate those data silos to provide management the information they need.

SG24-6653-00 ISBN 0738493732

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