GETTING STARTED WITH

Data Warehousing

A book for the community by the community

Neeraj Sharma, Abhishek Iyer, Rajib Bhattacharya, Niraj Modi,
Wagner Crivelini

FIRST EDITION

First Edition (February 2012)

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Notices

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

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

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

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

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

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.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

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

Table of Contents

Preface

Keeping your skills current in today's world is becoming increasingly challenging. There are too many new technologies being developed, and little time to learn them all. The DB2® on Campus Book Series has been developed to minimize the time and effort required to learn many of these new technologies.

This book intends to help professionals understand the main concepts and get started with data warehousing. The book aims to maintain an optimal blend of depth and breadth of information, and includes practical examples and scenarios.

Who should read this book?

This book is for enthusiasts of data warehousing who have limited exposure to databases and would like to learn data warehousing concepts end-to-end.

How is this book structured?

The book starts in **Chapter 1** describing the fundamental differences between transactional and analytic systems. It then covers the design and architecture of a data warehouse in **Chapter 2**. **Chapter 3** talks about server and storage hardware design and configuration. **Chapter 4** covers the extract, transform and load (ETL) process. Business Intelligence concepts are discussed in **Chapter 5**. A case study problem statement and its end-to-end solution are shown in **Chapter 6**. **Chapter 7** covers the required tasks for maintaining a data warehouse. The book concludes discussing some trends for data warehouse market in **Chapter 8**.

The book includes several open and unanswered questions to increase your appetite for more advanced data warehousing topics. You need to research those topics further on your own.

Exercises are provided with most chapters. **Appendix A** provides a list of all database diagrams, SQL scripts and input files required for the end-to-end case study described in **Chapter 6**.

Appendix B shows the instructions and links to download and install the required software used to run the exercises included in this book.

Finally, **Appendix C** shows a list of referenced books that the reader can use to go deeper into the concepts presented in this book.

A book for the community

The community created this book, a community consisting of university professors, students, and professionals (including IBM employees). The online version of this book is released at no charge. Numerous members around the world have participated in developing this book, which will also be translated to several languages by the community. If you would like to provide feedback, contribute new material, improve existing material, or help with translating this book to another language, please send an email of your planned contribution to db2univ@ca.ibm.com with the subject "Getting Started with Data Warehousing book feedback."

Conventions

Many examples of commands, SQL statements, and code are included throughout the book. Specific keywords are written in uppercase bold. For example: A **`NULL`** value represents an unknown state. Commands are shown in lowercase bold. For example: The **`dir`** command lists all files and subdirectories on Windows. SQL statements are shown in upper case bold. For example: Use the **`SELECT`** statement to retrieve information from a table.

Object names used in our examples are shown in bold italics. For example: The ***`flights`*** table has five columns.

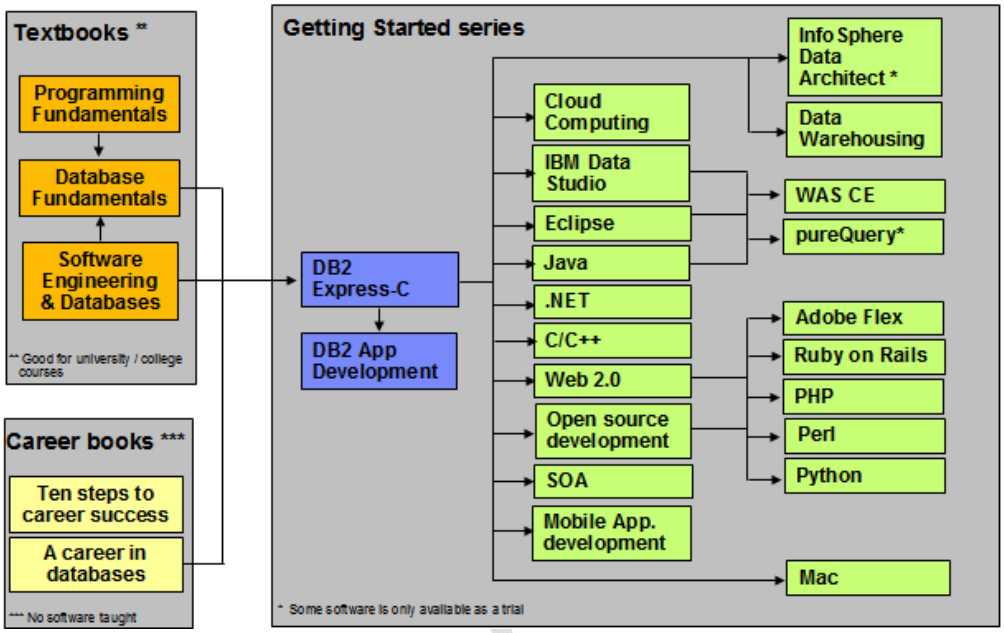
Italics are also used for variable names in the syntax of a command or statement. If the variable name has more than one word, it is joined with an underscore. For example: **`CREATE TABLE`** ***`table_name`***

What's Next?

We recommend you to read the following books in this book series for more details about related topics:

- *Getting Started with Database Fundamentals*
- *Getting Started with DB2 Express-C*
- *Getting started with IBM Data Studio for DB2*

The following figure shows all the different eBooks in the DB2 on Campus book series available for free at ibm.com/db2/books



The DB2 on Campus book series

About the Authors

Neeraj Sharma is a senior software engineer at the Warehousing Center of Competency, India Software Labs. His primary role is in the design, configuration and implementation of large data warehouses across various industry domains, creating proof of concepts; execute performance benchmarks on customer requests. He holds a bachelor's degree in electronics and communication engineering and a master's degree in software systems.

Abhishek Iyer is a Staff Software Engineer at the Warehousing Center of Competency, India Software Labs. His primary role is to create proof of concepts and execute performance benchmarks on customer requests. His expertise includes data warehouse implementation and data mining. He holds a bachelor's degree in Electronics and Communication.

Rajib Bhattacharya is a System Software Engineer at IBM India Software Lab (Business Analytics). He has extensive experience in working with enterprise level databases and Business Intelligence. He loves exploring and learning new technologies. He holds a master's degree in Computer Applications and is also an IBM Certified Administrator for Cognos BI.

Niraj Modi is a Staff Software Engineer at IBM India Software Lab (Cognos R&D). He has worked extensively on developing software products with the latest Java and open source technologies. Currently Niraj is focused on developing rich internet application products in the Business Intelligence domain. He holds a bachelor's degree in Computer Science and Engineering.

Wagner Crivelini is a DBA at the Information Management Center of Competence, IBM Brazil. He has extensive experience with OLTP and Data warehousing using several different RDBMS's. He is an IBM Certified DB2 professional and also a guest columnist for technical sites and magazines, with more than 40 published articles. He has a bachelor's degree in Engineering.

Contributors

The following people edited, reviewed, provided content, and contributed significantly to this book.

Contributor	Company/University	Position/Occupation	Contribution
Kevin Beck,	IMB US Labs	DWE Development - Workload Management	Development of content for database partitioning, table partition, MDC.
Raul F. Chong	IBM Canada Labs – Toronto, Canada	Senior DB2 and Big data Program Manager	DB2 on Campus Book Series overall project coordination, editing, formatting, and review of the book.
Saurabh Jain	IBM India Software Labs	Staff Software Engineer	Reviewed case study for flow and code correctness.
Leon Katsnelson	IBM Canada Labs – Toronto, Canada	Program Director, IM Cloud Computing Center of Competence and Evangelism	Technical review
Ganesh S Kedari	IBM India Software Labs	Quality Engineer, IBM Cognos	Helped in Cognos content development.
Sam Lightstone	IBM Canada Labs	Program Director, DB2 Open Database Technology	Development of content for database partitioning, table partition, MDC.
Amitkumar D Nagar	IBM India Software Labs	Quality Engineer, IBM Cognos	Helped in Cognos content development.
Kawaljeet Singh	IBM India Software Labs	Quality Engineer, IBM Cognos	Helped in Cognos content development.
Avinash M Swami	IBM India Software Labs	Manager - System Quality Dev, IBM Cognos	Overall coordination, for Cognos content development.
Xiaomei Wang	IBM Toronto Lab	Technical Champion, InfoSphere Balanced	Technical content review.

		Warehouse	

Acknowledgements

We greatly thank Natasha Tolub for designing the cover of this book.

1

Chapter 1 – Introduction to Data Warehousing

A **Warehouse** in general is a huge repository of commodities essentially for storage. In the context of a Data Warehouse as the name suggests, this commodity is **Data**. An obvious question that now arises is how different is a data warehouse from a database, which is also used for data storage? As we go along describing the origin and the need of a data warehouse, these differences will become clearer.

In this chapter, you will learn about:

- A brief history of Data Warehousing
- What is a Data Warehouse?
- Primary differences between transactional and analytical systems.

1.1 A Brief History of Data Warehousing

In the 1980's organizations realized the importance of not just using data for operational purposes, but also for deriving intelligence out of it. This intelligence would not only justify past decisions but also help in making decisions for the future. The term **Business Intelligence** became more and more popular and it was during the late 1980's that IBM researchers Barry Devlin and Paul Murphy developed the concept of a **Business data warehouse**.

As business intelligence applications emerged, it was quickly realized that data from transactional databases had to first be transformed and stored into other databases with a schema specific for deriving intelligence. This database would be used for archiving, and it

would be larger in size than transactional databases, but its design would make it optimal to run reports that would enable large organizations to plan and proactively make decisions. This separate database, typically storing the organization's past and present activity, was termed a ***Data Warehouse***.

1.2 What is a Data Warehouse?

Similar to a real-life warehouse, a Data Warehouse gathers its data from some central source, typically a transactional database and stores and distributes this data in a fashion that enables easy analytics and report generation. The difference between a typical database and a data warehouse not only lies in the volume of data that can be stored but also in the way it is stored. Technically speaking, they use different database designs, a topic we will cover in more detail in **Chapter 2**.

Rather than having multiple decision-support environments operating independently, which often leads to conflicting information, a data warehouse unifies all sources of information. Data is stored in a way that integrity and quality are guaranteed. In addition to a different database design, this is accomplished by using an **Extract, Transform** and **Load** of data process -- also known as **ETL**. Along with corresponding **Business Intelligence Tools,** which collate and present data in appropriate formats, these combination provides a powerful solution to companies for deriving intelligence.

The ETL process for each Data Warehouse System is defined considering a clear objective that serves a specific business purpose; the data warehouse focus and objective directly influence the way the ETL process is defined. Therefore, the organization's business objective must be well known in advance, as it is essential for the definition of the appropriate transformation of data. These transformations are nothing more than restructuring data from the source data objects (source database tables and/or views) to the target ones.

All in all, the basic goal of the ETL is to filter redundant data not required for analytic reports and to converge data for fast report generation. The resultant structure is optimized and tailored to generate a wide range of reports related to the same business topic. Data is '**staged**' from one state to another and often different **stages** suit different requirements.

1.3 OLTP and OLAP Systems

This section describes two typical types of workloads:

- Online Transaction Processing (***OLTP)***
- Online Analytical Processing (***OLAP***)

Depending on the type of workload, your database system may be designed and configured differently.

1.3.1 Online Transaction Processing

Online Transaction Processing (OLTP) refers to workloads that access data randomly, typically to perform a quick search, insert, update or delete. OLTP operations are normally performed concurrently by a large number of users who use the database in their daily work for regular business operations. Typicaly, the data in these systems must be consistent and accurate at all times. The life span of data in an OLTP system is short since its primary usage is in providing the current snapshot of transient data. Hence, OLTP systems need to support *real-time* data insertions, updates and retrievals, and end up having large number of small-size tables.

Consider an online reservation system as an example of an OLTP system. An online user must be presented with accurate data 24 x 7. Reservations must be done in a quick fashion and any updates on the reservation status must be reflected immediately to all other users.

In addition to online reservations systems, other examples of OLTP systems, include banking applications, eCommerce, and payroll applications. These systems are characterized by their simplicity and efficiency, which help enable 24 x 7-support to end users.

OLTP systems use simple tables to store data. Data is ***normalized***, that is, redundancy is reduced or eliminated while still ensuring data consistency. Data is stored in its utmost raw form for **each** customer transaction.

For example, *Table 1.2* shows rows of a normalized transaction table. Picture this scenario: A customer with customer Id ***C100102*** goes early in the morning and buys a shaving razor (***P00100***) and an after-shave lotion (***P02030***). These transactions are stored in rows 1 and 2 in the table. When he arrives home, he realizes he is short of shaving cream as well. Therefore, he goes back to the store and buys the shaving cream of his choice (***P00105***) which is shown in the last row. As you can see, a separate independent entry was stored in the table even for the same customer and data was *not* grouped in any form. Each row

represents an individual transaction and all entries are exposed equally for general query processing.

Customer_id	Order_id	Product_id	Amount	Timestamp
C100102	1101	P00100	120	2012-01-01 09:30:15:012345
C100102	1101	P02030	535	2012-01-01 09:30:15:012351
C010700	1102	P24157	250	2012-01-01 09:35:12:054321
C002019	1103	P87465	250	2012-01-01 09:45:12:054321
C002019	1103	P00431	420	2012-01-01 09:45:15:012345
C100102	1104	P00105	150	2012-01-01 10:16:12:054321

Table 1.2 Normalized transaction table

An alternative approach of storage would be the **denormalized** method where there are different *levels* of data. Consider the same data stored in a different way in *Table 1.3* below.

Customer_id	Transactions			
	Order_id	Product_id	Amount	Timestamp
	1103	P87465	250	2012-01-01 09:45:12:054321
C002019	1103	P00403	420	2012-01-01 09:45:15:012345
C010700	1102	P24157	250	2012-01-01 09:35:12:054321
	1101	P00100	120	2012-01-01 09:30:15:012345
	1101	P00100	535	2012-01-01 09:30:15:012351
C100102	1104	P00105	120	2012-01-01 10:16:12:054321

Table 1.3 Denormalized transaction table

In *Table 1.3*, data is grouped based on **`Customer_id`**. Now, in case there is a requirement to fetch all the transactions done on a particular date, data would need to be resolved at two levels. First, the inner group pertaining to each **`Customer_id`** would be resolved and then data would need to be resolved across all customers. This leads to increased complexity that is not recommended for simple queries.

Since OLTP databases are characterized by high volume of small transactions that require instant results and must assure data quality while collecting the data; they need to be normalized.

There are different levels of database normalizations. The decision to choose a given level is based on the type of queries that are expected to be issued. Lower normal forms offer greater simplicity, but are prone to insert, update and delete anomalies and they also suffer from functional dependencies. In fact, the table shown in *Table 1.2* is in the Third Normal Form (3NF). Although there are other normal forms higher than 3NF, suitable for specific business cases, 3NF is the most common and usually the lowest normal form acceptable for OLTP databases.

Note:

For more information about normalization and different normalization levels, refer to the book *Database Fundamentals* which is part of the DB2 on Campus book series.

Another key requirement of any transactional database is its reliability. Such systems are critical for controlling and running the fundamental business tasks and are typically the first point of data entry for any business. Reliability can be achieved by configuring these databases for high availability and disaster recovery. For example, DB2 software has a featured called High Availability Disaster Recovery (HADR). HADR can be set up in minutes and allows you to have your system always up and running. Moreover, thanks to Cloud Computing, an HADR solution is a lot more cost-effective than in the past.

Note:

For more information about HADR, refer to the book *Getting started with DB2 Express-C*, which is part of the DB2 on Campus book series.

1.3.2 Online Analytical Processing

Online Analytical Processing (**OLAP**) refers to workloads where large amounts of historical data are processed to generate reports and perform data analysis. Typically, OLAP databases are fed from OLTP databases, and tuned to manage this type of workload. An OLAP database stores a large volume of the same transactional data as an OLTP database, but this data has been transformed through an ETL process to enable best performance for easy report generation and analytics. OLTP systems are tuned for extremely quick inserts, updated and deletes, while OLAP systems are tuned for quick reads only.

The lifespan of data stored in a Data Warehouse is much longer than in OLTP systems, since this data is used to reflect trends of an organization's business over time and help in decision making. Hence OLAP databases are typically a lot larger than OLTP ones. For

instance, while OLTP databases might keep transactions for six months or one year, OLAP databases might keep accumulating the same type of data year over year for 10 years or more.

As compared to OLTP systems, data in an OLAP data warehouse is less normalized than an OLTP system. Usually OLAP data warehouses are in the 2nd Normal Form (2NF). The great advantage of this approach is to make database design more readable and faster to retrieve data.

Some examples of OLAP applications are business reporting for sales, marketing reports, reporting for management and financial forecasting.

The large size of a data warehouse makes it not economically viable to have a high availability and disaster recovery setup in place. Since OLAP systems are not used for real-time applications, having another exact replica of an existing huge system would not justify neither the costs, nor the business needs.

Sophisticated OLAP systems (warehouses) which typically comprise of M servers do offer High Availability options in which M primary servers can be configured to failover to N standby nodes where M > N. This is made possible by using shared storage between the M primaries and the N standby nodes. Typically, for small to medium warehouses, the (M+1) configuration is suggested.

1.3.3 Comparison between OLTP and OLAP Systems

As mentioned before, OLTP and OLAP systems are designed to cater for different business needs and hence they differ in many aspects. The table below lists the differences between them based on different factors.

Differentiating Factor	OLTP Systems	OLAP Systems
Business Need and Usage	These systems are data stores for real time transactional applications and are typically the first data entry point for any organization. They are critical for controlling and running the fundamental business tasks of an organization.	These systems are needed by an organization to generate reports, run analytics useful for decision making in a multiple decision support environment. Data in such systems is sourced from various OLTP systems and consolidated in a specific format.
Nature of Data Stored	Data stored in such systems represent the current snapshot of	Such systems contain historical data that is gathered from operational

	transient data. Data is collected real time from user applications. There is no transformation done to the data before storing it into the system.	databases over a period. The data stored reflects the business trends of the organization and helps in forecasting. After transformation (ETL), data is generally loaded into such systems periodically.
Database Tuning	Database is tuned for extremely fast inserts, updates and deletes.	Database is tuned only for quick reads.
Data Lifespan	Such systems deal with data of short lifespan.	Such systems deal with data of very large lifespan (historic).
Data Size	Data in OLTP systems is raw and it is stored in numerous but small-size tables. The data size in such systems is hence not too big.	Data in OLAP systems is first transformed and usually stored in the form of fact and dimension tables. The data size of such systems is huge.
Data Structure	Data is stored in the highest normalized for possible. Usually 3NF.	Data is somewhat denormalized to provide better performance. Usually under 2NF (important: this denormalization applies to dimension tables only).
Data Backup and Recovery	One of the main requirements of an OLTP system is its reliability. Since such systems control the basic fundamental tasks of an organization, they must be tuned for high availability and data recovery. Such systems cannot afford to go offline since they often have mission critical applications running on them. Hence, an HADR setup with the primary and secondary (with their respective storage) installed in different geographies is recommended.	Such systems need not require high availability and data may be archived in external storage such as tapes. In case such a system goes down, it would not necessarily have a critical impact on any running business. Data can be reloaded from archives when the system comes up again. In case required, a high availability solution with shared storage between M primaries and N secondaries (M > N) can be setup for an OLAP system.
Examples	Banking Applications, Online Reservations systems, ecommerce etc.	Reporting for sales, marketing, management reporting and financial forecasting.

1.4 Case Study

Let's take a look at an example. Consider a retail outlet **GettingStarted Stores** having its outlets across the country. Apart from the normal daily transactional processing in the stores, the owner of the company wants certain reports at the end of the day that can help him see the trends of his business all over the country. For example, which product is selling the most in a given region, or which area contributes more to maximize profit? Let's take a look at the following two sample reports:

- Regional contribution to sales profit. (Organized by region, zone and area)
- Product (category) wise contribution to sales profit.

The transaction table in the database server of **GettingStarted Stores** would look as illustrated in *Table 1.1*.

Order_id	Product_id	Store_id	Amount	Timestamp
1101	P001	S001	120.00	2007-01-01 09:30:15:012345
1102	P102	S002	250.00	2007-04-10 09:31:12:054321

Table 1.1 Transaction table of GettingStarted Stores

The mapping of **`Product_id's`** to its description, category, associated margin etc. would be maintained in separate tables on the server. Similarly, **`Store_id`** would be mapped to Store name, Region, Zone, Area in separate tables. (Figure 1.1 illustrates such database model).

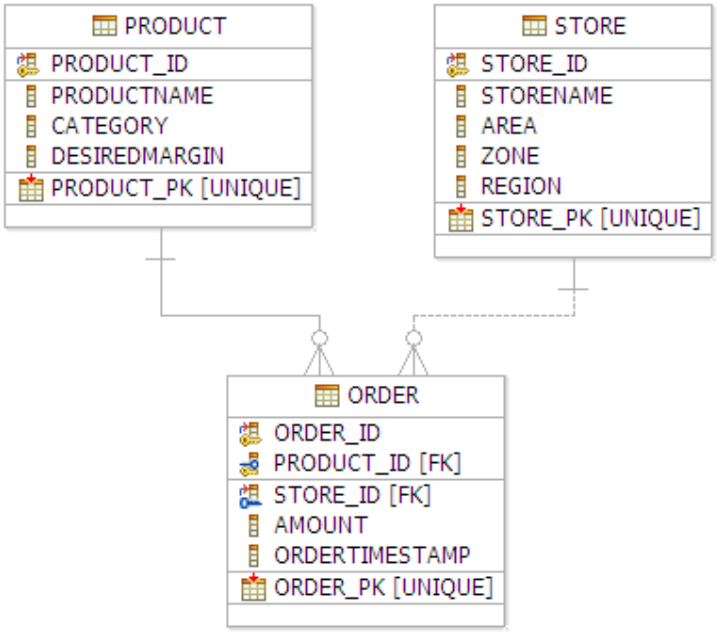


Figure 1.1 Example of a database model used for transaction processing

Generating the required reports by writing SQL to fetch and relate data from such tables would not only be a tedious task, but would also not be scalable at all. Any minor change required in the report would require changing many SQL scripts. (Refer to **Appendix A** for SQL scripts examples).

To show an example of how the data is restructured, consider the requirements of a report needed by the top management of **GettingStarted Stores** which shows the product (category) wise contribution to sales profit at the end of the year.

The main transaction table that would store each transaction, taking place across the country would look like the one shown in Table 1.4 below.

Order_id	Product_id	Store_id	Amount	Timestamp
1102	P24157	DF002	250	2007-04-10 09:35:12:054321
1103	P87465	DF002	250	2007-04-10 09:45:12:054321
1104	P10267	DF101	155	2007-04-10 09:46:12:054321
1104	P11345	DF101	550	2007-04-10 09:46:12:054321
1105	P10342	DF223	100	2007-04-10 09:48:12:054431
1106	P32143	DF312	175	2007-04-10 09:49:12:054321
1106	P32145	DF312	345	2007-04-10 09:49:13:054321

Table 1.4 Main transaction table

Apart from this main transaction table, the organization would also have another table, which would contain information pertaining to each product like its description, category, associated margin etc.

Product_id	Prod_Name	Prod_Category	Prod_BaseCost	Prod_Margin
P00001	Milk Powder	Groceries	55	10
P00002	Coffee	Groceries	120	12
P00003	Detergent Soap	Groceries	105	15
P00005	Tomato Juice	Groceries	70	5
P00006	Antiseptic Lotion	Health Care	25	5
P00007	Cotton	Health Care	15	7
P00008	Beer	Liquor	65	10

Table 1.5 Product table

There might also be other tables, which would have information about various stores across the country, for instance (e.g., mapping of the store id to its state, region, zone and area). But in the OLTP database, there might be several tables to store all this regional data. So, when putting all this data together, the user will need to join all those tables. (This is, by the way, one of the ETL processes usually required when creating a data warehouse).

Store_id	Store_State	Store_Region	Store_Zone	Store_Area
S001	Karnataka	Bangalore	Bangalore South	Jayanagar
S012	Maharashtra	Mumbai	Mumbai Central	Andheri
S043	Delhi	New Delhi	South Delhi	South Extension
S101	Maharashtra	Mumbai	Mumbai South	Colaba
S223	Maharashtra	Pune	Pune Central	F.C Road
S312	Karnataka	Bangalore	Bangalore East	Whitefield
S415	Gujarat	Ahemadabad	Ahemadabad Central	Ashram Road

Table 1.6 Regional Data

Those source tables with different regional data might exist in different databases from the OLTP system. The size of such tables will mostly be constant and it will be change only when a new product or a store is added.

On the other hand, the main transaction table will frequently change and its size will be constantly increasing.

This main transaction table in OLAP terminology is called a **'Fact'** table. The transactional details stored in the fact tables (like amount, dollars and so) are also known as '**Measures'**. And the data that categorizes these measures or facts is termed as **'Dimensions'**. In other

words, the dimensions provide information on how to qualify and/or analyze the measures. In the above example, the Product table and the Regional table are the two dimensions associated with the main fact table.

The combination of a fact and its associated dimensions is termed as an **OLAP Cube**. An OLAP cube is the basic architectural block of a Data Warehouse. The source transactional data from the OLTP database is **staged** systematically via the ETL process to converge into OLAP cubes. Certain columns that would be never used for report generation (like Customer Id) may also be dropped from the main transactional table before creating a Fact Table from it. In addition, many smaller tables may have to be joined to derive the different dimension tables. In the example mentioned earlier, the Product and the Regional Data tables were a result of many joins performed on smaller tables.

1.5 Summary

Any system whether transactional or analytical is solely dependent on its positioning in the solution and its intended usage. Same database engines can be tuned to meet either transactional requirements or the analytical requirements. Despite that, some database vendors have separate database engines for transactional and warehousing applications.

1.5 Review Questions

1. What does OLTP stand for?
    A. Online Travel Processing
    B. Online Travel Planning
    C. Online Transactional Processing
    D. Offline Transactional Processing

2. What does OLAP stand for?
    A. Online Analytical Processing
    B. Online Analytical Programming
    C. Offline Analysis and Programming
    D. Online Arithmetic Programming

3. Which one of the following is/are true for an OLTP system?
   A. Tuned for quick reads
   B. Tuned for quick inserts, updates and deletes
   C. Need not require backup and recovery
   D. Contain historical data in huge tables

4. Which one of the following is/are true for an OLAP system?
   A. Tuned for quick reads
   B. Tuned for quick inserts, updates and deletes
   C. Need not require backup and recovery
   D. Contain historical data in huge tables

5. OLAP systems are used as data stores for real-time applications.
   A. Yes
   B. No

6. In a transactional system, data is normalized to the highest form possible.
   A. Yes
   B. No

7. Data is generally extracted from an OLAP system, transformed and then loaded into an OLTP system
   A. Yes
   B. No

8. An OLAP structure is the basic architectural block of a Data Warehouse.

A. Yes

B. No

9. In an analytic system, data is normalized to the fastest form possible.

   A. Yes

   B. No

1.6 Exercises

Design a railway reservation analytic system/architecture and depict it using a block diagram, which consists of the following components:

A. A Web interface that accepts online data from the user in form of new reservation request, reservation modification request and reservation cancellation requests.

B. A real-time transactional system, which stores, retrieves and updates the live reservation data.

C. An ETL system that extracts data from a data source in real-time, does some transformation and loads into target data warehouse.

D. A Data Warehouse where the transformed data is loaded.

E. A reporting interface, which is used to present the analytic reports to the end-user.

2

Chapter 2 – Data Warehouse Architecture and Design

In this chapter, we will provide a brief overview of the warehouse architecture and the design in general with references to IBM Infosphere Warehouse.

2.1 The Big Picture

As discussed in **Chapter 1**, a data warehouse is a huge repository of data, created with the purpose of retrieval as and when business needs any data to support business decisions. Before we get into the discussion of each individual component, let us first try to visualize how all these component fit together. Figure 2.1 depicts how various components are placed from a bird's eye view.

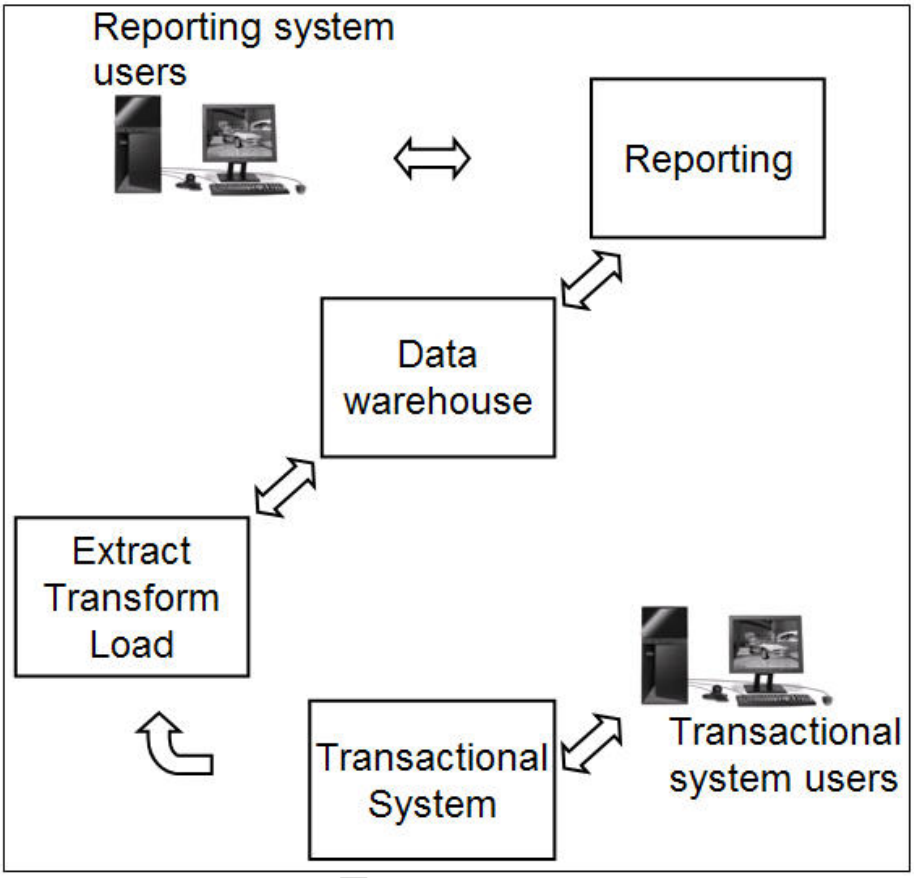


Figure 2.1 High-level view of component placement for business intelligence

Figure 2.1 shows logical placement of components and direction of information exchange between them. All these components may co-exist on a single system or they can be deployed on different systems connected via LAN infrastructure. Figure 2.2 shows placement of these components with one, two and three tier architecture.

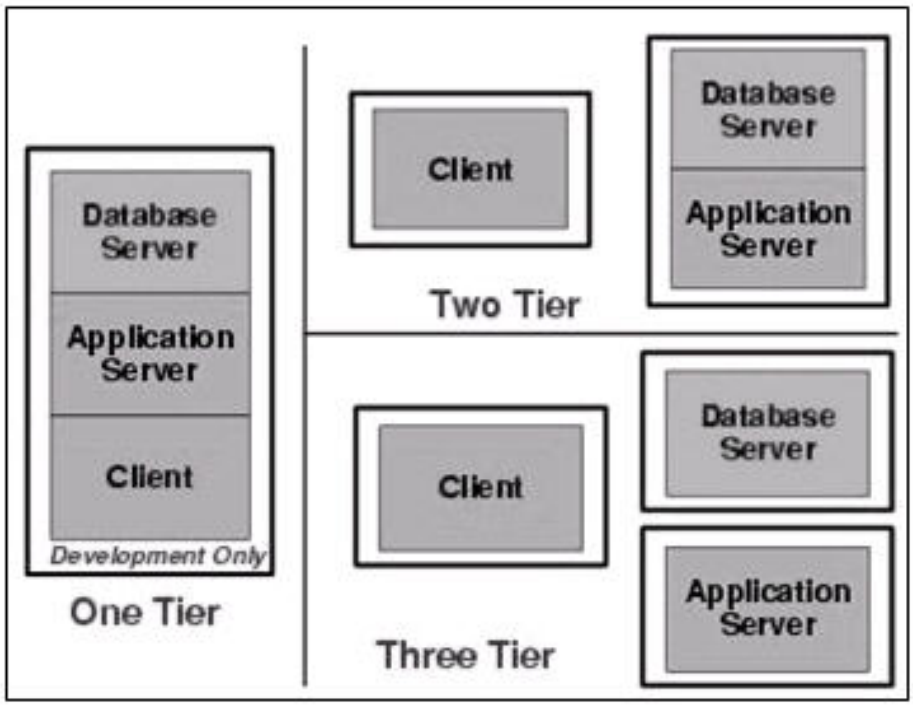


Figure 2.2 Logical placements of components for one, two and three tier architecture

Figure 2.2 above shows one, two and three tier architecture for any generic system. As shown in the figure, during development phases of any application (especially during prototyping) all components sit on the same system. For commercial deployment of the system, a two-tier model becomes a necessity. In fact, three-tier model is always recommended for any commercial/production system as it provides higher level of scalability and performance, and is easy to administer and maintain.

2.2 Online Analytical Processing (OLAP)

Online analytical processing is a system and method to store data objects in a form that will help quick processing of multidimensional queries. The heart of any OLAP system is called an OLAP Cube (or OLAP database if you will). An OLAP cube is a subset of the Data Warehouse, with a very specific subject. Usually OLAP cubes focus on departmental needs, while the Data Warehouse keep the focus on the organization as a whole.

An OLAP cube comprises of **Measures** and **Dimensions**. Measures are metrics that help business users in assessing business operation efficiency by providing appropriate data in

a context. Dimensions are business objects that provide context to Measures. For example, “Average attendance of Students” for each “Course Offered” in the university in “Current Year”. Here, “Average attendance of Students” is a Measure that indicates students’ interest in attending classes for courses offered (courses dimension) in current year (time dimension). The metadata describing relationship of these Measures and Dimensions to the underlying base tables is called the OLAP Cube definition or simply an **OLAP Cube**.

Online analytical processing stores these cubes in different modes. These modes are of following types:

1. Multidimensional OLAP (MOLAP)

   This is the traditional mode of storing OLAP data, as multidimensional structures. This data can be read from this multidimensional OLAP system via MDX (**M**ulti **D**imensional e**X**pression). Although the MDX syntax might seem similar to the SQL syntax at first look, MDX takes advantage of dealing directly with multidimensional structures to provide a much simpler way to query multidimensional data. All Multidimensional OLAP providers have their on proprietary way of storing and retrieving data. As of date, even MDX is not yet a standard. MOLAP systems provide very good performance for analytic operations (slice, dice etc), but they are usually limited to a relatively small data set (few Gigabytes). MOLAP systems have a very specialized analytics engine, which helps in performing complex analysis.

2. Relational OLAP (ROLAP)

   When data is stored in a two dimensional tabular format, ROLAP systems break down MDX query into a SQL query and retrieves the data using SQL from relational database (obviously, the R in the acronym stands for Relational). ROLAP systems have relatively slow response time (with respect to MOLAP system); however, ROLAP can handle very large data volumes (of the order of Terabytes). Analysis capability of a ROLAP system is limited to analytic capability of underlying relational database in use.

3. Hybrid OLAP (HOLAP)

   Hybrid OLAP, as the name suggests, stores data in both relational and multidimensional format, and therefore provides advantages of both MOLAP and ROLAP systems. The system is smart enough to identify when to use MOLAP and when to use ROLAP.

2.3 The Multidimensional Data Model

Data stored in transactional databases typically is in the relational form. That simply means a set of tables with various constraints and referential integrity conditions. However, this relational data format is not the way data is required for making business decisions. Remember referential integrity is particularly important when inserting and/or updating data and data warehouses deal primarily with read-only data! Figure 2.3 shows a sample transactional system recording sales in an electronic items store.

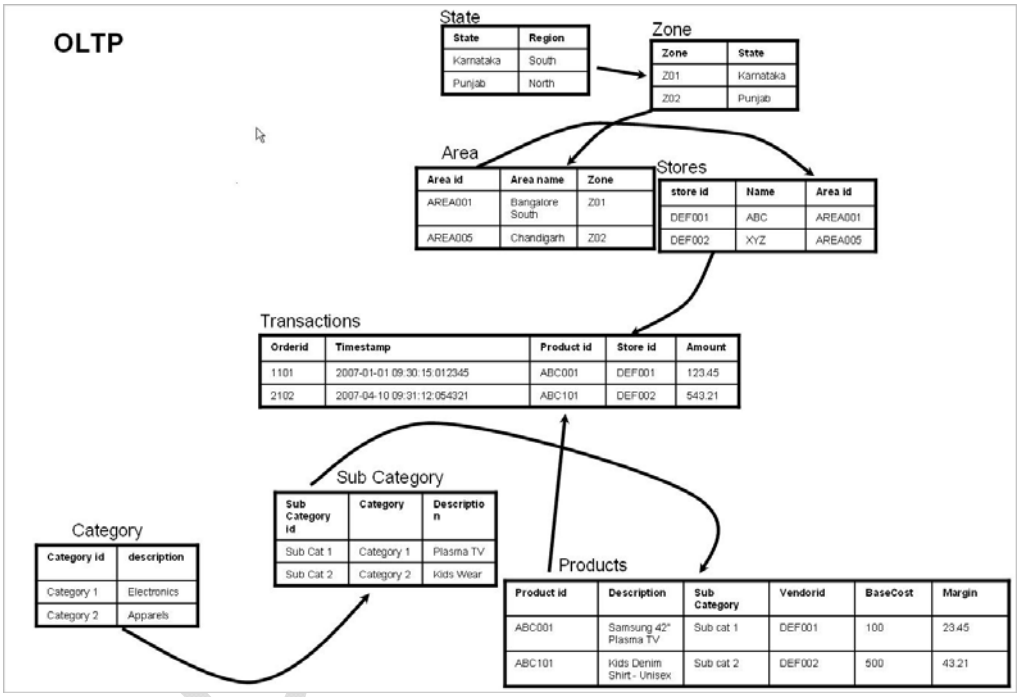


Figure 2.3 Relational data model for a sales transactional system

Figure 2.3 shows a very simple sales transactional system capturing only order id, timestamp (of order), product id (of product ordered), store id (where order was placed) and amount of the product for each order placed.

The Product Id maps to respective details like, product category, its description, sub category etc. in different tables, which may or may not reside in the same database. Similarly, the store id maps to its store name, and its area id in separate tables as shown above.

Now, suppose there is a need to know total sales for “Electronics” category for “North” region during year “2007”. To answer this query, we will have to join all the eight tables shown in figure 2.3 and filter it out for year “2007”. If the above store has 10,000 products in “Electronics” category, 1000 stores in “North” region and considering that a typical number of transactions in a single year is around a few million, it is highly likely that the system will never return the query result due to such a large table join that might not even fit in system’s physical/virtual memory. Therefore, it is desired to keep data in a format that requires minimum runtime table joins and calculations (like filter on year in example above). Data in a transactional system is stored at a normalization level that is most suited for quick inserts, updates and deletes. In case of an analytics system, it is stored in a normalized form that is fastest for select (read) statements.

Figure 2.4 shows OLAP converged table structure for OLTP system shown in figure 2.3.

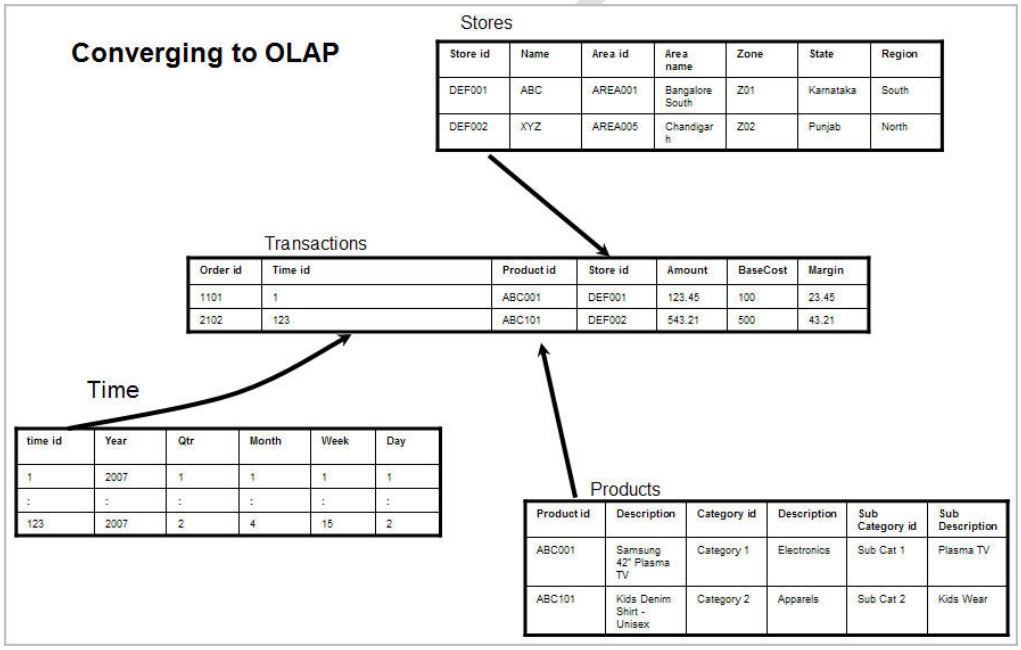


Figure 2.4 Restructured tables suitable for OLAP queries

By restructuring our system as shown in figure 2.4, we have reduced number of table joins from eight to just four. This restructuring of database objects based on specific reporting needs is nothing but what we termed as ‘ETL’. We can further reduce the amount of data read by partitioning data across multiple systems. For details on this subject, please refer to IBM Redbook “Database Partitioning, Table Partitioning, and MDC”.

The table structure as shown in figure 2.4 is a simple representation of a star schema, a special multidimensional design for relational database. In this example, the tables "Products", "Stores" and "Time" are dimensions and the "Transactions" table is the fact table. Figure 2.5 depicts the same in terms of facts and dimensions.

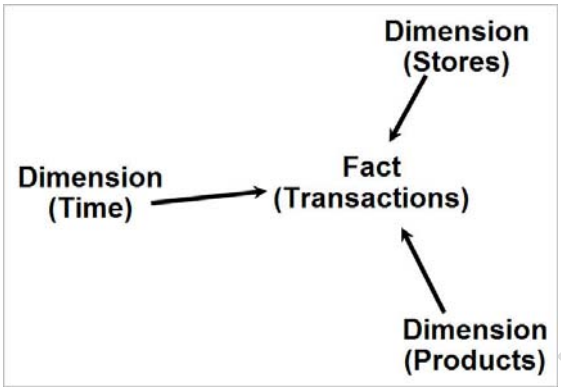


Figure 2.5 Star schema depicting fact and dimension tables

Star schema is the preferred schema to implement a data warehouse; it leaves the database in the second normal form (2NF). However, there are other analytical-oriented design techniques, such as the snowflake schema, which leaves the database in the third normal form (3NF).

In this book, we will focus on the star schema and we will use figure 2.4 to explain more about dimensions, measures and facts.

2.3.1 Dimensions

A **dimension** gives context to information thereby making it useful and meaningful.

For example, when we say profit margin of an electronic store is 'x', it gives us no usable information that can be used for any real purpose. Alternatively, saying profit margin of an electronic store is 'x' for "Electronic" products during the year "2008" give us meaningful and useful information. In this example, products and time are two dimensions (or contexts) for which value 'x' (profit margin) indicates measurable performance.

Dimensions have at least one **level** and one **hierarchy**.

A **hierarchy** expresses a chain of **levels** in such a way that one level must be the "parent" of the preceding level. For example, a time dimension typically has columns that we can

consider as levels of a hierarchy within that dimension, such as “Year”, “Quarter”, “Month” and “Day”. Notice that one day belongs to one month and one month only. When creating a hierarchy, it is mandatory that exists this **parent-child relationship** within each pair of levels within that hierarchy, like “Year” & “Quarter”, “Quarter” & “Month” and “Month” & “Day”.

So this “Calendar Hierarchy” considers DAY→MONTH→QUARTER→YEAR. These are contexts that help us analyze the information we have.

In case a dimension must include other “contexts” that does not fit to a given hierarchy, but shows a parent-child relationship with any column of that dimension table, then we are able to define a new hierarchy within that dimension. For instance, if we include a new column “WeekDay” in that time dimension, this new column does not fit the existing hierarchy, as a weekday can not “roll up” to months or any other column we have. However, weekday shows a clear parent-child relationship with column “Day”.

So we can define a second hierarchy within this dimension, which we can name as “Week Day Rollup”. This new hierarchy has two levels only: DAY→WEEKDAY.

Another example of a hierarchy within the time dimension is “Fiscal Year”. Although it has similar levels as the hierarchy “Calendar Year”, the start and end dates of each Fiscal Month and/or Fiscal Year could be different from the ones considered for Calendar Months and Calendar Years. As you can notice, there is no direct parent-child relationship within Calendar Months and Fiscal Months, for instance, and therefore these new columns (FiscalMonth, FiscalYear) have to be part of a separate hierarchy.

2.3.2 Measures

Measures are business parameters that indicate business performance in given context. In our example “Profit margin” (or “percentage profit margin”) for given set of products, this measure will indicate which products (or product category) have high profit margin and which have low profit margins. Based on this data, business owners/analysts either might decide to stop selling low margin products or might create a product campaign for those product lines.

2.3.3 Facts

As the name suggests, “**fact**” is the lowest level of business transaction data that forms the basis of calculating “Measures”. Profit margin of any product sold is solely based on the

actual sale that happened at the store. Once the sale is made, there is no way this fact can be changed. We might debate that facts and measures are the same thing; however, that is not true. We can have a measure like “Average electronics items per Bill”. To get this information we will have to count total number of “Electronics” items sold and divide it by total number of bills generated. In this case, we will be using two different facts (electronic item count and bill count) to calculate one measure. A table that stores data, which is used to calculate measures, is called a fact table. In real-world data warehouses, the fact table alone usually responds for more than 95% of database size.

2.3.4 Time series analysis

Time series analysis considers the fact that data samples captured over time may have internal relationships like trends, correlations, variations etc.

Time series analysis [23] is applicable to knowledge domains, where data generated is primarily a function of time, for example, energy usage analysis, sales forecasting, share trading analysis etc. We will not be covering details of time series analysis in this book.

2.4 Looking for Performance

Data warehouses keep growing in size with time and as the size increases, it becomes very important to keep performance of the database within acceptable limits. Having better/faster hardware surely helps in reading the disk faster; computing large calculation and table joins etc. However, this is not the solution to all performance related challenges. Figure 2.6 represents a monolithic database with data as various shaped objects.

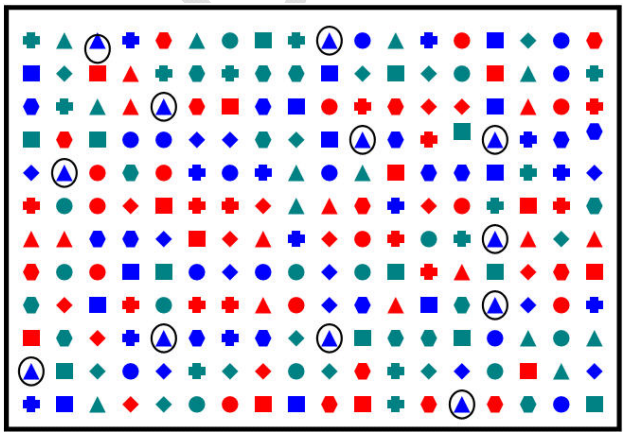


Figure 2.6 Monolithic databases (data represented by different shapes and shades)

Data objects indicated by circled objects in figure 2.6 above are the objects, which will be selected and returned as a result to the query under execution against database. In this case, irrespective of amount of data returned, database manager will have to scan the entire table to find the objects matching the query under execution. **IBM DB2 Enterprise Edition** database server supports following types of database optimizations:

- Indexes
- Database partitioning
- Table partitioning
- Clustering
- Materialized views

There are several different DB2 editions and Enterprise Edition is the most complete one, with all existing DB2 features. Even so, some of those features mentioned above are also available in other DB2 editions. For more information, please refer to http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.licensing.doc/doc/r0053238.html

2.4.1 Indexes

Indexes provide first level of performance enhancement by keeping index column values sorted and stored in a tree form with the row ID (or RIDs) address/pointer. During retrieval, the index tree is scanned to find values in the tree, and then all the relevant rows as pointed by RIDs are retrieved. This avoids the scan of the entire table.

2.4.2 Database Partitioning

Database partitioning, also known as **vertical partitioning** is a method of distributing a table by distributing its data based on a column. Partitioning can happen either by column value of by value hashing. Even though it has same volume of data to be scanned, it can happen in parallel. Figure 2.7 shows a database with three partitions P1, P2 and P3. In this case, we can retrieve the data in one third of time as compared to that with non-partitioned database (assuming parallelization overhead to be negligible).

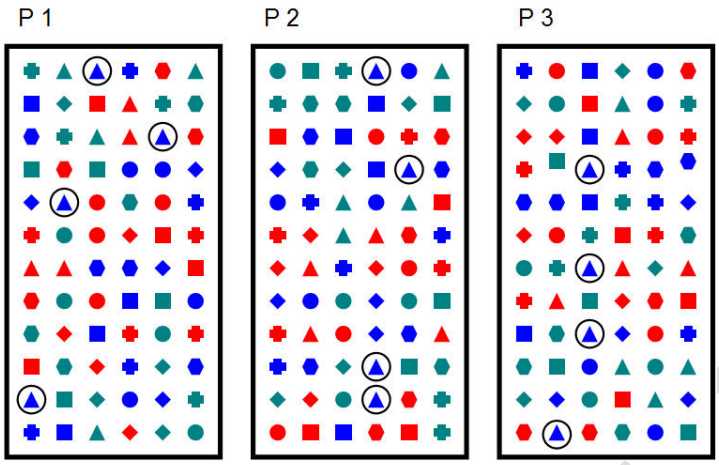


Figure 2.7 Partitioned database with three partitions

Database partitions as shown in figure 2.7 can exist either on the same physical system or on separate systems. Partitions here are referred as "**logical partitions**".

2.4.3 Table Partitioning

Table partitioning, also known as **horizontal partitioning** is a method of distribution of table by splitting data based on data value ranges. Figure 2.8 shows a partitioned table split by date.

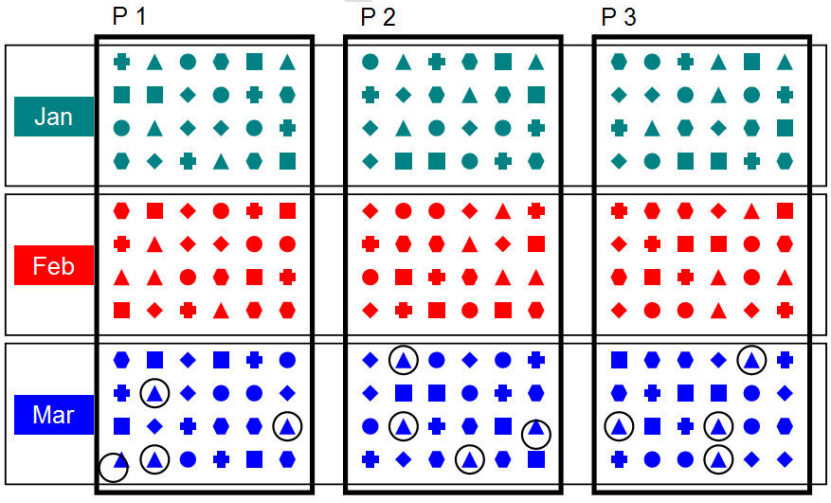


Figure 2.8 Partitioned table on a partitioned database

Partitioned table in a partitioned database as shown in Figure 2.8 reduces the data scan size to one third in each partition. Assuming negligible overheads, we can read this data nine times faster (three times performance benefit due to database partitioning and each partition having three times performance due to reduction in data scan size).

2.4.4 Clustering

Next level of performance enhancement is achieved via data clustering. **Clustering** means storing similar data together on the disk. Doing so, sequential disk reads could be maximized, therefore providing better read performance. Figure 2.9 shows a database with clustered data. A physical disk has single head for reading and writing. To read data that is stored in fragments over the disk, read/write head will have to move to various disk sectors to "pinpoint" the requested data. When using clustering, the head will find data in close sectors of the disk, significantly reducing the time to execute the same operation.

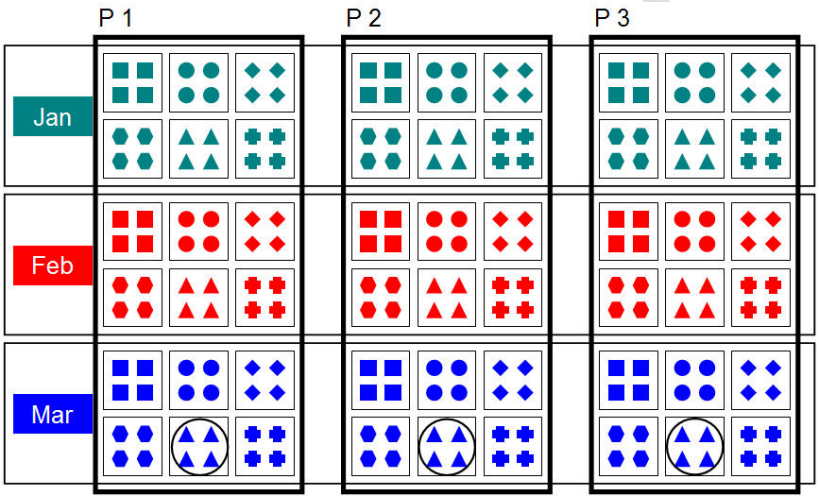


Figure 2.9 Data clustering in a partitioned table with database partitioning

Data clustering as shown in Figure 2.9 uses block indexes instead of row indexes, therefore system has less number of indexes to read in order to reach the correct data block to start reading. Data clustering can be done on one or more table columns. Multi-column clustering is also termed as **Multi Dimensional Clustering** or **MDC** in short.

2.4.5 Materialized Views

A materialized view is a view with “stored” data. Unlike a view, materialized view caches the result set in a permanent storage device. This result set is used for any future references in any query that can be satisfied by using subset of the materialized view data. Materialized views in DB2 are termed as **Materialized Query Table** or simply **MQT**. A MQT is always transparent to the application issuing the query and therefore query performance of a database can be improved significantly by creating them without any changes to data/table layout. Typically MQT significantly reduces the data reads, however if the base table itself has several billions of rows, then MQT will also end up having few hundred million rows. This will surely provide better response time, however it might not be within acceptable time limits. To improve the query time further, materialized views can also be used together with any of the performance enhancement techniques discussed above.

2.5 Summary

Data warehouse is a large repository of data and therefore special care is required during database architecture and design activity. Depending on the business domain (e.g., banking, retail, banking, finance) suitable logical and physical data model is selected. Once physical data model is identified, appropriate database optimization techniques should be applied. Although all the optimization techniques discussed above can be applied to any database tables, it is not recommended to use them all at the same time, unless there is a need for it.

2.6 Review Questions

1. Which of the following is called vertical partitioning of the database?
    A. Database partitioning
    B. Table partitioning
    C. Multi Dimensional Clustering
    D. Materialized Query table
2. Which of the following is called horizontal partitioning of the database?
    A. Database partitioning

B. Table partitioning

C. Multi Dimensional Clustering

D. Materialized Query table

3. Which of database schemas below are suitable for data warehouses?

   A. Star schema

   B. Snowflake schema

   C. Second normal form

   D. Third normal form

4. Which type of architecture is suitable for production/commercial systems?

   A. One tier

   B. Two tier

   C. Three tier

   D. Four tier

5. “Levels” and “Hierarchies” are attributes of

   A. Dimensions

   B. Measures

   C. Tables

   D. Facts

6. Can vertical and horizontal data partitioning be implemented simultaneously?

   A. Yes

   B. No

   C. Depends

7. Aggregations (sum, average, count, etc.) are applicable to which of the following OLAP objects?

   A. Facts

   B. Dimensions

C. Measures

D. None of above

8. What is the relationship between OLAP structures (facts, dimensions) and database storage structures (vertical/horizontal partitioning, MDC)?

   A. OLAP structure has dependency on database storage structure.

   B. Database storage structure has dependency on OLAP structure

   C. There is no relation at all

   D. There is no relation, but changes in one impacts performance of other

2.7 Exercises

9. Give two examples each for tables for which horizontal, vertical partitioning and multidimensional clustering is suitable.

10. Implement a table with only vertical table partitioning and compare the select performance with non-partitioned table.

11. Implement a table with only vertical and horizontal table partitioning and compare the select performance with non-partitioned table.

12. Implement a table with only vertical, horizontal table partitioning and MDC, and compare the select performance with non-partitioned table.

13. Implement any dimension table and compare its select performance with transactional system (multi-table join).

3

Chapter 3 – Hardware Design Considerations

In this chapter, you will learn about:

- Storage design considerations
- Network design considerations
- OS configuration and tuning

There are two primary situations to talk about hardware configuration and design:

- When setting up a new data warehouse.
- When reconfiguring and/or tuning an existing data warehouse.

We will start with how to tune an existing infrastructure and follow with conclusions. Then we will cover a new data warehouse hardware setup, requirements, and its configuration.

3.1 The Big Picture

A high performance data warehouse system is always the result of the combination of good software and hardware design. In **Chapter 2**, we discussed software design aspects and, in this chapter, we will make sure that the hardware is configured properly to support a good design.

3.2 Know Your Existing Hardware Infrastructure

In order to tune and reconfigure, first requirement is to know your existing infrastructure. Here is the checklist that should be completed as the first task of the redesign activity.

- Number of CPUs (Cores) and their type: like p6 570, 4.7GHz, 4 cores or Intel Xeon 2,1GHz, 4 cores
- Total memory available to these CPUs (Cores): like 8GB/16GB/32GB etc
- Storage type, capacity and availability mode: such as External storage DS4800, 10TB, SAN based
- Disk size of the disk used in above storage.
- Number of disk arrays or LUNs.
- Number of network adapters available per physical server.
- Number of optical fiber or SCSI adapters available per server.

Note:

The use of internal IDE disks should NOT be used at any circumstance for database storage due to performance reasons.

In Figure 3.1, there is an example of such a checklist.

Hardware Component	**Count / Info**
1. xSeries server	x3655 (Pentium 4)
Operation system	Windows XP
CPUs (Cores)	2
Memory	2GB
Network ports and speed	1 (100Mbps)
Fiber ports	1
2. xSeries server	x3650 (Xeon dual core)
Operation system	SuSE Linux 10
CPUs (Cores)	4
Memory	8GB
Network ports and speed	2 (100Mbps, (1Gbps))
Fiber ports	2
3. xSeries server	x3650 (Intel core 2 duo))
Operation system	Windows 2003 Server
CPUs (Cores)	4
Memory	4GB
Network ports and speed	1 (100Mbps)

Fiber ports	2
4. xSeries server	x3650 (Xeon dual core)
Operation system	Windows 2008 Server
CPUs (Cores)	4
Memory	8GB
Network ports and speed	4 (1Gbps)
Fiber ports	4
5. pSeries server	p6 570 (p5 processor)
Operation system	AIX 5.3
CPUs (Cores)	4
Memory	8GB
Network ports and speed	2 (1Gbps)
Fiber ports	2

Figure 3.1 Existing hardware list with details

Lets' assume figure 3.1 to be our current infrastructure where we run several instances of our Relational Database Management System (RDBMS). We will try to converge to a new architecture with one database instance that is easily scalable. In addition, as we do not have any performance requirements here to meet, we will reconfigure this hardware to an optimal and generic form.

3.2.1 Know Your Limitations

Any redesign activity requires a list of activities and options that might not be either practical or possible in the existing context. List of such limitations is quite long; here is a subset of that list for our reference:

- Budgetary constraints
- Time available for design/redesign
- Resources (skills/people/hardware) available
- New resources that can be induced (skills/people/hardware).

For our current redesign activity let us assume we have unlimited budget, time and resources.

3.2.2 Identify the Bottlenecks

The first thing we need to look for in the system is a case of inconsistencies and non-uniformities. The basic rule here is that all physical and logical database partitions should be exactly the same in terms of Operating System, CPU, memory, storage, file systems. In Figure 3.1, we can see that the current infrastructure is quite heterogeneous. Servers have different number of CPUs, network adapters and optical fiber adapters. Even if the data is perfectly distributed across all servers, some servers will be heavily loaded and others will have quite less load due to our hardware non-uniformity.

3.3 Put Requirements, Limitations and Resources Together

By now, we have complete list of requirements, our limitations and the resources available for our redesign activity.

3.3.1 Choose Resources to Use

In Figure 3.1, we know that we have five servers available for this redesign activity. Assume that current warehouse infrastructure uses any one of the servers as warehouse and there is a need to redesign it to make it a clustered database with two servers. From close review of the information, we can see that Server 2 and Server 4 are the best match in terms of CPU, Memory, Network and fiber adapters.

3.3.2 Make Changes in Hardware to Make All Servers Homogenous

The differences in selected servers are as follows:

A. Only one Gbps network adapter in Server 2

B. Server 2 with just two fiber adapters (for storage connectivity)

For network adapter difference, we do not need to make any change as we can use 1Gpbs network adapter on both servers for warehouse servers' internal communication/data transfer. For fiber adapters, we will add two more fiber adapters to Server 2 (remember our assumption that we have unlimited time, money and resources available). In case addition

of adapters is not an option, we can limit our usage of fiber adapters on Server 4 as well to 2 fiber adapters.

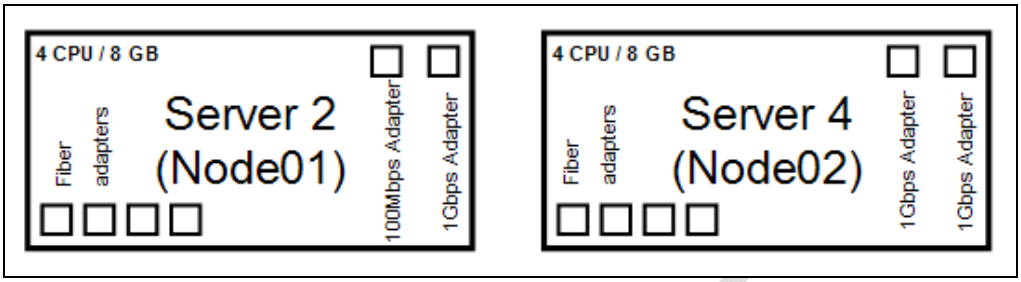


Figure 3.2 Servers with high-level hardware components

Figure 3.2 shows the final outlook of server we plan to use for our redesign activity. The next step is to define network and data connectivity. In the later sections, we will refer Server 2 as Node01 and Server 4 as Node02. In distributed databases, it is very important to provide a good speed network between the database nodes in order to process a query fast.

3.3.3 Create a Logical Diagram for Network and Fiber Adapters' Usage

Assuring we have same CPU, Memory, Fiber adapters and network adapters we guarantee that both nodes have equal processing power and I/O capability.

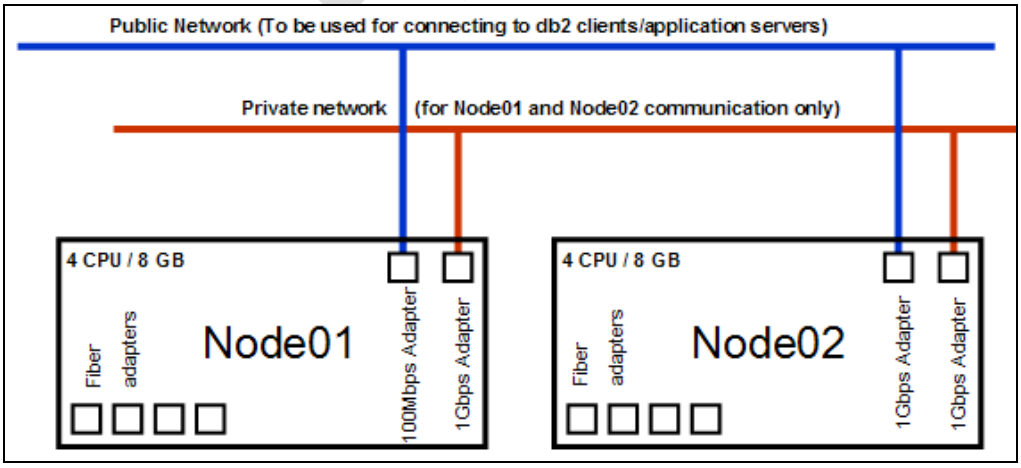


Figure 3.3 Servers with multiple network connectivity

We will use 1Gbps network adapter on each node for inter node communication. This network is also called as the private network or DB2 FCM (fast communication manager) network and 100Mbps network adapter will be used for connecting to application servers, this network is also known as public network because it is shared and used by other applications.

3.3.4 Configure Storage Uniformly

Server configuration created above would provide its peak performance only if storage is configured to support maximum I/O and read/write parallelism. Let us assume we have 20 disks (all with same capacity) available with storage controller. The task we have in hand is to use these 20 disks in the best way possible by making 10 disks available to each Node.

These 10 disks can be made available to Node01 and Node02 in different ways. If disk high availability is not required then all the 10 disks can be exported as 10 LUNs, however that is not recommended.

A very interesting way to use several disks as a single logical unit is known as Redundant Array of Independent Disks or RAID. RAID provides scalability and reliability based on redundancy. There are several different architectures that one can achieve using RAID, but when it comes to database storage, the preferred ones are RAID5 and RAID10. Simply put, RAID5 provides parallel processing by striping data to several disks. RAID10 also does the striping, but adds mirroring to the architecture. That being said, we leave to the reader the task to search for detailed information about the different RAID architectures.

In our current solution, we will use RAID5 (four 4+p arrays) which will result in two 4+p arrays per Node.

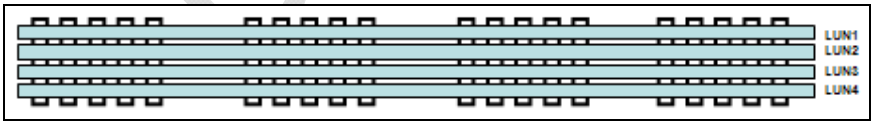


Figure 3.4 LUN structure and striping over multiple storage arrays

Typically, most storage administrators will create LUNs as several stripes over disk arrays as shown in Figure 3.4. However, it is not recommended to configure storage for a warehouse in such way. Why? Because if LUNs are configured the way shown in Figure 3.4 then the database manager will attempt to write data to four different file systems (mapped to LUN1, LUN2, LUN3 and LUN4) in parallel. Even though logically we are writing to different file systems, it is highly probable that it might be writing to same disk (due to

striping). All disks have single head for read or write. So only one read or write can happen in parallel resulting in I/O wait.

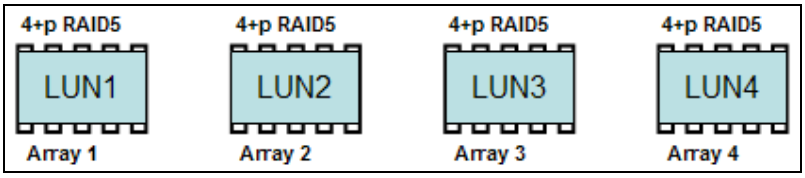


Figure 3.5 LUN structure and striping over single storage array

The recommended method to create arrays and LUNs is shown in Figure 3.5. Complete RAID5 array of 4+p is exported as one LUN, which should be used completely to create one file system. These LUNs can be made available either via direct attach or via a SAN switch.

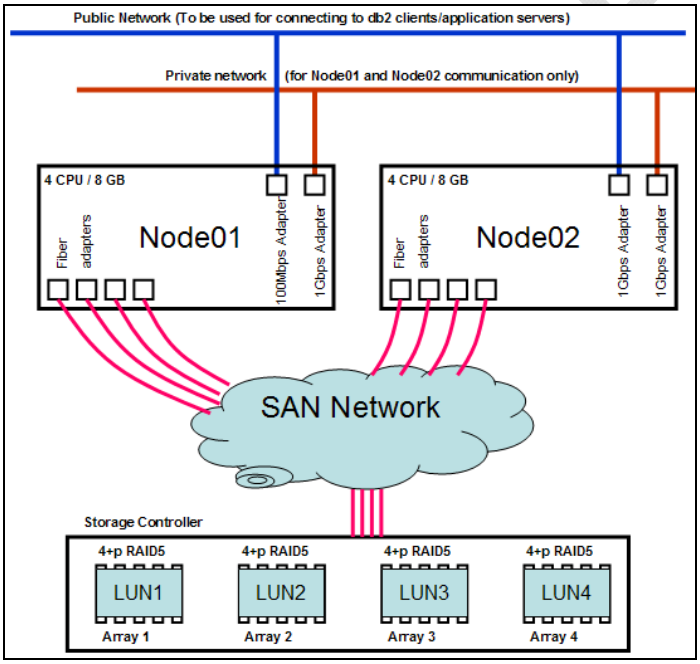


Figure 3.6 Storage configuration and connectivity via SAN network

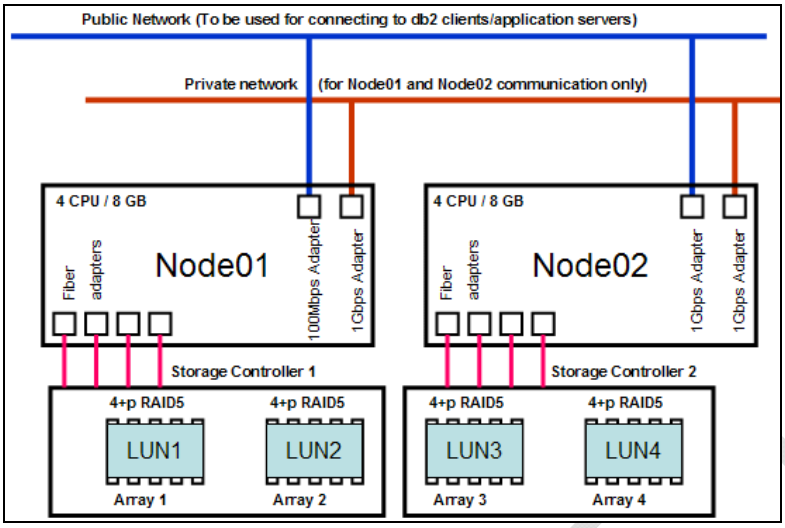


Figure 3.7 Storage configuration and connectivity for direct attach

In case of direct attach approach, we would need one storage controller per node as shown in Figure 3.7

3.4 Summary

In this chapter, we discussed hardware design considerations in terms of processing power (CPU), memory (RAM) and storage. Hardware architecture and design decides maximum peak performance that a data warehouse can achieve. Software solution design activity on top of this infrastructure tries to reach this peak performance by deploying suitable data model and appropriate database performance objects.

3.5 Review Questions

1. What all hardware component any database server must have?
    A. Internal storage
    B. External storage
    C. Network interfaces
    D. Fiber channel interfaces
    E. SCSI disk interfaces

F. CD Drives

G. USB interfaces

H. Firewire interfaces

2. What are typical constraints in any data warehouse redesign/implementation?

   A. Time

   B. Money

   C. Resources

   D. All of above

3. What are possible bottlenecks for a data warehouse?

   A. CPU

   B. Memory

   C. Storage

   D. All of above

4. Data nodes of a distributed data warehouse should be uniform in terms of hardware specifications.

   A. True

   B. False

5. How many LUNs per disk array give best performance?

   A. One

   B. Two

   C. Three

   D. It is independent of umber of LUNs

6. Which is not suggested mode to connect storage to server

   A. Network

   B. Storage Area Network (SAN)

   C. Direct Attach

7. How many network interface a database server should have at minimum?
    A. One
    B. Two
    C. Three
    D. Four

3.6 Exercises

8. Read more on various types of RAID strategies with advantages and disadvantages/overheads for each.
9. Implement a server with following types of storages
    A. internal disk drive
    B. external storage (raid5)

    and compare the read/write performance.
10. Configure two servers with
    A. 100Mbps network
    B. 1Gbps network

    and compare network throughput.
11. Create multiple file systems on a server on
    A. arrays with 1 LUN
    B. arrays with multiple LUNs

    and compare concurrent read/write performance.
12. Create two file systems on a server on
    A. RAID1 array
    B. RAID5 array

    and compare concurrent read/write performance.

4

Chapter 4 – Extract Transform and Load (ETL)

In this chapter, we will talk about the ETL component of a data warehouse. Some data warehouse experts place ETL components outside of warehouse boundary, as well as source system (transactional system). However here we are not going to debate on its placement. All we want to focus here is “how ETL integrates data warehouse and source systems seamlessly”.

4.1 The Big Picture

Consider a company having several decades of history & operations in multiple domains and industries. Over a period, the company went through various changes in both internal operations and external customer interactions. Moreover, to achieve these changes, the company acquired several specialized tools and systems from different technological background and vendors. As a result, company ended up in having various disconnected systems having silos of data, for example, payroll department using SAP, marketing department with custom build solution, HR department using Oracle solution, finance, sales and few other systems running on IBM solution stack (Windows, Unix and Mainframes).

Having these kinds of isolated systems from various vendors is a quite common scenario in any organization. However, this introduces the challenge of creating a holistic view of company as the data to calculate various performance metrics reside in these discrete and disconnected systems. There are two ways to address this challenge:

- Pull data from all these systems manually, then analyze via manual means. This method is good only if the analysis is needed once in a while, however if some

reports are required every day at close of business, every week, every month etc. then for all practical purposes manual analysis is not a workable solution.

- Pull data from all these systems via automated means, convert data in a format that supports analysis, and save it for future references/reuse.

The second solution above is what we call **Extract**, **Transform** and **Load** (ETL).
Commercial off-the-shelf ETL suites like IBM DataStage or Informatica provide tools to extract data from different systems (Windows/Unix/Mainframes), cleanse and transform, and finally merge/load into data warehouse as shown in figure 4.1.

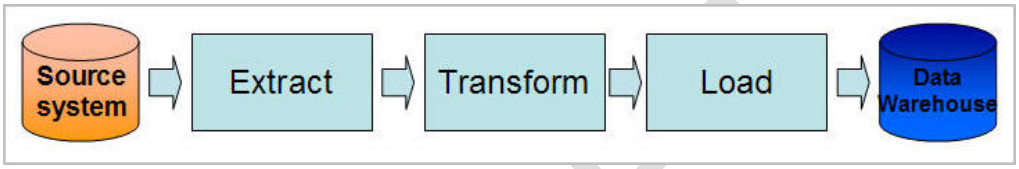


Figure 4.1 High level ETL view

With the advent of much better, faster and efficient databases, some experts even advocate for ELT (Extract Load and Transform) way for bringing data into warehouse from source systems. However, in this chapter we will not discuss or conclude anything on ETL vs. ELT and keep the focus on ETL only.

4.2 Data Extraction

Data extraction, as the name suggests, is the process of pulling data from different source systems, be it flat files or commercial databases like IBM DB2, Oracle, SQL Server, MySQL, etc. Data to be extracted may be pulled partially, incrementally or in its entirety depending upon business requirement. Storage space where this extracted data is stored is referred as **staging space** or **staging area**. Staging area can be a database storage or operating system file space. Sometimes a few tables exist only in staging area. These tables typically help in resolution of various source or target table mappings. We will see a real life usage of staging tables while working on our case study in **Chapter 6**. Some latest ETL tools provide extraction of unstructured data, XML data, voice/video data, streaming data, data from web servers etc.

4.3 Data Transformation

Once data is available in staging area, we can perform appropriate transformation on it so that it can be loaded in the target system. Consider the following example: in a company, the HR system stores an employee name in two different columns “FirstName” and “LastName”, but payroll system stores this name in one column only, called “FullName” (having first name and last name together separated by space).

Now in order to combine employee information from these two systems we either need to join “FirstName” and “LastName” from HR system into one single string (if target data warehouse needs it in single string). Or the other option is to split out column “FullName” from payroll system into columns “FirstName” and “LastName” (if target data warehouse needs it as two strings).

4.3.1 Data Quality Verification

During the transformation phase, the extracted data is submitted to several validations. Checks and conversions are performed, like data format, data width, case and data type conversions, data lookups, data splitting and joining, null value handling etc.

If the data does not pass the quality criteria, it is rejected and information about it is recorded in a log or a table for later reference, update or review.

This verification is very important as it prevents invalid data from entering into data warehouse and eventually stops it from participating in business performance metrics.

IBM QualityStage is one of the industry leader products for handling data quality during transformation stage. IBM QualityStage is part of IBM Information Server software. QualityStage helps organizations ensure that their strategic systems including data warehouses, ERP, CRM and legacy systems deliver accurate and complete information to business users across the enterprise. Equipped with trusted information, organizations can make timelier and better-informed decisions. This module includes an easy-to-use Graphical User Interface (GUI) and capabilities that can be customized into specific company business rules, control over international names and addresses, phone numbers, birth dates, email addresses, and other descriptive and comment fields. It discovers relationships among them in an enterprise and Internet environment, batch and real time. As a result, companies gain access to accurate, consistent, consolidated views of any individual or business entity and its relationships across the enterprise.

4.4 Data Load

After data has passed through quality checks and required transformations, it is ready for loading into the data warehouse. Once data is in a data warehouse, all subsequent reports after data load will reflect data with updated information. Figure 4.2 shows data loading in a partitioned database in massively parallel computing environment.

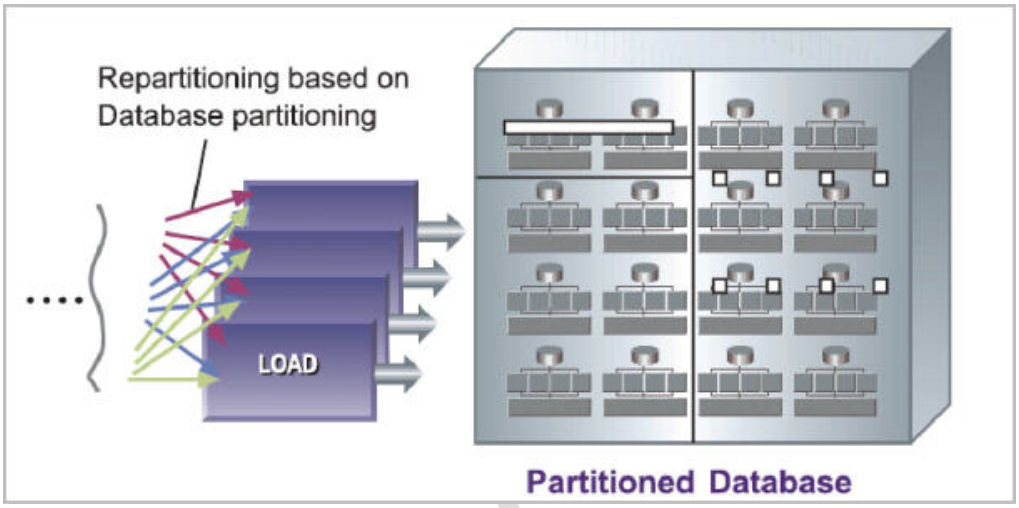


Figure 4.2 Data loading in partitioned database

Commercial off-the-shelf ETL tools support data loading in sequential and parallel modes. They also support all major databases as target databases (data warehouse) and leverage database specific performance features (if any).

4.5 Summary

A scalable infrastructure should provide native, high-performance parallel components, in particular sorting, aggregation, joins, and so on. However, because any large enterprise has special and customized needs, a scalable infrastructure should be extensible in order to integrate existing programs and third-party tools as part of the information integration process. These programs, originally written to execute sequentially, should be able to execute in parallel on a partition of data, regardless the programming language used (C, C++, COBOL, etc.).

In order to integrate existing software code, a key requirement is the ability to operate only on the data (i.e. the columns of each record) and let the infrastructure simply pass the rest

of the unused (i.e. data that was not touched and/or changed) through the component to the next downstream component in the data flow. This is generally referred to as column or schema propagation. This is a critical aspect in order to integrate existing applications without change, making them more portable and useable. With this ability, software can be integrated and parallelized. Figure 4.3 shows end-to-end ETL flow with parallelism in every phase of the process.

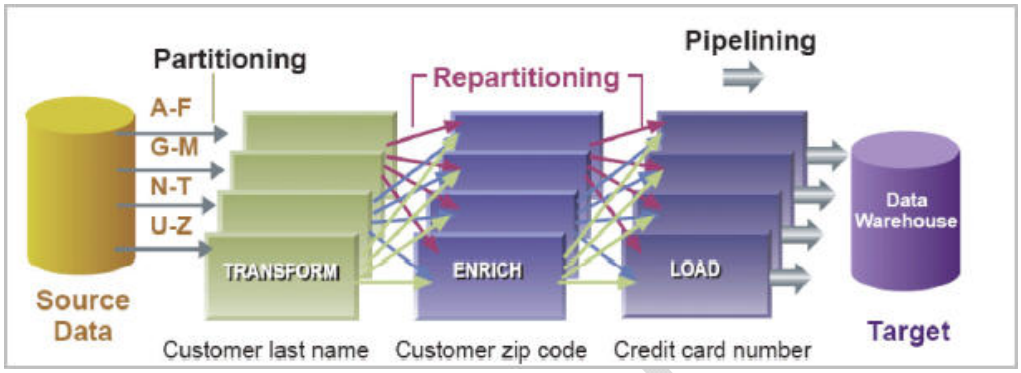


Figure 4.3 Typical extract, transform and load process in parallel environment

While selecting the right tool for ETL processing, various considerations should be made. The tool should support all current data source types that an organization has. In addition, it should have capability for future extension, a wide function set for data quality validation and transformation. It should also support data loading in parallel for MPP and cluster based databases. Figure 4.4 shows IBM Datastage capabilities in brief.

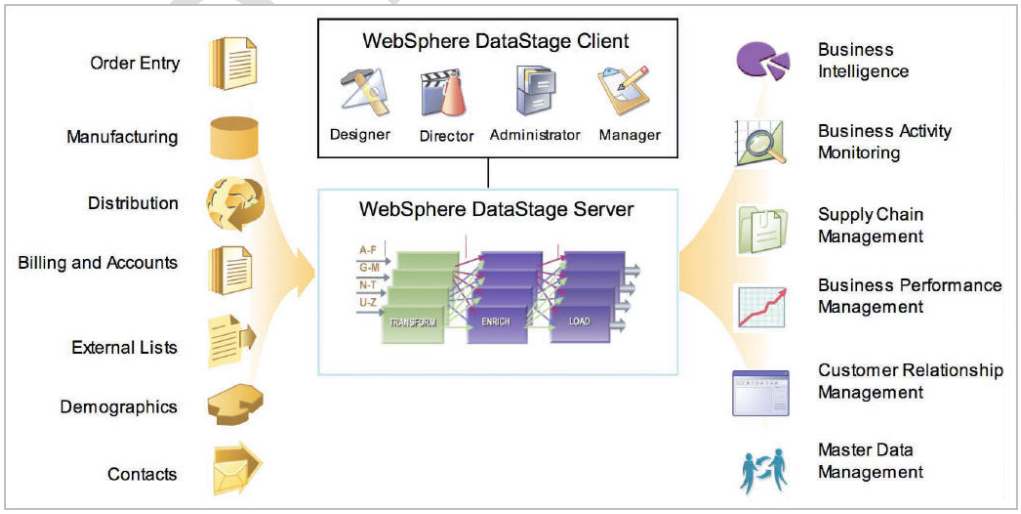


Figure 4.4 IBM Datastage product overview

IBM Datastage product overview depicted in figure 4.4 above is only a high level view .

4.6 Review Questions

1. What does ETL stand for?

2. Can all ETL operations be achieved using SQL statement only?
   A. Yes
   B. No
3. To connect to multiple data sources, are multiple ETL tools required?
   A. Yes
   B. No
4. Can a flat file (comma delimited) also be a valid data source and destination?
   A. Yes
   B. No
5. Can data be loaded into a single table via multiple parallel loads?
   A. Yes
   B. No
6. Can ETL tools read and write streaming data?
   A. Yes
   B. No
7. What is the storage space used during data extraction called?
   A. Temporary area
   B. Temporary storage
   C. Staging area
   D. Staging storage

8. Which following operation does data transformation include?

   A. Format conversion

   B. Data size conversion

   C. Data type conversion

   D. NULL value handling

   E. All of above

9. For high performance data loading in partitioned environment, it should be executed in

   A. Sequential mode

   B. Parallel mode

   C. Mixed mode

10. Which IBM product is suitable for ETL processing?

    A. IBM Datastage

    B. IBM QualityStage

    C. IBM Information Server

    D. All of above

4.7 Exercises

11. Create an ETL data flow using any tool or SQL statements, which extracts "Timestamp" from source system and stores it in three target columns as "Year", "Month" and "Day"

12. Create an ETL data flow using any tool or SQL statements, which extracts "FirstName" and "LastName" from source system and stores it as "FullName".

13. Create an ETL data flow using any tool or SQL statements, which reads "Customer number" from two source systems. First source system stores "Customer number" as a string of five characters long and second source system stores it as string of seven characters. Target system should have all customer numbers as string of seven characters long. First source system's values should be appended with "00" on left to make it seven characters in length.

14. Create an ETL data flow using any tool or SQL statements, which reads data from a flat file and a database, joins the result set on a common column, and loads this data into staging area having data of “full outer join” between the two data sources.
15. Create an ETL data flow using any tool or SQL statements, which reads data from a flat file having date in format “MM/DD/YYYY”, converts the format to “DD-MMM-YY”, and loads it into target database. This flow should extract and convert incrementally added source data as well to the target instead of processing whole table during every execution.

5

Chapter 5 – Using the Data Warehouse for Business Intelligence

“Even the best judge cannot come to the right decision if he’s not aware of all the facts”

Organizations need to know all the critical information and facts when business decisions need to be made. These decisions need to be based on experience and current market trends. Business intelligence tools will help the organization to achieve this.

There are quite a few challenges in knowing the right information at the right time:

1. Typically, in any organization there are multiple systems available, each taking care of different activities for the organization. For example, a company might have a Human Resource Management System, Finance, Enterprise Resource Planning, and a Customer Relationship Management system. Data is scattered among the different systems, and it is a problem to gather related information from all these systems and present it in a holistic view to the user.
2. Once you cross the first hurdle, you are now dealing with a huge set of data in a repository, which you need to mine to fetch the right information.
3. This fetched information needs to be summarized and presented to the user in the most intuitive fashion with the help of charts, graphs, tables and reports.
4. At last, after preparing these reports, it is important to be able to send them to the user through various means, including email, mobile, and dashboards.

The domain of applications that deal with and provide solutions to the above challenges is called **Business Intelligence (BI)**.

5.1 The Big Picture

BI is not something new nor did it come suddenly. The roots of BI applications lie in the Decision Support Systems (DSS). To analyze data in the early days, organizations had their own decision support systems. These systems were built on top of their existing databases. Decision support systems provided the user various ways of reporting information by making use of graphs and charts. These DSS's had quite a few limitations:

- The very common problem with all these DSS's was that they were very tightly coupled to the business data model they were representing, For example, if data model is changing very frequently then DSS also needs to be updated.
- In addition, this information was managed in an OLTP system, where information related to all the transactions is stored. Generating reports on top of these kinds of systems usually causes performance issues both for the DSS and OLTP system as they share the same infrastructure. Besides, searching a normalized database (3FN) requires many table joins and it is time consuming.
- Business rules change often and the need to enhance the DSS was quite frequent.

Today's DSS is just one part of the Complete BI solution.

Formal Definition of BI:

"Business intelligence is all about extracting information from the data collected by various systems within your organization and presenting this information in an intuitive fashion to the business experts through various channels like email, mobile, and dashboards so as to provide the capability of making intelligent business decisions."

Therefore, any BI application must have these capabilities:

1. Ability to fetch data from a single or multiple data warehouses.
2. Ability to summarize this data and be able to present it in multiple formats like tables, graphs, and charts.
3. Dashboards are the latest addition to BI applications which provide the capability to view the information in real-time. This information is very critical in monitoring organization/business performance in real-time.

4. The BI application should be capable of sending these reports via different channels to the user, including print, email, mobile, dashboard, or alerts & notifications. The application should be able to send these on some defined schedule.

Figure 5.1: The Business Intelligence Data Flow

As shown in *Figure 5.1*, in order to obtain valuable reports and charts, all business data must be collected and then stored in a structured manner.

Benefits of BI

After successfully implementing Business Intelligence, enterprises can make business decisions quickly, which gives them an edge over their competitors. Some of the benefits of BI implementations are the following:

- Reporting, querying and analysis of data are simpler and data consolidation is easier.
- The same reporting standard can be maintained across all the reports created.
- Data stored in other systems (like ERP and CRM, that have limited reporting capabilities) can be used to produce new reports, crossing in a single report data from different sources.
- Reports can be generated in different formats as per user requirements.
- Sending reports to the users via email can be automated and hence, reduces manual intervention.
- Since the reports can be available over the web, the cost of printing and distributing paper reports can be saved.

5.2 Business Intelligence Tools

Business Intelligence tools (or BI tools in short) are specialized software tools that use BI terminology and interactively facilitate creation of various domain independent objects (like facts, dimensions, measures, metrics, reports, dashboards, etc.).

One of the most popular BI tools in the market is IBM Cognos. IBM Cognos comprises of many tools, all performing some specific purpose:

1. Extract Transform Load (ETL) Tool – IBM Cognos 8 Data Manager
2. Data Modeling Tool – Framework Manager
3. Reporting Tools – Report Studio, Query Studio & Analysis Studio.
4. Measuring KPIs – Metric Studio
5. Event notification – Event Studio

5.3 Flow of Data from Database to Reports and Charts

1. Database Modeling: defining the warehouse model and creating the physical objects (metadata is generated in this phase).
2. ETL: this process is used to populate the Data Warehouse from various types of databases.
3. Data model: abstracting Data Warehouse metadata and making it available for report generation. This is the roadmap on how to find information in the Data Warehouse.
4. Query, Reporting & Analysis: This process works on top of the data model to deliver the information in form of reports/tables & charts.

Figure 5.2 below show the data flow in a graphical form.

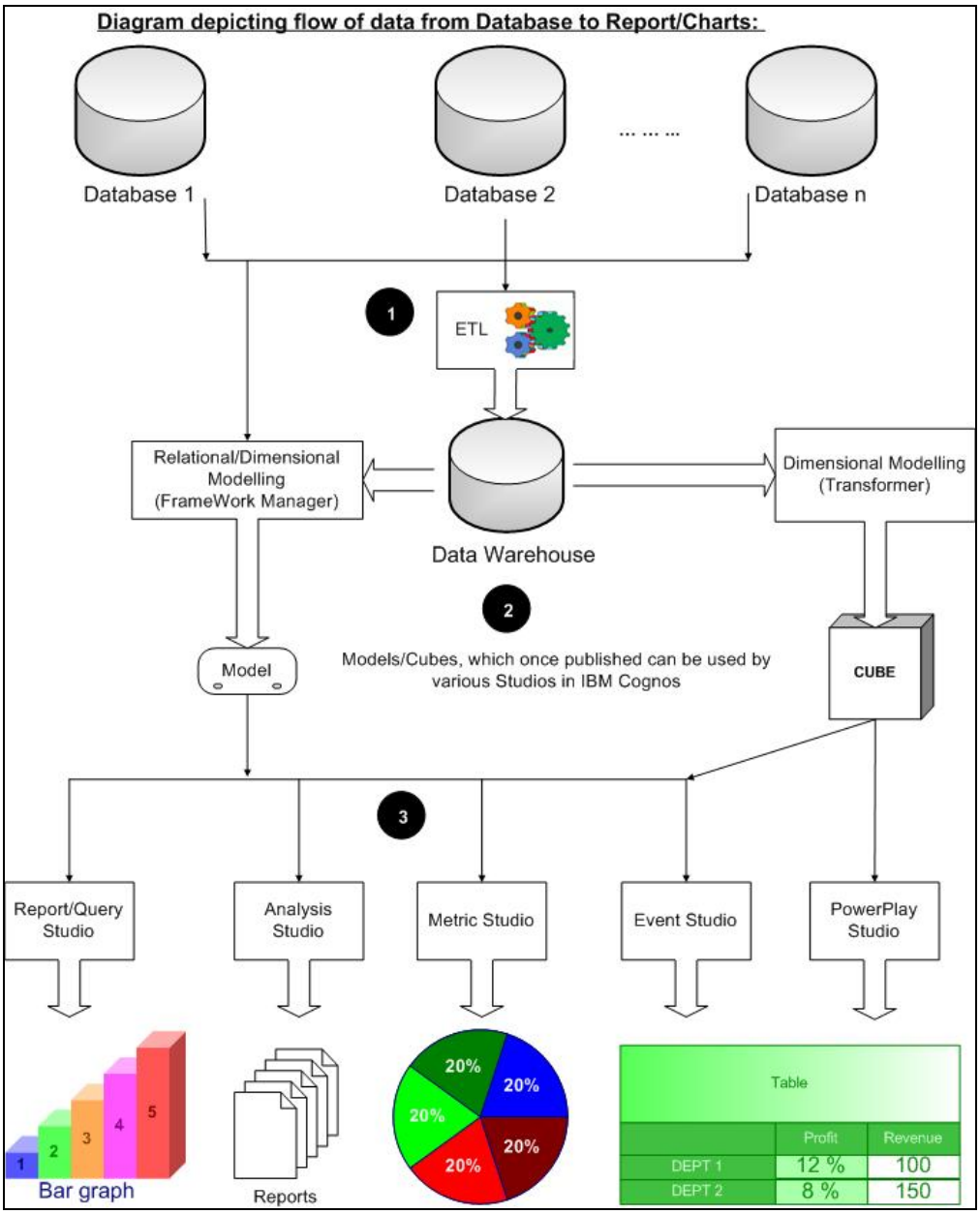


Figure 5.2 Data flow from database to reports and charts

5.4 Data Modeling

To create effective business reports, the main requirement is to make sure that all the data objects required for business are correctly and completely represented. This representation is known as a data model.

For Business Intelligence systems, a data model is critical for simplifying the reporting and analysis processes. It gives control over how enterprise data is presented to the end user. Well-modeled, centralized data takes the complexity out of underlying data structures, and creates a simplified presentation view for authors and business analysts to do predictive analysis and create appropriate reports.

Cognos Framework Manager is a data model development environment from IBM Cognos 10 BI. It is used to create simplified business presentations of metadata derived from one or more data sources like Oracle, SQL Server, DB2, Sybase etc. Using the Framework Manager you can create metadata objects called a Model designed according to business and reporting requirements.

When the reports are generated from the data model, the query that runs against the Data Warehouse is generated using the data model. In this way, the actual metadata information will remain unavailable for the user, as shown in figure 5.3.

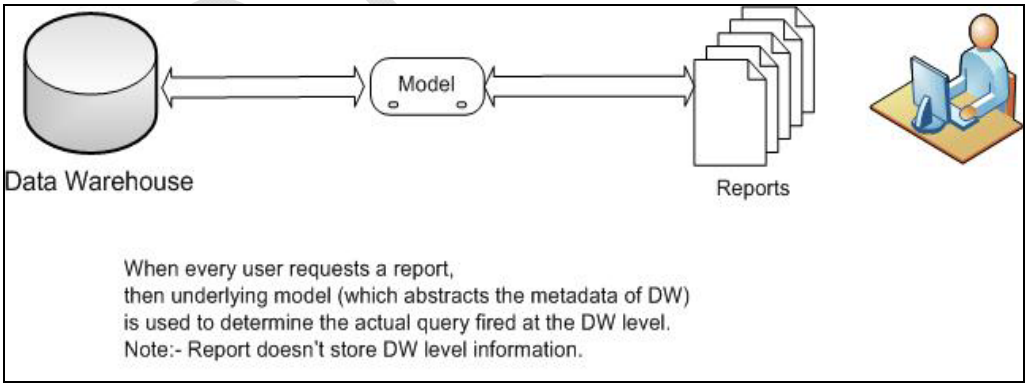


Figure 5.3 Metadata abstraction from reports and users

5.4.1 Different Approaches in Data Modeling

There are two main approaches used in Data Modeling:

Relational Modeling: In this method, normalized data is stored in a relational database. This model is based on the relationships between two-dimensional tables. A relationship captures the way in which these tables are related. A good relational model is the foundation for a dimensional model.

Dimensional Modeling: In the BI world, we frequently have to execute complex queries against the data warehouse, which would create performance issues in case of a running them against a relational database. Dimensional Models are used to overcome these performance issues. This modeling technique is very popular in Data Warehousing mainly for two reasons:

- It is much easier for the business users to understand the data model
- It offers much better query performance compared to the relational models (as discussed in the previous chapters).

A dimensional model generally consists of a huge table containing facts (also known as fact table), while the other tables contain descriptive and/or contextual data for these facts, such as time, geography, product and so on. These descriptive table are known as dimensions.

One of the major advantages of dimensional modeling are the Drill-up and Drill-down features, which allow users to easily summarize data to see the big picture or to go into details, respectively.

5.4.2 Metadata Modeling Using Framework Manager

Using Framework Manager Modeler, you can create two model types:

- **Relational Models** consist of query subjects, which are similar to database tables, and query Items, which are similar to table columns. These are imported from databases and used for reporting.
- **Dimensionally Modeled Relational (DMR) Model** consists of a relational database using a dimensional model. Although this is essentially a relational database, it allows authors to perform OLAP style queries.

Dimensional Modeling Concepts:

The concepts like Dimensions, Measures, and Facts have been covered in previous chapters (Refer to Section 2.2). The other concepts that are necessary to understand are the following:

Regular Dimensions, Levels and Hierarchies: Regular Dimensions represent descriptive data. This data provides the context for the data modeled in Measure Dimensions. A regular dimension can be broken into groups called levels. Again, the levels are organized into hierarchies. For example, a Course dimension can contain levels Course Subject, Course Type and Course in a single hierarchy Course.

Measure Dimension: Measure dimension is the object that contains fact data. Measure Dimension represents the quantitative data that is described by Regular Dimensions. A relationship between two measure dimensions is a regular dimension that is based on query subjects.

Scope Relationships: Scope relationships exist only between a measure dimension and a regular dimension. These relationships are not similar to joins. These relationships actually define the level at which the measures are available for reporting.

Framework Manager enable the users to create a single metadata model that spans all enterprise data sources and applications. It also gives a consistent data view across the enterprise, and a common foundation for information sharing on an enterprise scale. Snapshot of Framework Manager user interface is shown in figure 5.4.

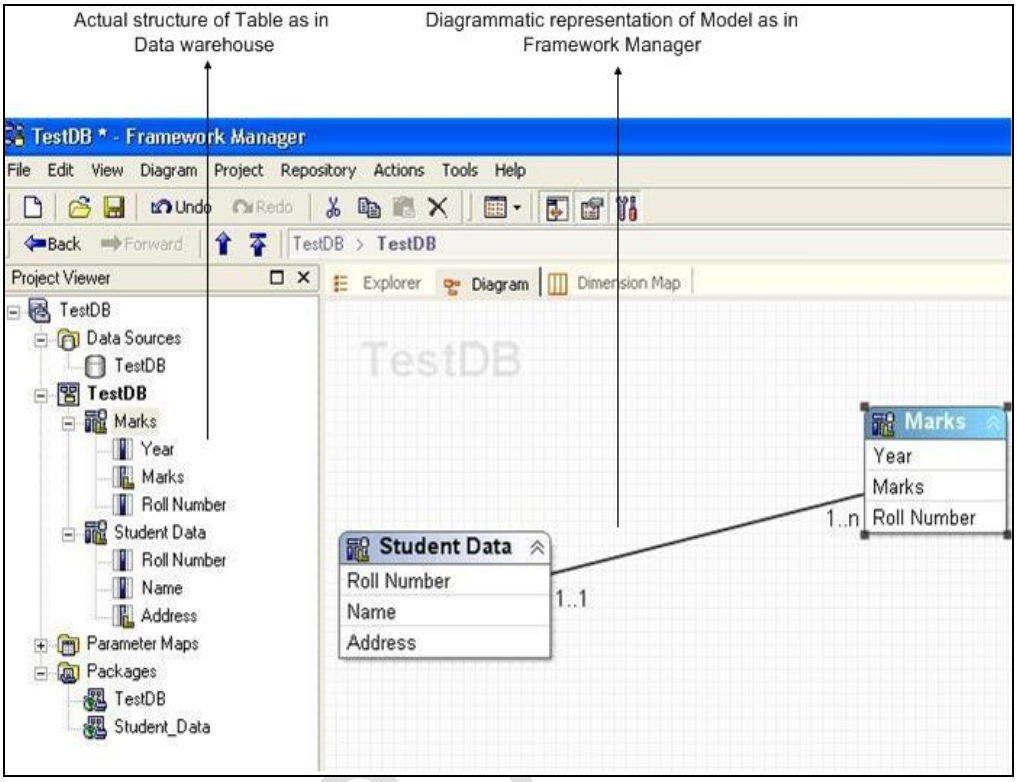


Figure 5.4 Framework Manager Interface

5.4.3 Importing Metadata from Data Warehouse to the Data Modeling Tool

While importing metadata from Data Warehouse the following layers are imported:

Physical Metadata: This layer contains the query subjects that represent the imported tables and the relationships between these query subjects.

Dimensions: This layer contains Regular Dimensions, Measure Dimensions and Scope Relationships.

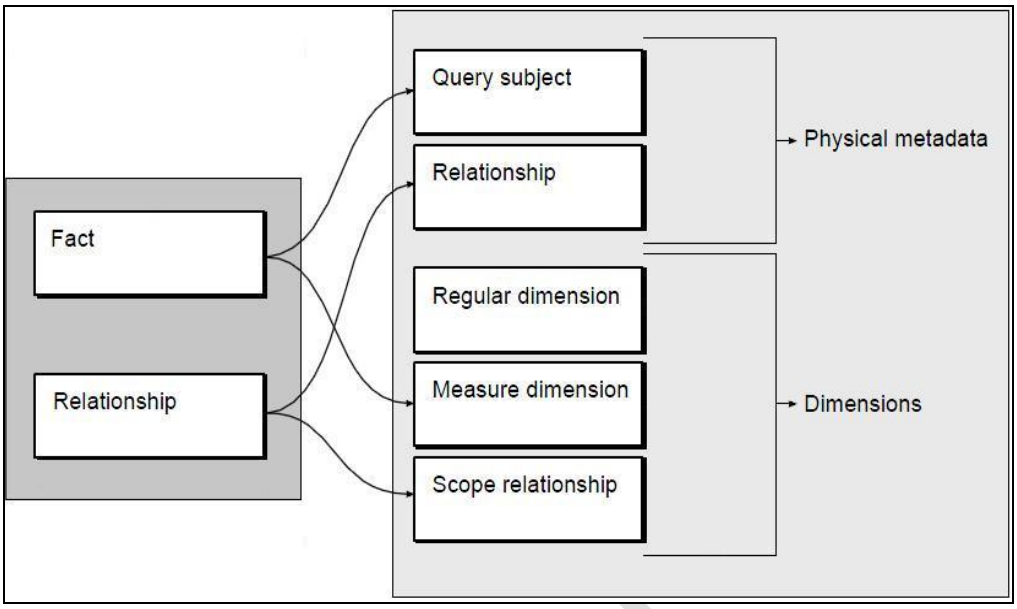


Figure 5.5 Importing Metadata from Data Warehouse

While creating the data model using Cognos Framework Manager, the metadata can be imported from a data warehouse using one or more data sources. These data sources are defined inside Cognos before the metadata is imported.

After the model is designed, a package is created using Cognos Framework Manager based on the model and the package is then published to the Cognos portal so that it becomes available for all the studios.

5.4.4 Cubes

A cube (or an OLAP cube) is a data source containing multidimensional representation of data. It allows faster retrieval and analysis of data. Operations like drill up, drill down etc. are possible using cubes. The cube is also considered as data abstraction that allows a user to view aggregated data from multiple perspectives.

In many cases, cubes contain pre-calculated summarized data, that is: when we load data to the cube, this source data is summarized considering some calculations that were

defined in the cube design phase. So, by calculating aggregations before-hand, we can get faster response times to access the cube data.

Drilling Up and Drilling Down within the cube: Drill Up is an operation of navigating from one level of data to a lesser detailed level of data. In other words, when drilling up a report, the business analyst is summarizing data. Similarly, drill down represents the navigation from one level of data to a more detailed level of data. So, when drilling down, the user is looking for detailed information about a subject. Figure 5.6 depicts drill up and drill down in a graphical form.

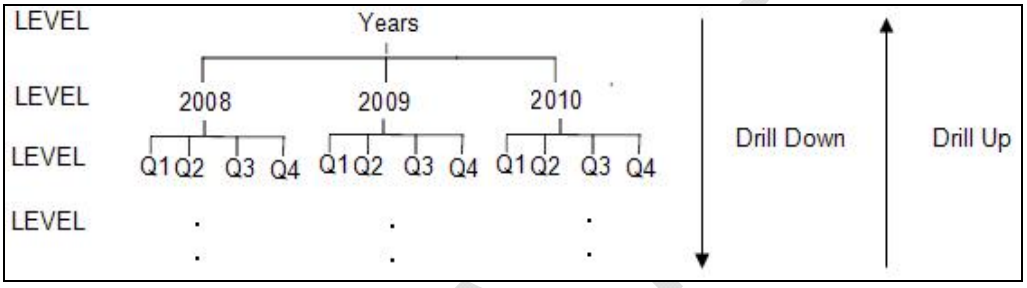


Figure 5.6 Drill Up and Drill Down

Cognos Transformer is used to create cubes (also called PowerCubes). PowerCubes can be created from data warehouse directly and also on top of a Cognos Framework Manager model. After creating the cube, it is published to the Cognos portal so that it becomes available for the different studios.

Drilling Through the cube: Drill Through is a functionality that enables the business user to see detailed information that goes beyond the cube scope, that is, data stored in the source Data Warehouse. The user drills down until the most detailed information available in the cube and if he or she asks for more details, then the application will "drill through" and bring data directly from the Data Warehouse. This process is seamless to the business user, as the software handles all necessary connections and/or selections.

5.5 Query, Reporting and Analysis

After creating the data model from the Data Warehouse, the most important part is delivering the information to the business users. This Information Delivery stage considers three segments:

Query: Most of the information is delivered through queries, which makes Query Management an important element in this process. Ad hoc queries play an important role in the data mining realm, although the use of data mining techniques is still very limited nowadays due to the complexity of the process to collect all necessary data. Users can also save query templates, making it easy to create similar queries.

IBM Cognos Query Studio is a web-based tool that can be used for simple ad hoc queries that the business users might have. Using Query Studio, information can be viewed and exported in different formats like html, excel, pdf, etc. Query Studio is not meant to be used for Professional Report Authoring.

Reporting: Professional reports can be created for the business users based on the data warehouse's data model. Professional reports include graphs, charts etc. Hence, the reports are much more presentable than queries. On the other hand, reports that are available to the business users are generally based on the queries that had been created, so they are not as flexible as queries.

IBM Cognos Report Studio is a web-based tool for professional report authors. Sophisticated business reports can be created using Report Studio from different types of databases. The reports can include multiple pages, multiple queries, charts, graphs, and calculations. These reports support the business needs of the organization, for example monthly sales reports and invoices. Snapshot of Report Studio graphical interface is shown in figure 5.7.

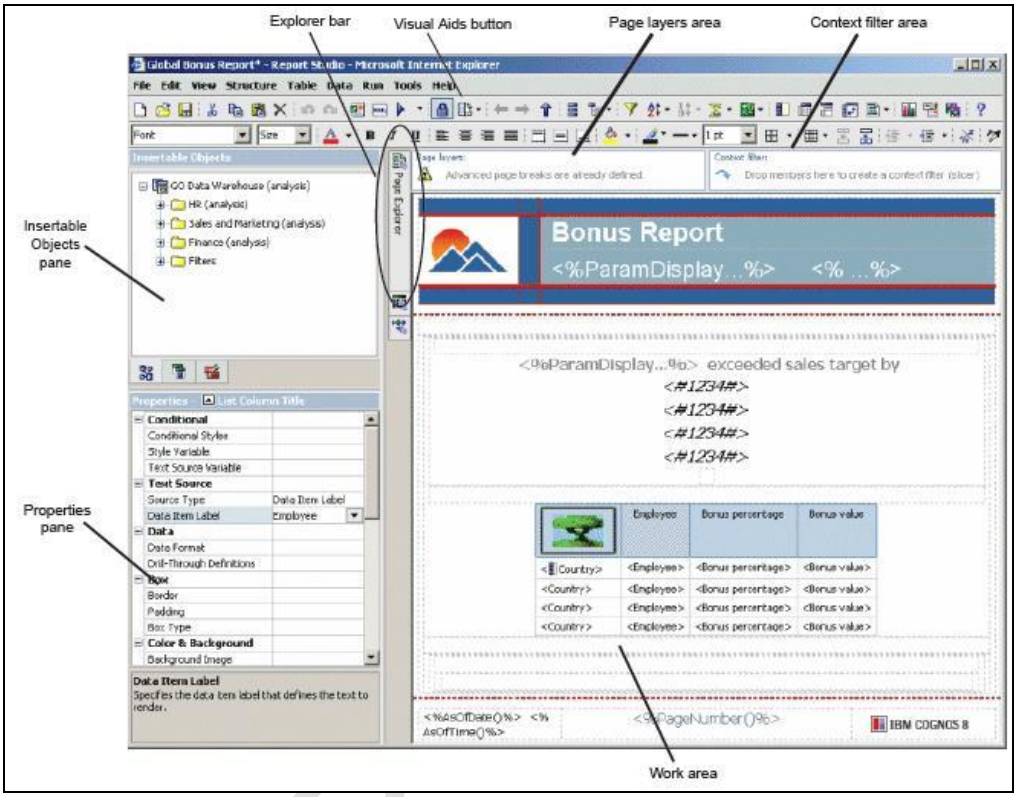


Figure 5.7 Report Studio Interface

Analysis: Analysis can be defined as a set of related queries. Since the Data Warehouse contains all the historical data about business processes, it is suitable to perform analysis. The users can range from market researchers to business strategists. Analysis is performed on multidimensional model using operations like drill up and drill down.

IBM Cognos Analysis Studio is a web-based tool for exploring and analyzing dimensional data that can help the users to answer typical business questions for an enterprise. The interactive drag-and-drop environment in Analysis Studio helps the users to find and focus on the items that are important to their business and compare and summarize as per requirement. Analysis Studio is for the business user who must understand and discover answers to business questions from company data.

5.6 Metrics or Key Performance Indicators (KPIs)

"Key Performance Indicator (KPI) is a measure of performance, which helps define and evaluate an organization success. KPIs help in tracking the progress of organization towards its vision and long-term organizational goals."

KPIs may differ from organization to organization as per their nature of business and strategies. For example:

1. In a business organization, an Increase in ***Average Revenue per Customer year-on- year*** can be a KPI.
2. In an academic institute, ***the failure rate of its students*** can be a KPI.
3. In a manufacturing organization ***Cycle time to manufacture a product (total time from the beginning to the end)*** can be a KPI.

Scorecard

"A scorecard is a collection of metrics that represents the performance of one unit or aspect of an organization."

An organization uses metrics (KPIs) that compares actual results to target, and records that is responsible for the results and the impact of the metric. Once the metrics are defined, a scorecard is required, which is a collection of performance metrics and projects that reflects the strategic goals of a unit in an organization. Scorecards can contain other scorecards to show the organizations in the business.

IBM Cognos Metric Studio is a tool used to achieve the above goals. It helps an organization to create a customized scorecard to monitor and analyze business metrics throughout an organization. Figure 5.8 shows graphical user interface of IBM Cognos Metric Studio.

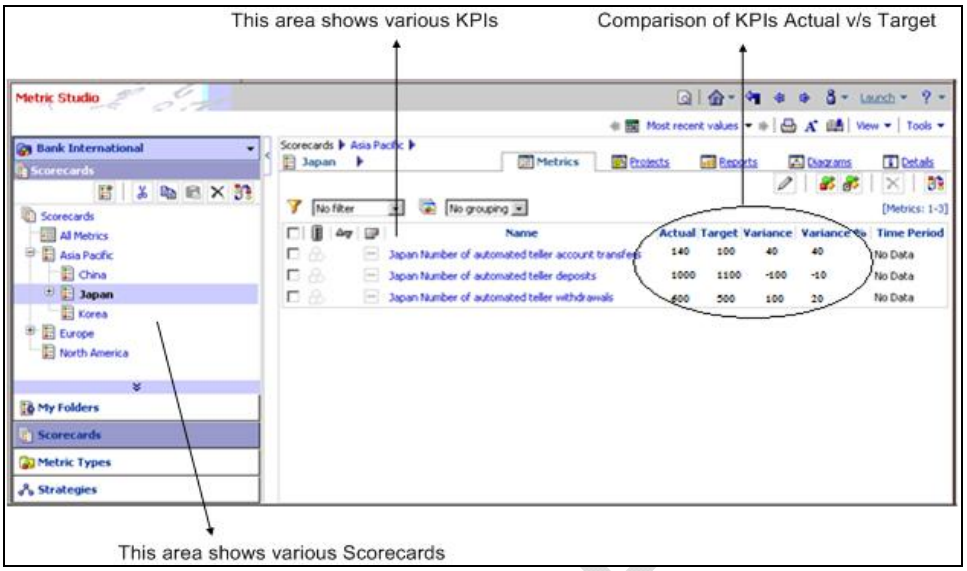


Figure 5.8 IBM Cognos Metric Studio graphical interface

5.7 Events Detection and Notification

"An **event** is a situation that can affect the success of your business. An event is triggered when specific items in your data reach significant values." Notification of events as they happen to the decision makers is very important need in any organization. Timely communicated events lead to effective decision making. In any event notification system, there are two main steps involved:

1. Agent: An agent is a monitor that identifies the event based on any specific condition. The condition can be any change in the organization data that is important to you and you would like to be informed of it.

2. Notification: When an agent detects an event, it can perform tasks to notify that a particular event has occurred, such as sending an email, adding information to the portal, and running reports.

For example in any academic institution, it is very important to notify students and professors regarding the exam schedule. Here exam is an event that needs to be notified

to the students and professors. Another example can be a specific scenario when the attendance of any student is short and this needs to be notified to the professor. In addition, an event can be sent to the student that misses the deadline to return the book to the library via email. Figure 5.9 shows various ways how an event can trigger either an email, report update or notification publication on a portal.

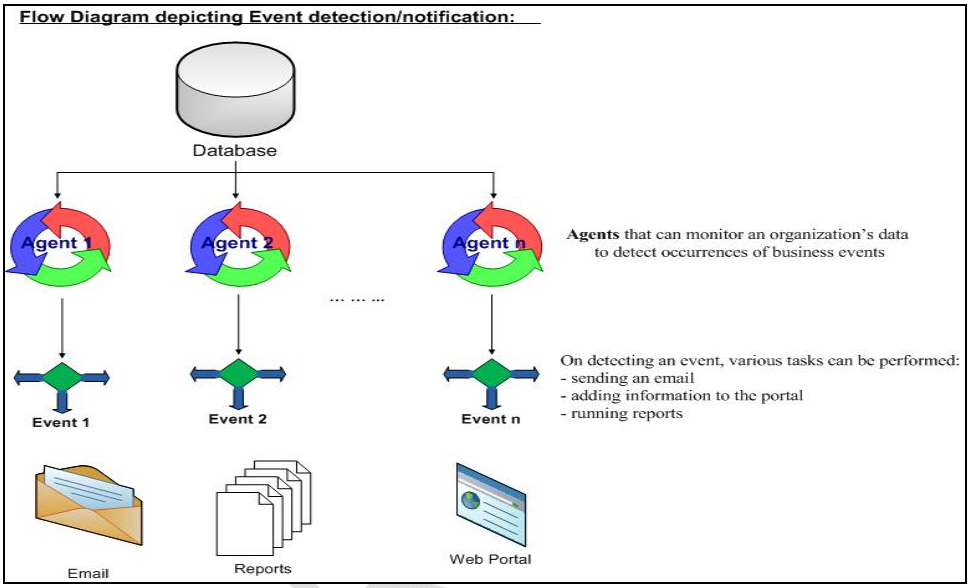


Figure 5.9 Events detection and notification modes

With IBM Cognos Event Studio, users can create **agents** that can monitor an organization's data to detect occurrences of business events. When an agent detects an event, it can perform some specified tasks. For instance, if the quantity of an item is less than the required value in the store, then Event Studio can send an email to the appropriate person. Event Studio is a Web-based tool for creating and managing agents that monitor data and perform tasks when the data meets predefined thresholds. Event Studio can be used to notify decision makers in an organization of events as they happen, so that they can make timely and effective decisions. Figure 5.10 shows snapshot of IBM Cognos Event Studio web interface.

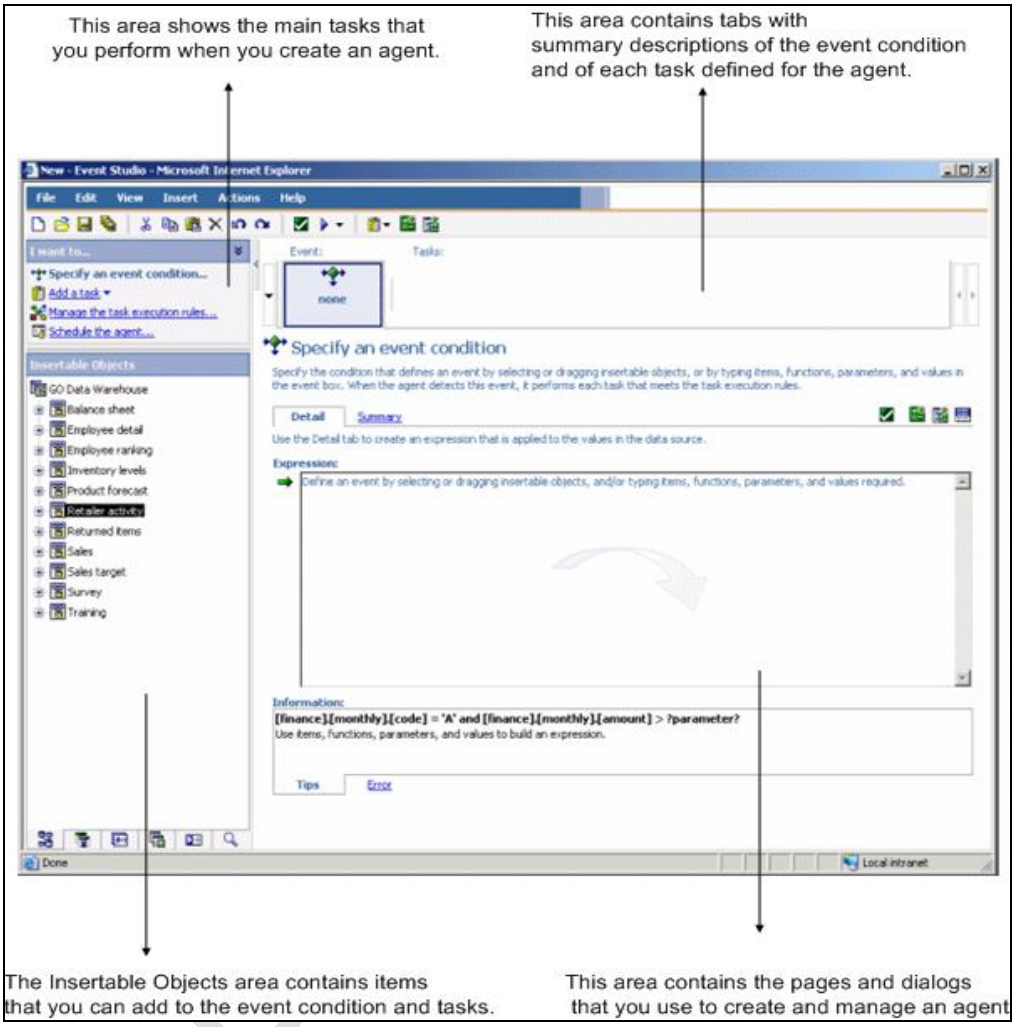


Figure 5.10 IBM Event Studio web interface

5.8 Summary

Business Intelligence Tools help organizations to fetch information and facts from various data sources (OLTP or OLAP) and present them in the form of intuitive reports & charts, which can be shared across the web, email or mobile at any scheduled time. This information helps business managers monitor the organization performance and helps them make appropriate business decisions, which is key to its success and leads to a smarter organization.

5.9 Review Questions

1. Define Business Intelligence and list the capabilities of a Business Intelligence tool?
2. List all the tools that are part of the IBM Cognos Business Intelligence suite.
3. Does IBM Cognos Business Intelligence tool modify the underlying data warehouse while generating reports/charts?
4. Can IBM Cognos Business Intelligence tools generate reports/charts on top of both OLTP (Online Transaction Processing) & OLAP (Online Analytical Processing) data sources?
5. Using IBM Cognos Business Intelligence tools, what all types of reports can you generate?
6. IBM Cognos Business Intelligence tools are used for
    a. Fetching information from a huge data repository called a data warehouse
    b. Preparing complex reports & charts
    c. Scheduling and emailing reports
    d. All the above
7. Which of the following IBM Cognos tools is an Extract Transform Load tool?
    a. Report Studio
    b. Query Studio
    c. Data Manager
    d. None of the above
8. IBM Cognos Framework Manager is used for
    a. Database Metadata modeling to abstract the underlying complexity
    b. Author Reports on top of the metadata models
    c. Both a & b
    d. None of the above
9. Which of the following tools is not part of IBM Cognos Business Intelligence suite?
    a. Framework Studio
    b. Report Studio
    c. Query Studio
    d. Metrics Studio

10. Which IBM Cognos Business Intelligence tool is used for sending notification & alerts?

    a. Report Studio
    b. Analysis Studio
    c. Event Studio
    d. None of the above

5.10 Exercises

11. Open Report Studio and create any report consisting of a bar graph and a cross table. Data in the bar graph should be in harmony with the cross tab.
12. Open Event Studio and create a few events, for example whenever “Avg attendance” of any student goes below “50%”.
13. Open Analysis Studio and do some analysis on any available data. Also, apply runtime analysis functions like “Percentage of total”, “Avg of total”, etc.
14. Create a report using Report Studio which has “Drill-up” and Drill-down” functionality.
15. Create a report using Report Studio which has “Drill-through” functionality.

6

Chapter 6 – A Day in the Life of Information (an End to End Case Study)

The primary purpose of this chapter is to put in perspective all the concepts described so far. From **Chapter 1** to **5**, we discussed what a data warehouse is, its logical and physical designs, the hardware selection, and the OLAP structures. We will try to put together all these concepts into a real-life example, thereby applying to a practical example all the knowledge gained so far.

6.1 The Case Study

GettingStarted Institute of Technology is an engineering college, which currently offers engineering in two streams, computer science and civil engineering. The college has 40 students at each branch per semester. Each stream considers 4 years for conclusion. Currently the college has an automated system to record the attendance of students (only students) and in fact, that is the only automated system the institute has. All other information lies on paper files. All students have to use their RFID based ID cards while entering the lecture sessions, labs, department or library to record the attendance. Once the attendance records are sent to university, they are purged from the system. This typically happens on the first day of each month. In few exceptional cases, this activity completes on the fifth day of the month. Therefore, this system could hold attendance data for up to a period of one month plus five days. So this system has been designed to handle and store data for up to thirty six (thirty one plus five) days only. Nowadays the institute has four lecture rooms, one lab and one library. The college has mandatory hotel accommodation on campus. And all facilities open seven days a week throughout the year.

Classes or lectures, seminars, tests, sports and other miscellaneous activities can be scheduled anytime between 8 AM and 5 PM from Monday to Sunday with lunch break from noon to 1pm. But all labs, departments and the library is open for 24 hrs throughout the year.

The college dean was asked by the management to put a system in place that shows some performance metrics indicating how well the college is doing. This system should also show resource utilization in the college. The resources include lecture rooms, labs, library and teaching staff. He was explicitly asked not to use examination results or campus placements, salary or fee information in any of these metrics. He can use existing students' attendance system but is not allowed to change anything in it.

6.2 Study Existing Information

As we know, only the attendance system has data in a form that can be leveraged effectively. We will try to keep our solution design focusing on that. Now we will add more items only if they are essential to meet the requirements.

6.2.1 Attendance System Details

The attendance system has tables that are updated in real-time whenever a student enters any lecture room, lab or library.

Note: To keep things simple the exit time of a student is not recorded. When a student enters, we assume that he/she attends the complete lecture/lab. In later part of our solution, we will find how we will utilize attendance system data for library performance.

Table Name and structure	Description
student_master *student_name* CHAR(50) *enrollment_no* INTEGER *stream* CHAR(10) *semester* SMALLINT	Table containing student master data, having name, enrollment number, stream and semester of all students
facilitator_master *facilitator_id* SMALLINT *facilitator_name* CHAR(50) *subject_id* INTEGER *role* CHAR(10)	Table containing facilitator master data, having facilitator id, name and his/her role and the subject presented by this facilitator (assuming it is only one subject per facilitator).
resources	Table containing all resources available in

resource_id SMALLINT *resource_name* CHAR(20) *resource_type* CHAR(20) *capacity* SMALLINT	the institute. This stores resource id, resource name, its type and capacity
attendance_record *enrollment_no* INTEGER *resource_id* SMALLINT *enroll_timestamp* TIMESTAMP	Table containing real-time attendance data, capturing enrollment number, resource id and time of recording the attendance.
time_table *start_time* TIMESTAMP *end_time* TIMESTAMP *resource_id* INTEGER *facilitator_id* INTEGER	Table containing schedule of lectures, having lecture start time, end time, place of lecture (resource), professor id (facilitator id).

Table 6.1 contains list of tables used on the Attendance system.

By looking at the above table structure and the amount of data captured, you may argue that this system should have more details. However we are not going to have complete data here, so that we can show you how we handle the situation in case of insufficient data at the source system when designing a data warehouse. Figure 6.1 shows above mentioned tables and their relationships.

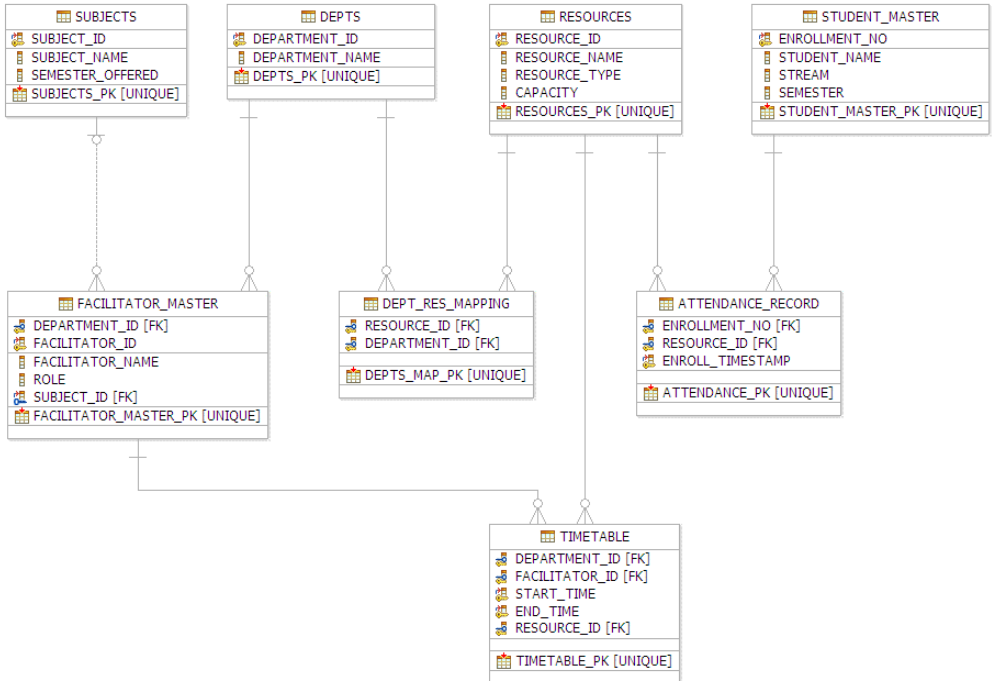


Figure 6.1 Attendance system database model

As one can see in Figure 6.1, table *attendance_record* references tables *student_master* and *resources*. However it has indirect relationships with all other tables, as it happens with table *facilitator_master*, which is related via table *timetable*. To populate data in attendance system tables, please refer to **Appendix A**. About information on which software to use and how to install it, please refer to **Appendix B**.

6.2.2 Study Attendance System Data

Let us assume that the data transfer to university is done via FTP. Attendance records are exported from the database in a comma-delimited plain text file. This exported file is then transferred to university repository via FTP by the institute's administration staff. As we are allowed to use this data for our data warehouse consumption, we can also get the attendance dump (comma delimited plain text file) by sharing our file location with the administration team. For now, we will import the complete data as is. Once everything is done, we can extract the necessary data from the attendance dump.

6.3 High Level Solution Overview

Let us try to figure out how we can use available data to fulfill requirements of a proposed data warehouse. From this case study, we can conclude that the requirements can be divided into two categories: performance metrics and resource utilization.

To break this further down, we need to decide what kind of performance parameters we can derive from the attendance system. One of them can be the percentage of classes attended by students during the semester. Assuming that attending a class adds value to students, if students are not attending some particular professor or subject, then it can be used as an indicator of possible improvement (e.g., either by improving the professor's teaching style or by updating content of course). There can be many other options; however, to keep the design as simple as possible, we will focus our implementation on just one performance metric for now.

For resource utilization, we can investigate the utilization of lecture rooms (number of hours the lecture room was in use versus total availability of eight hours a day), library utilization (percentage of students using it, broken down by stream, semester, subjects), lab utilization, overall resources utilization during weekends and off-working hours (time period outside 8 AM – 5 PM). This information can be used to decide whether labs or the library

should be kept open on weekends or off-working hours. Utilization of professors (number of hours taught versus total hours available). Here as well we can explore many other scenarios of resource utilization, however we will focus our implementation on only three scenarios, that is, lecture room, library and labs utilization.

To summarize, here it is a list of what we will implement in the solution for our case study.

1. Performance parameters
    A. Statistics of students attending lectures
2. Resource utilization
    B. Lecture room utilization
    C. Library utilization
    D. Labs utilization

We will now discuss the dimensions for performance metrics listed above, which will be call **measures**.

From a business standpoint, this implementation is oversimplified. We could easily think of several other metrics that could be more useful and representative than the ones listed here. But please remember the data model in this study is designed to keep the solution as simple as possible. We expect to make the whole process more understandable by beginning with a few simple steps.

6.4 Detailed Solution

In this section, we will discuss the details of how to implement the proposed solution.

6.4.1 A Deeper Look in to the Metric Implementation

Now that we have a list of measures to implement, let us examine them one by one and decide on what dimensions these measures will have to allow useful interpretation.

Statistics of students attending lectures

The recorded attendance data has the information of which lecture room a student has entered, therefore if we can link this information with the timetable of the institute and find out the subject of the lecture a student is attending. From the student enrolment number,

we can find out which semester the student is enrolled in. We can also find out the student's engineering stream. Therefore, we can have following dimensions for this measure:

1. By Engineering Stream / Year / Semester / Subject
2. By Facilitators

Lecture room utilization

This is something simple to calculate. It is the total number of hours a resource was in use versus the total duration of resource availability. This ratio is expressed as a percentage.

The resource availability is eight hours per day for lecture rooms (as the timings are 8 AM to 5 PM minus the lunch break) and twenty-four hours for labs and the library (as they are open all day long). In this category, we can have one more measure as "**Capacity utilization**", which indicates the percentage of capacity used when the resource was in use. For example, if a lab was in use for twenty four hours a day then the room utilization is 100%, however if only twenty students on average attended a lab which can accommodate forty students, then the capacity utilization is 50%. For now, we will fix our implementation on the first case of "**Room utilization**" only. You can implement "**Capacity utilization**" as an add-on exercise to this chapter.

Now we can present this data over the same dimensions as discussed above, however that is not going to make the results more useful. Therefore, we will limit analysis of this measure only with time dimension.

The way to calculate library utilization is similar to the lecture room utilization except that the availability window of the library is twenty-four hours as compared to eight hours of lecture rooms. This dimension is also analyzed over time dimension only.

Please note that, similar to what we did with the "**Lecture Room capacity utilization**" mentioned above, you have an additional assignment waiting for you as to implement the "**Library capacity utilization**".

Labs utilization

The lab utilization will be exactly like the library utilization and we will use time dimension to analyze this measure.

Again, you are invited to implement by yourself the "**Lab capacity utilization**".

6.4.2 Define the Star Schema of Data Warehouse

We already have four measures and four dimensions for our case study ready for implementation. Here is the summary:

Measure

1. Statistics of students attending lectures
2. Lecture room utilization
3. Library utilization
4. Labs utilization

Dimensions

1. Time
2. Engineering stream / Semester / Subject
3. Resources
4. Facilitators

In our implementation we will use one fact table only with perfect star schema (as opposed to a snowflake schema, please refer to **Appendix B** for addition information on star and snowflake schemas). The fact table should have the following table structure in order to fulfill our requirements.

Table structure	Description
attendance_fact *enrollment_id* INTEGER *resource_id* SMALLINT *time_id* INTEGER *subject_id* SMALLINT *facilitator_id* INTEGER	This table shows each individual attendance as a single "fact": it informs the date (*time_id*) when each student (*enrollment_id*) attended to which resource (*resource_id*), the subject and the facilitator to that class (*subject_id* and *facilitator_id* respectively)

Notice that this fact table has no cumulative measure, that is: no column with a measure that is summed up, like dollars or quantities. This is due to the nature of the current Data Warehouse we are creating, as the business metrics in our case study are all calculated by counting the number of facts in the fact table.

We will now discuss the table structure of our dimension tables.

To implant these objects, we used IBM DB2 Express-C and it's integrated development environment, IBM Data Studio. For detailed information on how to install the software and/or references on how to use them, please refer to **Appendix B**.

Table structure	Description
resource_dim *resource_id* INTEGER *resource_name* CHAR(50) *category_id* INTEGER , *category_name* CHAR(50) *type_id* INTEGER , *type_name* CHAR(50)	Table with resource dimension's levels and hierarchy information. Category => Type => Resource
time_dim *time_id* INTEGER *time_name* CHAR(50) *year_id* INTEGER *year_name* CHAR(50) *sem_id* INTEGER *sem_name* CHAR(50) *month_id* INTEGER *month_name* CHAR(50) *day_id* INTEGER *day_name* CHAR(50) *start_time TIMESTAMP* *end_time TIMESTAMP*	Table with time dimension's levels and hierarchy information. Year => Semester => Month => Day => Time
subject_dim *subject_id* INTEGER *subject_name* CHAR(50) *semester_id* INTEGER *semester_name* CHAR(50) *year_id* INTEGER *year_name* CHAR(50)	Table with subject dimension's levels and hierarchy information. Year => Semester => Subject

facilitator_dim *facilitator_id* INTEGER *facilitator_name* CHAR(50) *department_id* INTEGER *department_name* CHAR(50)	Table with facilitator dimension's levels and hierarchy information Department => Facilitator

How to populate these tables will be discussed further in ETL Section of this chapter.

Notice that **Engineering Stream** is left aside on this database model. We do so to emphasize an important consideration when designing a star schema. There are several ways to include this information in our model, but each solution assumes a special consideration. This is quite common in the design phase of any data warehouse.

For instance, we can include the Engineering Stream as the top level of the hierarchy described in table *subject_dim*. But if we do so, we are implicitly considering that each *subject_id* is part of one stream only (please refer to the discussion about hierarchies presented in Section 2.3.1). This might not be true, as there are subjects that are common to any Engineering stream, as 'Calculus', for instance. We would need to change our data model to have different *subject_id's* for the subject Calculus for the different streams. This would not be a great solution, as in that case the column *subject_name* would become a rollup for column *subject_id.* And of course, this would not be so obvious to the business user, which might generate errors when creating reports.

An alternative approach is to create a new dimension *student_dim*, as shown in table below.

student_dim *enrollment_id* INTEGER *student_name* CHAR(50) *stream_id* INTEGER *stream_name* CHAR(50)	Hypothetical table with student information. Stream => Student

But again, we have a drawback. Remember that when creating a data model you need to think of any possible scenario that your system might be exposed to. And in this solution, a student would not be allowed to attend to different streams with the same *enrollment_id*, otherwise we would not have the desired parent-child relationship for this hierarchy. Again we may consider that the same student uses different *enrollment_id's* for different streams, but we fall into the same risk of causing errors when creating reports.

A third solution is to add a new column to the fact table, called *stream_id*. This is clearly the worst option, as we have two major problems:

1. We are adding a column to a huge table that may represent 95% of the database size. Adding a column to that table is something to avoid at all costs.
2. Besides, you still need to create a new dimension, *stream_dim*, which would inform the stream name.

Finally, another solution is to introduce a new dimension to our star schema, called *stream_dim*. This dimension cave have columns such as stream, subject and enrollment, with a composite primary key on *enrollment_id* + *subject_id* . This new dimension needs a relationship with the fact table based on those two key columns (*enrollment_id* and *subject_id*). This solution is not ideal, but yet is a better option than the previous ones. You can implement this deviation later on as add-on practice assignment to the current chapter.

6.4.3 Data Size Estimation

Estimating data size is necessary in order to acquire suitable hardware and storage. Let us revisit our case study once more regarding the size of source system. System keeps up to thirty six (current week + last 4 weeks) days data for eighty students (forty students each in two streams offered in each semester) during 4 years. Assuming that one lecture lasts an hour, once student can have possible eight entries per day for twenty two working days (no scheduled classes on weekends). In addition, they also use library and labs in off-working hours and number of max possible entries depends on the capacity of the library and labs. For the purpose of simplicity, we will assume library and lab usage is half as much as it is for lectures during weekdays.

Source system data size calculations

```
Total students in year 1,2,3 & 4     = 320    [ 4x(40+40)            ]
All these students must be attending to lectures/sessions during the day,
therefore
Assuming 8 lectures/session per day:
  max attendance entries during day  = 2560   [ 8x320               ]
Assuming 22 working days per month:
  max attendance entries per month   = 56320  [ 22x2560              ]
Assuming half the students use library and labs:
  Total entries for lab & library    = 28160  [ 56320 / 2            ]
Grand total of entries               = 84480  [ 56320 + 28160        ]
```

```
Row size of attendance table         = Int(4)+Int(4)+Timestamp(12) Bytes
                                     = 20 Bytes
Attendance table size for a month    = 84480 x 20 Bytes
                                     = 1689600 Bytes
                                     = 1.6 MB (approx)

Addition entries for 5 days          = 12800  [ 5x2560                ]
Total entries for lab,lib attendance = 6400   [ 12800 / 2             ]
  Additional data in table           = 1228800[ (12800+6400)*20 Bytes ]
                                     = 0.4 MB (approx)
Additional data size due to other tables
                                     = 1.0 MB (approx)
Total system size for one attendance cycle
                                     = 1.6 + 0.4 + 1.0 MB
                                     = 3 MB
Total system size with indexes       = 6 MB ( approximately double the
                                              source data size)
```

We need to size two systems here, one is the ETL (extract, transformation and load) system that stores the data that will be processed and modified per the requirements for the warehouse, and the other is the warehouse system that will store the processed data for reporting.

ETL Data size calculations	
Source system data size	= 3.0 MB (only data size)
As general practice, we should have approximately three times storage spaces as that of source system.	
ETL storage size	= 9.0 MB

Once the data is processed, it is ready to be loaded into the data warehouse. Prior to that, we will bring in another term here called “data lifetime”. That is, once the data is loaded into the data warehouse, how long will it remain in the system before becoming stale or of minimal use. Considering the duration of the engineering course is four years, data for any student will remain relevant for at least four years. In addition, the institute might want to keep the data for another few more years so that they can investigate historical data for a longer period. We now assume this additional period to be four more years as well. One more thing that we need to bring to attention and discuss is the level of data aggregation. As you might have noticed, the time dimension discussed above uses month as the lowest

granular level, which indicates that we are aggregating data at the month level. The lower the aggregation level is, the more the data will be handled.

Data warehouse size calculations

6.4.4 The Final Schema

We will finalize our star schema for the proposed warehouse at this step. Figure 6.2 shows how our star schema looks like.

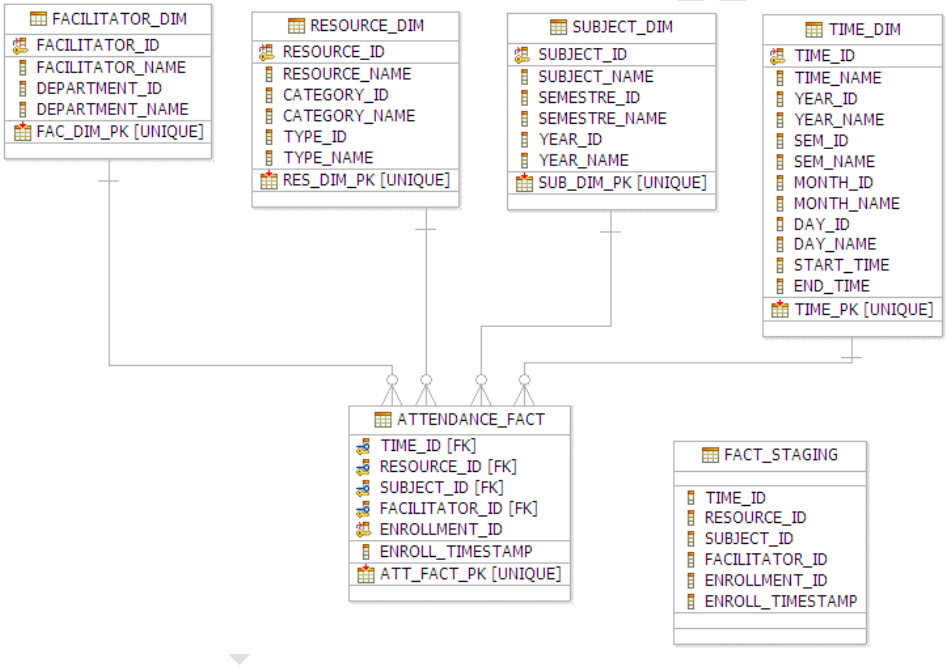


Figure 6.2 Data warehouse star schema

6.5 Extract, Transform and Load (ETL)

So far we have studied our transactional system ((attendance) and data model of analytics system (data warehouse). The process of linking the data flow between transactional to analytics system is called ETL. This name comes from the abbreviation of the steps

involved in this process, as mentioned in **Chapter 4**. We extract the data from a source or transactional system, do some transformations so that data aligns with the target data warehouse data model, and finally load it into a warehouse.

Let us divide our ETL process into five activities in parallel, called **jobs** in ETL terminology. We will have one job for each dimension table and one job for the fact table. We will fix our ETL implementation with simple SQL/Shell scripts, as the transformations will be simple in nature. However, in production environments the use of commercial tools such as IBM Information Server (Datastage, QualityStage) is recommended.

Before we start the ETL process, we have to revise one important concept with dimension tables that is "Dimension Hierarchy", as introduced in Section 2.3.1. As the name suggests, it is the logical placement of dimensions at various granular levels. For example, if we use the "Area Dimension" table to define the hierarchy, we should define levels like "Pin code", "City", "District", "State", "Country" and then arrange these levels in right order of granularity. In our example, the correct order will be Country, State, City, District, Pin code, where the "Pin code" level is at the lowest granularity level.

Typically the levels discussed above are implemented as IDs in most systems (e.g., `area_id`, `city_id`, `state_id`) and these IDs are used to join tables. However, we cannot use IDs in the reports, as they are not intuitive to understand. Therefore, it is a good practice to add one more column for every ID column as a label for each ID column, and then use these labels in the reports. Table 6.1 lists the same table for "Area Dimension" with the "String Labels" column.

pin	pin *name*	city id	city_*name*	dist id	dist *name*	state id	State *name*	country id	Country name
56007	ECity	11201	Bangalore	11223	South	43	Karnataka	744	India
95134	Tasman Dr	66763	San Jose	23455	East	66	California	899	USA
2232	Pavas	8338	San Jose	87654	North	99	Costa Rica	454	Mexico

Table 6.1 Sample Dimension table for Area.

Table 6.1 represents sample area dimension table with data. Note that IDs are for joining this table with the fact table or any other table, and label columns are only for the reporting display purpose. Figure 6.3 shows the levels and hierarchy of the area dimension.

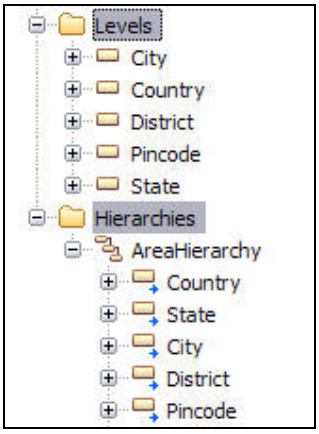


Figure 6.3 Levels and hierarchy view of Area dimension

The Area dimension has one hierarchy as shown in Figure 6.3, but has several levels. Any dimension has to have at least one level and one hierarchy.

6.5.1 Resource Dimension

All resources in the institute fall under three basic categories:

1. Lecture Rooms
2. Labs
3. Library

For resource dimension, we will have to define an intermediate table that will do resource to department mapping. Let us call this table as *dept_res_mapping* as defined below. (Please refer to **Appendix A** for the complete definition). This table has resource_id and department_id columns.

```
CREATE TABLE staging.dept_res_mapping  (
               resource_id INTEGER NOT NULL ,
               department_id INTEGER NOT NULL ) ;
```

A lecture room can be either a general purpose or a department one. Labs are always in a department. A library can also be either a departmental library or Central Library. **Table 6.2** shows the table structure for the resource dimension.

resource_id	resource_*name*	category_*id*	category_*name*	type_*id*	type_*name*

1000	LR001	100	Departmental	10	Lecture Room
1001	LR002	100	Departmental	11	Lecture Room
1002	LR003	100	Departmental	12	Lecture Room
:	:	:	:	:	:
1010	LIB02	150	Central Library	15	Library
1011	LAB05	200	CS Lab	16	Lab

Table 6.2 Resource Dimension table structure

Table 6.2 represents table structure for resource dimension with sample data. The hierarchy used here is Resource type, Category, Resource, where "Resource" is the lowest level. This table can be created using the SQL statement below. (Please refer to **Appendix A** for the complete definition).

```
CREATE TABLE dw.resource_dim  (
               resource_id INTEGER NOT NULL ,
               resource_name CHAR(50) NOT NULL ,
               category_id INTEGER ,
               category_name CHAR(50) ,
               type_id INTEGER ,
               type_name CHAR(50) ) ;
```

To populate the above table, we will have to execute the following SQL statement:

```
INSERT INTO dw.resource_dim
(resource_id, resource_name,
      category_id, category_name, type_id, type_name)
SELECT R.resource_id, r.resource_name,
d.department_id, d.department_name,
      CASE r.resource_type
            WHEN 'Lecture Room' THEN 1
            WHEN 'Library' THEN 2
            WHEN 'Lab' THEN 3
            END AS type_name, r.resource_type
FROM tx.resources R
INNER JOIN staging.dept_res_mapping M
      ON R.resource_id = m.resource_id
INNER JOIN staging.depts D
```

```
      ON D.department_id = m.department_id  ;
-- notice the above query could be written without using JOINs
-- but JOINs are part of SQL standard
-- and therefore it is a good idea to always use them
```

We have used a merge statement because it can update and insert data in a single SQL statement. After the execution of the above statement, we will have all resources, resource names and types in the dimension table. Now we also need to update the dimension table with department IDs. Since the resource to department mapping does not exist in a transactional system, we will have to create an intermediate table that will be used by the ETL job to populate the department information and resource to department mapping. The SQL statement below will update the department id and names in our dimension table:

```
MERGE INTO dw.resource_dim dest
  USING      (SELECT      department_id,      department_name      FROM
staging.dept_res_mapping) src
  ON (src.resource_id = des.resource_id)
  WHEN MATCHED THEN
     UPDATE SET
        dest.department_id = src.department_id,
        dest.department_name = src.resource_name ;
```


6.5.2 Time Dimension

Time dimension is typically the only dimension that exists in all data warehouses, however the number of levels and the granularity varies from each data warehouse implementation as it is very specific to the domain.

In our case study, we divide a day into the following six periods:

1. Before College (Time before college classes starts, i.e. before 8AM)
2. Session 1 (8AM – 10AM)
3. Session 2 (10AM – 12Noon)
4. Session 3 (1PM – 3PM)
5. Session 4 (3PM – 5PM)

6. After college (Time after college classes end, i.e. after 5PM)

Note: Lunch hour (12Noon to 1PM) is left out intentionally as an assignment to implement later.

This table can be created using the following SQL statement. (Please refer to **Appendix A** for the complete definition).

```
CREATE TABLE dw.time_dim (
       time_id INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
                  START WITH +1
                  INCREMENT BY +1
                  MINVALUE +1
                  MAXVALUE +32672
                  NO CYCLE
                  NO CACHE
                  NO ORDER ) ,
       time_name CHAR(50) NOT NULL ,
       year_id INTEGER NOT NULL ,
       year_name CHAR(50) NOT NULL ,
       sem_id INTEGER NOT NULL ,
       sem_name CHAR(50) NOT NULL ,
       month_id INTEGER NOT NULL ,
       month_name CHAR(50) NOT NULL ,
       day_id INTEGER NOT NULL ,
       day_name CHAR(50) NOT NULL ,
       start_time TIMESTAMP NOT NULL ,
       end_time TIMESTAMP NOT NULL )  ;
```

Table 6.3 shows the structure of time dimension that we use in our case study.

time id	time_*name*	year id	year_*name*	sem id	sem_*name*	month id	month_*name*	day id	day_*name*
1	Before College	2009	Year09	1	Fall	1	May	1	One
1	Session 1	2009	Year09	1	Fall	1	May	1	One
1	Session 2	2009	Year09	1	Fall	1	May	1	One
1	Session 3	2009	Year09	1	Fall	1	May	1	One
1	Session 4	2009	Year09	1	Fall	1	May	1	One

6	After College	2009	Year09	1	Fall	1	May	1	One

Table 6.3 Time Dimension table structure

Please note that in the time dimension table, we will have six rows for one day. This means 6x365 rows for one year. We will use a recursive SQL command to populate this time dimension table. We will also be adding two more columns to the table above, *start_time* as timestamp and *end_time* as timestamps.

The population of the time dimension table will be performed using a recursive SQL script. This script will populate the time dimension for 8 years of time from current year to (year – 7).

In our case study, we will limit the time dimension table to one month (August/2009) only. For one month, time*_dim* will have 31 days entries with six rows per day. That will be 426 rows. Time dimension generation can be very handy using any professional ETL tool (like IBM Datastage), for our case study, we will simply import the data from a comma, separated flat file generated manually.

1,Before College,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-00.00.00.000000,2009-08-01-07.59.59.999999

2,Session 1,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-08.00.00.000000,2009-08-01-09.59.59.999999

3,Session 2,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-10.00.00.000000,2009-08-01-11.59.59.999999

4,Session 3,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-13.00.00.000000,2009-08-01-14.59.59.999999

5,Session 4,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-15.00.00.000000,2009-08-01-16.59.59.999999

6,After College,2009,Year 09,1, Fall,8,August,100,Hundred,2009-08-01-17.00.00.000000,2009-08-01-23.59.59.999999

7,Before College,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-00.00.00.000000,2009-08-02-07.59.59.999999

8,Session 1,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-80.00.00.000000,2009-08-02-09.59.59.999999

9,Session 2,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-10.00.00.000000,2009-08-02-11.59.59.999999

10,Session 3,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-13.00.00.000000,2009-08-02-14.59.59.999999

11,Session 4,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-15.00.00.000000,2009-08-02-16.59.59.999999

12,After College,2009,Year 09,1, Fall,8,August,101,HundredOne,2009-08-02-17.00.00.000000,2009-08-02-23.59.59.999999

:

421,Before College,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-00.00.00.000000,2009-08-31-07.59.59.999999

422,Session 1,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-08.00.00.000000,2009-08-31-09.59.59.999999

423,Session 2,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-10.00.00.000000,2009-08-31-11.59.59.999999

424,Session 3,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-13.00.00.000000,2009-08-31-14.59.59.999999

425,Session 4,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-15.00.00.000000,2009-08-31-16.59.59.999999

426,After College,2009,Year 09,1,Fall,8,August,131,HundredThirtyOne,2009-08-31-17.00.00.000000,2009-08-31-23.59.59.999999

6.5.3 Subject Dimension

Subject dimension will give us information view in terms of subject, semester, and year. All subjects will be tied to a given semester, and semester will fall under particular year and that is the subject dimension hierarchy.

Note: There are few subjects that are electives and are not specific to any semester, in our case study, we will not consider those subjects. It can be implemented later as an assignment.

Table 6.4 shows subject dimension with sample data.

subject id	subject name	semester_*id*	semester_*name*	year id	year_*name*
200021	Mechanics-I	10	Winter	100	First Year
200040	Mechanics-II	11	Fall	101	First Year
400040	Data Structures	12	Winter	201	Second Year
400041	Automata Theory	12	Winter	201	Second Year

Table 6.4 Subject dimension table structure with sample data

The Subject dimension table can be created using following SQL statement. (Please refer to **Appendix A** for the complete definition).

```
CREATE TABLE dw.subject_dim  (
               subject_id INTEGER NOT NULL ,
               subject_name CHAR(50) NOT NULL ,
               semester_id INTEGER NOT NULL ,
               semester_name CHAR(50) NOT NULL ,
               year_id INTEGER NOT NULL ,
               year_name CHAR(50) NOT NULL )  ;
```

Please note that in our implementation if a subject is being offered in two different semesters or years, then it should have two entries in above table with different subject IDs. The Subject dimension table can be populated via the following SQL statement:

```
INSERT INTO dw.subject_DIM
SELECT s.subject_name, s.subject_name, m.semester,
       CASE m.semester
              WHEN 1 THEN 'First Sem'
              WHEN 2 THEN 'Second Sem'
              WHEN 3 THEN 'Third Sem'
              WHEN 4 THEN 'Forth Sem'
              WHEN 5 THEN 'Fifth Sem'
              WHEN 6 THEN 'Sixth Sem'
              WHEN 7 THEN 'Seventh Sem'
              WHEN 8 THEN 'Final Sem'
       END AS semester_name,
       CASE m.semester
              WHEN 1 THEN 1
              WHEN 2 THEN 1
              WHEN 3 THEN 2
              WHEN 4 THEN 2
              WHEN 5 THEN 3
              WHEN 6 THEN 3
              WHEN 7 THEN 4
              WHEN 8 THEN 4
       END AS year_id,
       CASE m.semester
              WHEN 1 THEN 'Year 1'
              WHEN 2 THEN 'Year 1'
              WHEN 3 THEN 'Year 2'
              WHEN 4 THEN 'Year 2'
              WHEN 5 THEN 'Year 3'
              WHEN 6 THEN 'Year 3'
              WHEN 7 THEN 'Year 4'
              WHEN 8 THEN 'Year 4'
       END AS year_name
FROM   staging.subjects s
       INNER JOIN tx.student_master m
              ON s.semester_offered = m.semester
GROUP BY s.subject_id, s.subject_name, m.semester  ;
```


6.5.4 Facilitator Dimension

Facilitator dimension can also take up a multilevel hierarchy (Department, Subject area, Dean/Prof/Asst Prof etc), however we will limit our scope to only Department level. Please note that rest is left for you to implement as an assignment.

Table 6.5 shows the Facilitator dimension table structure with sample data.

facilitator_*id*	facilitator_name	*department_id*	department_*name*
1001	Mark Taylor	100	Computer Science
1002	Sebastian Giraldo	100	Computer Science
1003	Toni Bollinger	100	Computer Science
1004	Sadagopan R	101	Civil Engineering
1005	Sonia Sharma	101	Civil Engineering

Table 6.5 Facilitator Dimension table structure with sample data

Table 6.5 shows the table structure of the Facilitator dimension. This table contains a list of all professors (at least one per subject), Lab in-charges and librarians.

The Facilitator dimension table can be created using the following SQL statement. (Please refer to **Appendix A** for the complete definition).

```
CREATE TABLE dw.facilitator_dim  (
               facilitator_id INTEGER NOT NULL ,
               facilitator_name CHAR(50) NOT NULL ,
               department_id INTEGER NOT NULL ,
               department_name CHAR(50) NOT NULL ) ;
```

To populate this table, we will use the following SQL statement:

```
INSERT INTO dw.facilitator_dim
SELECT f.facilitator_id, f.facilitator_name,
f.department_id, s.department_name
FROM tx.facilitator_master f
INNER JOIN staging.depts s
       ON f.department_id = s.department_id ;
```

Please note that in this particular case, we could have simply copied the whole table instead of update/insert. That decision is entirely up to the designer of the solution. It is also a function of table size. For large tables, update/insert will be faster than recreating a table.

6.5.5 Fact Table (Attendance fact table)

So far, we have defined four dimension tables, and the fact table we are going to define needs to be compatible with the four dimension tables. So the foreign key reference from the fact table to the dimension tables can be made without any mismatch on data format and size.

The fact table will have six columns. Four columns for joining with four dimension tables and the last two ones will be enrollment_no and enroll_timestamp. We will use count of "enrollment_no" as our **measure**. A **measure** is a metric that provides insight of business state and performance when it is presented in context (dimensions). For example, in our case study, the student count for a particular hour on a particular resource will provide us its utilization.

Table 6.6 shows the table structure of the attendance fact table.

time_*id*	subject_*id*	*resource_id*	facilitator_*id*	enrollment_no
112	231	44	660	95958570
113	354	23	2332	24324324
544	213	21	234	84545334

Table 6.6 Attendance Fact table structure with sample data

Table 6.6 shows the table structure and sample data of the attendance fact table. Even though the enrollment number is a number, we will be using distinct count of enrollment number. We will implement "distinct count" behavior during our OLAP metadata structure definition in Cognos in Section 6.6.

We have complete information now in order to create the ETL job that will read data from source system (attendance system) and modify it as per our destination system (data warehouse dimensions and facts).

Fact table can be created using the following SQL statement.

```
CREATE TABLE dw.attendance_fact  (
               time_id INTEGER NOT NULL ,
               resource_id INTEGER NOT NULL ,
               subject_id INTEGER NOT NULL ,
               facilitator_id INTEGER NOT NULL ,
               enrollment_id INTEGER NOT NULL
               enroll_timestamp TIMESTAMP NOT NULL )  ;
```

To populate fact table we will use an intermediate called staging table (*staging.fact_staging*). This table will have the same structure as the *time_dim* table but with one additional timestamp column. We will be using this timestamp to populate *time_id* for fact table.

To populate *fact_staging* table, we will use the following SQL statement.

```
INSERT INTO dw.fact_staging
(resource_id, subject_id, facilitator_id,
enrollment_id, enroll_timestamp)
SELECT a.resource_id, t.subject_id, t.facilitator_id,
a.enrollment_no, a.enroll_timestamp
FROM tx.attendance_record a
INNER JOIN tx.timetable t
       ON a.resource_id = t.resource_id
WHERE a.enroll_timestamp BETWEEN t.start_time AND t.end_time
       AND t.facilitator_id in
(SELECT facilitator_id FROM dw.facilitator_dim) ;

MERGE INTO dw.fact_staging dest
USING
(SELECT time_id, start_time, end_time FROM dw.time_dim ) src
ON dest.enroll_timestamp BETWEEN src.start_time AND src.end_time
WHEN MATCHED THEN
     UPDATE set time_id = src.time_id;
```

Now that the staging table is populated, we can move the data to the fact table and clean the stage table after that.

```
INSERT INTO dw.attendance_fact
SELECT time_id, resource_id, subject_id,
facilitator_id, enrollment_id
FROM dw.fact_staging;

DELETE FROM dw.fact_staging;
```

Now we have data warehouse in place with tables available along with data.

6.6 Metadata

At this point, we need to implement the business model described in Section 5.4 and Section 6.4.

Cognos offers a large suite of products that covers from small BI projects to the largest ones. In this book, we consider the use of **IBM Cognos Developer Edition**, so that the reader can easily download and install the software to reproduce the items shown in the current case study. Please refer to **Appendix B** for installation instructions.

We use in this book **Framework Manager**, a component of the **IBM Cognos Developer Edition** suite, to import the case study's metadata and define the business representation of the data.

Therefore, as the reader may have noticed, in this example we are not going to use a cube, i.e., a physical multidimensional database.

6.6.1 Planning the Action

As explained in **Chapter 5**, Framework Manager allows the user to import metadata and create a business models that will be the basis for query generation.

So, after defining the business model to create (please check Section 6.4), the reader must follow these steps:

1. create a **project** to handle all information
2. define a connection to a **data source**, that is, the database we created before
3. import **metadata** to recognize source objects (relationships between tables will be automatically identified if they are implemented in the source)
4. create a folder to hold the **business model** objects.
5. create **regular dimensions** to represent the desired business model
6. create the **measure dimensions**, the one that has the metrics to analyze
7. verify **model**
8. create and publish a **package** with this business model

6.6.2 Putting Framework Manager to Work

First step in this process is to create a project, as shown in Figure 6.4.

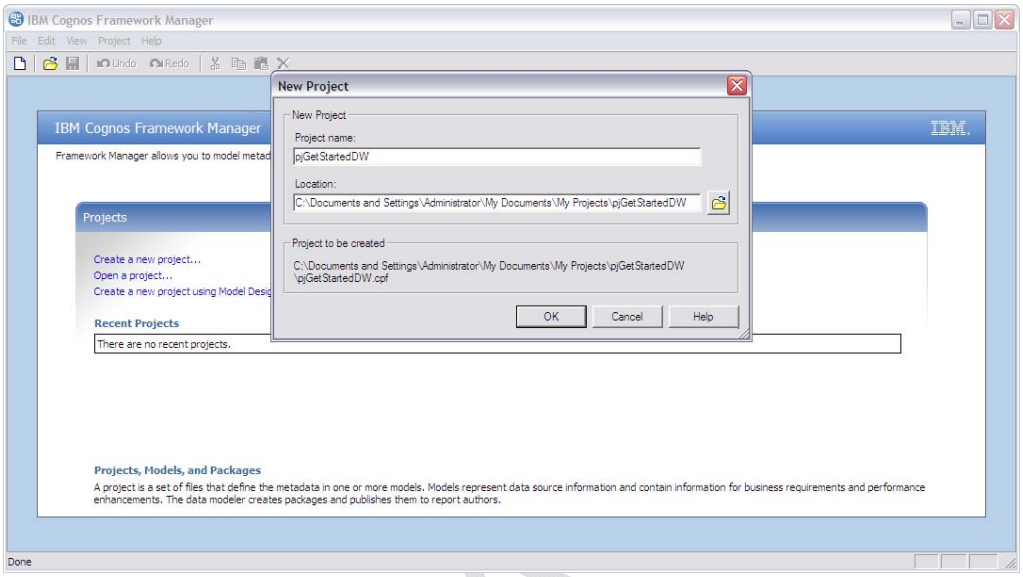


Figure 6.4 Creating a project in Cognos Framework Manager

Second, user needs to define a data source from which Framework Manager will import the metadata. Obviously, the referred data source is our DB2 database. This process is roughly the same thing as in any other software. Reader needs to ask for a **New Data Source**, name it, and specify the source type (IBM DB2). Finally, you will see a window to set up the connection string to that data source. You need to inform database name, your User ID and password (for DB2, the data source). Do not forget to check the **Password** checkbox so you can save your password. Figure 6.5 shows this window. Please make sure to test your connection with these credentials before moving to the next window.

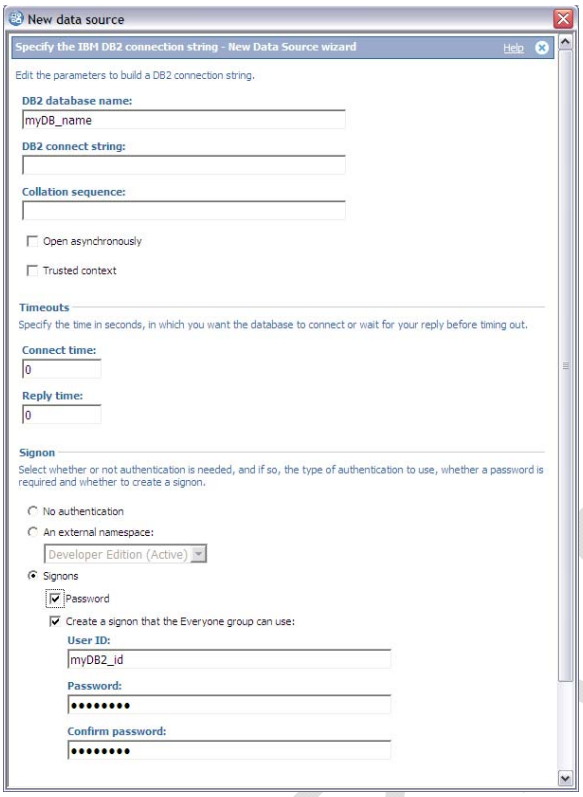


Figure 6.5 New Datasource Wizard – Connection String

Third step is to import metadata from the data source. To make it more understandable, we grouped the objects into three different folders: one for the objects associated to the Attendance System, another for the Staging tables and the last one for the objects from the Datawarehouse. In reality, in our case study we will use only the objects from the Datawarehouse folder. Staging and Attendance System table definition is imported just for reference (if required). Figure 6.6 shows the referred objects.

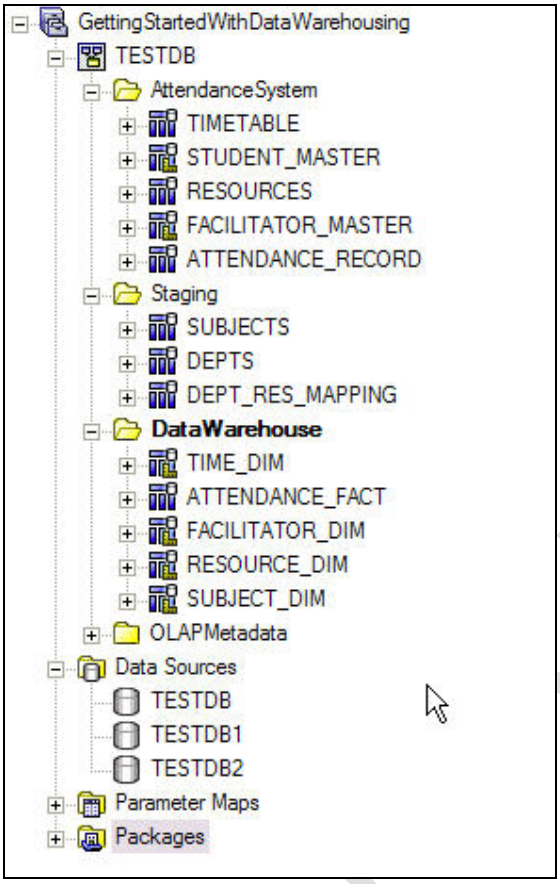


Figure 6.6 Cognos Framework Manager with all source tables

Forth step is to create a namespace or folder to hold the objects we are about to create. To do so, simply right-click on the data source name and choose Namespace or Folder, according to your choice, and give it the name “OLAPMetadata”. All objects we are going to create in this section will be placed on this new folder.

Now the fifth step, we will create the regular dimension TimeDim. To start the process, right-click the OLAPMetadata folder and choose **Create/Regular Dimension** to open the **Dimension Definition – New Dimension** window. Click the button **Add Hierarchy**, mark the **New Hierarchy** line and press F2 to change its name to “TimeDim”. Also edit the level called “New Hierarchy (All)” and change its name to “TimeDim (All)”. Then expand the folder Datawarehouse and the Time_Dim node. Now click the button **Add Level**. Drag the column Time_Name and drop it over the New Level line on the **Hierarchies** box. A new

window opens asking you to select the type of level you are inserting; choose the option **_memberCaption**. This is the most detailed level in this hierarchy, so make sure you check the **Unique Level** checkbox (only one level in each dimension should be the unique level).

At this point, we have a problem related to the nature of the data we have. As you can see in the script shown in Section 6.5.2, the column Time_Name does not have unique values. The labels “Before Session”, “After Session” and others repeat in several rows. This is a problem because this level should have unique values, as mentioned before. So column Time_Name can not be used alone, as it has no identifier at all. Therefore, in this special case, we need to add a second column to the level definition. It is very easy to do that in Framework Manager: simply drag the column Time_ID and drop it on the New Level line too. Make sure the attribute for this column is set to **_businessKey**. Now we have two columns defining this dimension level. That is all you need to do for this particularly special situation. Now, while still marking the New Level line, press F2 and change the level name to “Session”.

Next level to define is the Day level, which is based on column Day_Name. As you can check, this level shows repeated values on several rows. But remember this is not supposed to be a unique level, and so this is ok. Make sure you set for this column both attributes **_businessKey** and **_memberCaption**. Again mark the New Level line, press F2 and change the level name to “Day”.

Repeat the same process to add levels Month, Semester and Year. When you are done, you will see a window similar to Figure 6.7 and you only need to click OK to finish.

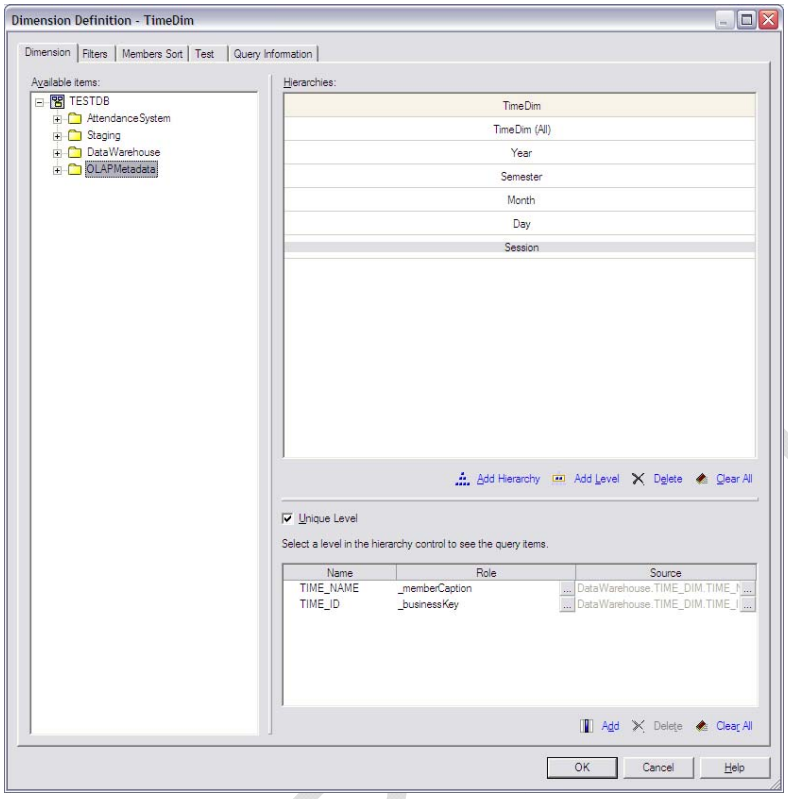


Figure 6.7 Creating Dimension TimeDim

This process must be repeated to create the remaining dimensions: Facilitator, Resource and Subject. An important remark here: those new dimensions are simpler than the Time dimension, because their unique levels consider columns which also have unique values. So, their unique levels will be based on only one column.

The sixth step is to define the measure dimension. In our case study, we will have only one metric, the student count. We need to click the OLAPMetadata folder and create a measure dimension. Again the **Dimension Definition** window will open, but this time the options are different, as we are creating a measure dimension. Now click **Add** in the low right corner of the window. The **Calculation Definition** window will open. Give the name "Student Count" to this metric. Move to the **Functions** pane, expand the **Summaries** group and click on the Count operation to see more details in the **Tips** box. Now double-click it to select. Do not forget to type the keyword DISTINCT in the **Expression Definition** box, as shown in the **Tips** box. Move back to the **Model** pane and expand the folder

Datawarehouse and the node Attendance_Fact in the **Available Components** box. Double-click the component Attendance_fact.Enroll_Timestamp. To finish it all, do not forget to type a close parenthesis character ")" in that sentence. Figure 6.8 shows this window. Click OK to close the windows and give the name "Measures" to the new dimension.

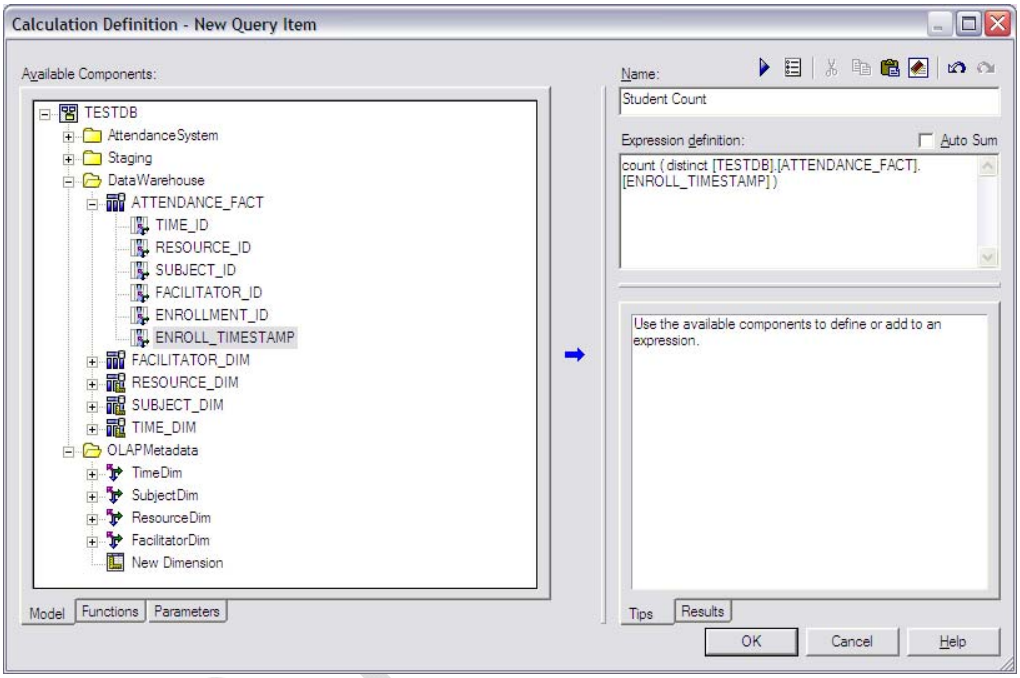


Figure 6.8 Creating the Measure Dimension and the Metric Calculation

At this point, we have the dimensions Time, Resources, Subject, Facilitator and Measures, and we will end up with a model like the one shown in Figure 6.9, which includes all hierarchies and levels.

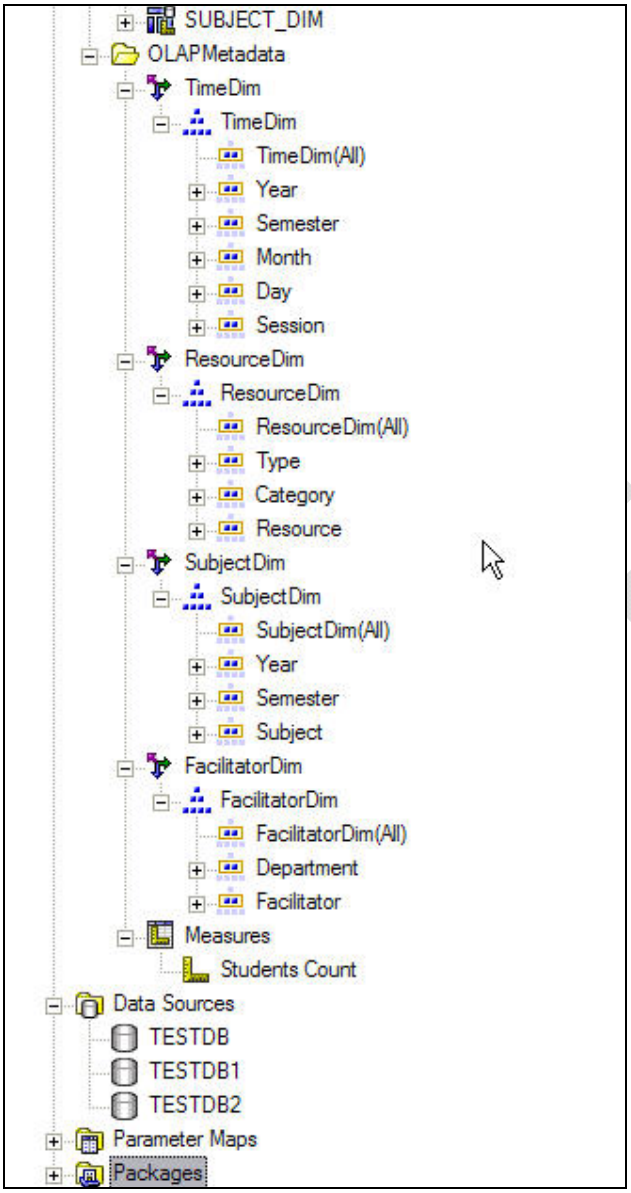


Figure 6.9 Completed OLAP metadata in tree browser

Now we have step 7, which is verifying the created model. To do so, go to the menu **Actions** and click **Verify Model**. This tool can show lots of information, but it is interesting that the novice user restrict messages to **Errors** only. Then click **Verify Model** and check results, as shown in Figure 6.10.

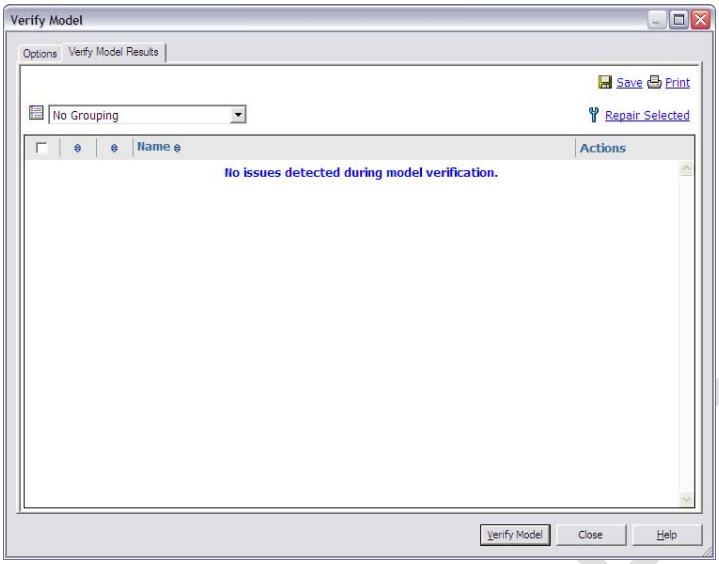


Figure 6.10 Completed OLAP metadata in tree browser

The eighth step is to create a package. On the Framework Manager main window, right-click the **Package** node on the **Project Viewer** pane. Choose **Create** and then **Package**. Name the package as "packGStartDW" and click next. When defining the objects to include, mark only the objects from the folder OLAPMetadata. In the next window, user can choose to add a set of functions to the package, so add the DB2 functions and click **Finish**.

A new window will open asking you if you want to publish the package using the Publish Wizard. This is a necessary step, so click Yes. This is a Next-Next-Publish process. And we are finally done with the business model.

6.7 Reporting

Once the metadata (facts, dimensions and measures created above) is published, we are ready to create reports required in our case study.

In this section, we use the **Report Studio** to create reports. To run it, open **IBM Cognos Developer Edition**, click **Launch** in the top right side of your page and select **Report Studio**, as shown in Figure 6.11.

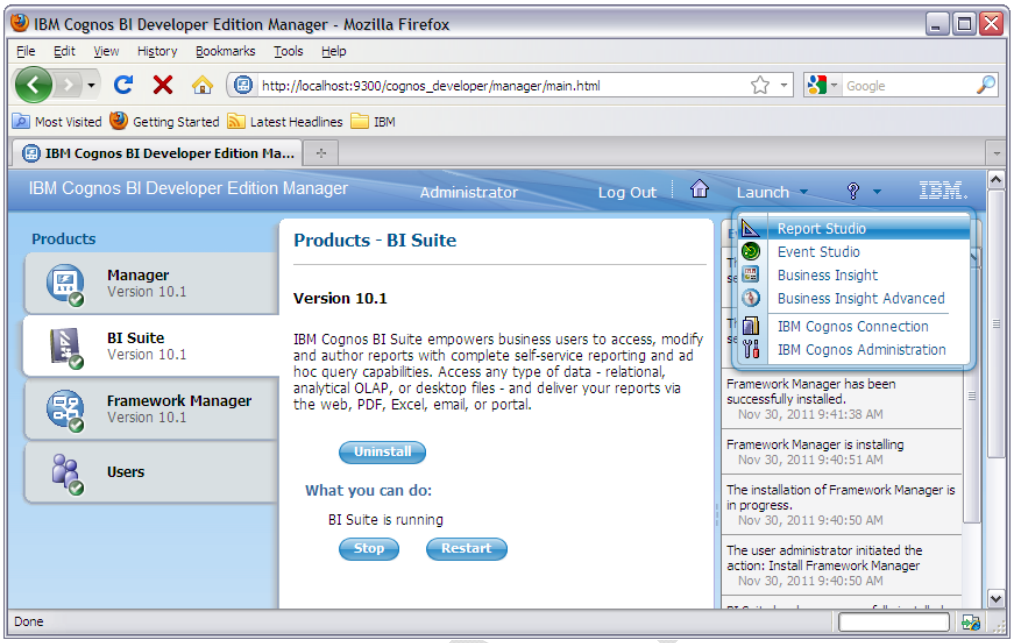


Figure 6.11 Opening Report Studio

First thing you see in the new page is the package you just created, packGStartDW. Click on the package to create a new report, but make sure your browser allows pop-up windows. Click **Create New** then you only need to select the report format as a table or crosstab or bar/line/pie chart and simply drag and drop dimensions around measure. To see results, click on the **Run Report** button in the toolbar.

At this point, we are able to create reports as indicated in **Chapter 5**. Figure 6.12 shows the student count by facilitator departments' year wise. The numbers shown here will be highly dependent on the quality of data in fact table. In our case study, we have used random number generator to populate our *attendance_record* table in attendance system therefore, we will not be able to conclude any useful information from this report. The point to emphasize here is that data source has to have reliable data in order to get a useful, informative and actionable report out of it.

Students Count	Year 1	Year 2	Year 3	Year 4
COMPUTER SCIENCE	10	9	4	9
CIVIL ENGINEERING	5	1	10	3
LABS	2	2	2	2
LIBRARY	1	4	3	3

Figure 6.12 Students count by subject (year) by facilitators

Figure 6.13 shows another sample report on student count by subjects (year) resource.

Students Count	Year 1	Year 2	Year 3	Year 4
Lecture Room	4	9	13	9
Library	11	2	2	3
Lab	3	5	4	5

Figure 6.13 Student count by time by resources

One conclusion we can draw from the report shown in Figure 6.13 is that first year students use library the most and third year students attend lectures the most. Actionable item as after effect of this report can be

- library does not have enough useful books for subjects offered in year 2 onwards, this requires a detailed investigation of the reason.
- first year students generally use library more to find out which books to buy/procure.
- facilitators of subjects in year 3 are really teaching well and therefore attendance is good in those subjects.

We will not be discussing or inferring above reports exhaustively here. One thing to note here is that absolute numbers in above reports do not show accurate picture of the situation. For example, the entire first year may have only four classes, and the third year has twenty classes. That would mean that the first year is attending 100% classes which is not the case with the third year. Therefore, above reports will present data/information

better if numbers are presented as percentage of total instead of actual numbers. This change of absolute number to percentage can be implemented as exercise to this chapter.

6.8 Summary

After implementing the solution in this case study, we should now know what it takes to implement a warehouse and develop a reporting system on top of it. Obviously the data model we used here is a simplified model, created solely to make it easier for the inexperienced reader to follow all the steps necessary to implement a real-world data warehouse and BI system.

We have generated random data in our case study because the data (attendance) is spread across twenty four hours. However in real world the data will show a very high concentration between 8 AM to 5 PM and almost negligible around midnight/early morning. We would also need to account for before college hours and after college hours while populating the fact_staging table, which is not done in current implementation.

6.9 Exercises

1. Implement all the enhancements that proposed in this chapter.

2. Another enhancement to the current model that you can try is to define and implement the necessary changes so that the university could track the current metrics (Lecture room, Library and Labs utilization) down to individual students.

7

Chapter 7 – Data Warehouse Maintenance

In this chapter, we will introduce tasks required to keep a data warehousing up and running, complying with performance criteria defined in **service level agreement** (SLA) for the data warehouse. In order to comply with SLA, the data warehouse will require prompt administration and maintenance. The system should also have resiliency to all single point of failures.

7.1 The Big Picture

Database maintenance is a routine activity, like maintenance of vehicles or buildings. In order to maintain sustained performance of the system, it should be monitored continuously via automated means, and an alert should be generated for any malfunction or usual behavior observed. Figure 7.1 shows a high-level view of database maintenance activities.

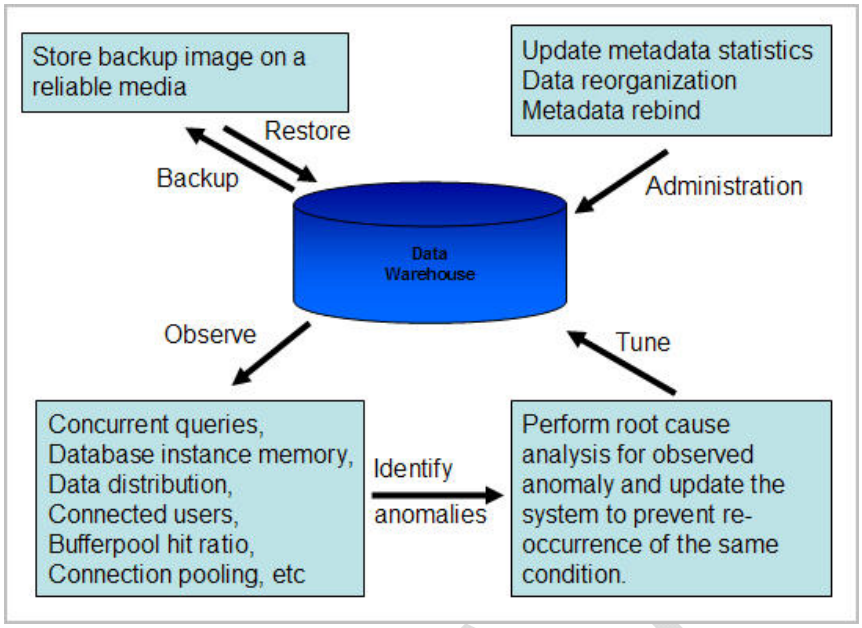


Figure 7.1 Holistic view of data warehouse maintenance activities

7.2 Administration

Data warehouse administration is primary task that is required to make data warehouse up and running. Now let us discuss details about who can do the administration and what it is needed.

7.2.1 Who Can Do the Database Administration

Authority levels provide a method of grouping privileges and higher-level database manager maintenance and utility operations. **Database authorities** enable users to perform activities at the database level. A user, group, or role can have one or more of the following authorities:

- Administrative authority level that operates at the instance level, SYSADM (system administrator)
- Administrative authority levels that operate at the database level:
  - DBADM (database administrator)

The DBADM authority level applies at the database level and provides administrative authority over a single database. This database administrator possesses the privileges required to create objects, issue database commands, and access table data. The database administrator can also grant and revoke CONTROL and individual privileges.

- SECADM (security administrator)

  The SECADM authority level applies at the database level and it is related to security administration, such as the authority required to create, alter (where applicable), and drop roles, trusted contexts, audit policies, security label components, security policies, and security labels, which are used to protect tables. It is also the authority required to grant and revoke roles, security labels and exemptions as well as to grant and revoke the SETSESSIONUSER privilege. A user with the SECADM authority can transfer the ownership of objects that they do not own. They can also use the AUDIT statement to associate an audit policy with a particular database or database object at the server.

  The SECADM authority has no inherent privilege to access data stored in tables and has no other additional inherent privilege. This authority can only be granted a user by a SYSADM user. The SECADM authority can be granted to a user but cannot be granted to a group, a role or to PUBLIC.

 System control authority levels that operate at the instance level:

- SYSCTRL (system control)

  The SYSCTRL authority level provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority.

- SYSMAINT (system maintenance)

  The SYSMAINT authority level provides the authority required to perform maintenance operations on all databases associated with an instance. A user with SYSMAINT authority can update the database configuration, backup a

database or table space, restore an existing database, and monitor a database. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority.

- The SYSMON (system monitor) authority level

  SYSMON provides the authority required to use the database system monitor. It operates at the instance level.

- Database authorities

  To perform activities such as creating a table or a routine, or for loading data into a table, specific database authorities are required. For example, the LOAD database authority is required for use of the load utility to load data into tables (a user must also have INSERT privilege on the table).

Figure 7.2 illustrates the relationship between authorities and their span of control (database, database manager).

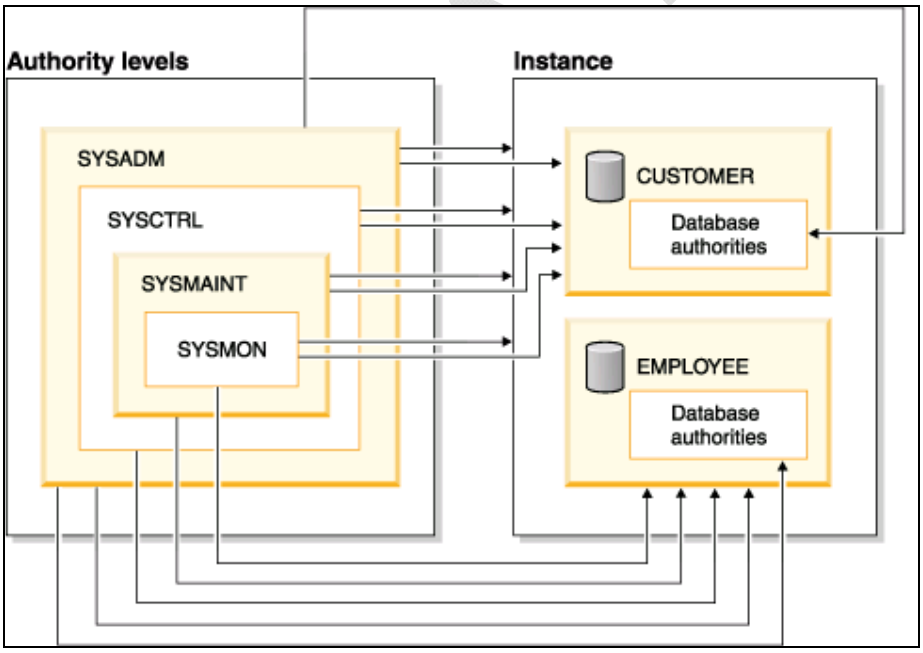


Figure 7.2 Hierarchy of Authorities

For more details on DB2 authorities, privileges and database object ownership, please refer to IBM Data Studio [30] or IBM DB2 Information Center [24].

7.2.2 What To Do as Database Administration

Important administration tasks are:

- Use workload management to manage user concurrency, query concurrency and system availability. Workload management is very much like a traffic signal at intersection. A traffic signal helps in maintaining flow of traffic from all directions and for all types of vehicles. Due to this traffic signal, some vehicles may have to wait for sometime for their turn to cross the junction. Similarly, workload management in database makes sure that all types of queries can be executed, from all valid system users. Here as well, a query or the user may have to wait if system resources are not available, or rules defined for that particular user/query does not allow immediate execution. For more details on IBM DB2® Workload Management, please refer to IBM Redbook “DB2 Workload Manager for Linux, UNIX, and Windows” [25]. Workload management best practices [26] are also available for handy reference. This best practice document covers workload management design, implementation and monitoring.
- Capture high water mark for number of
  - Database connections

    This helps database administrator to be aware of any unknown behavior to application or user queries.
  - Database users

    This helps database administrator to keep track of applications or users logging into system. Every connected user uses database server resources even if no query is executed on that connection. Therefore for optimal usage of database server resources, it is important to allow only desired users and applications to keep the connection idle.
- Keep history of changes in warehouse.

  Database objects and environment change log helps administrator to keep track all changes applied to database. This information is very helpful in event of database malfunction. Not only the change can be reverted (if system exhibits any issues

after this change), DBA can share this change set with product vendor to identify potential reason for database malfunction.

- Keep system as per the tested warehouse product stack

  All software and hardware products are supported on a well-defined environment. This environment includes hardware platform, firmware versions, operating system versions, compatibility with other applications etc. DBA should make sure that either there is no deviation from supported hardware and software stack, or all deviation should be approved from vendor before change implementation.

- Keep watch on database logs, operating system logs and other appropriate logs

  DBA should monitor appropriate logs (database, operating system, application) so that any suspicious event can he detected and handled in time. Absence of log monitoring leads to system failures leading to unplanned downtime.

- Scheduled downtime

  DBA should always plan for schedule downtime as permitted by SLA to perform any maintenance activity. Even if system has no symptoms of abnormal behavior, maintenance should be performed as a planned database job.

7.3 Database Objects Maintenance

In addition to system maintenance, database objects maintenance is also required to keep database performance in good health. The following set of database activities are required on an ongoing basis.

- Statistics updates

  Database engine collects statistics like table cardinality, column cardinality, index definitions etc. from database objects. These statistics are used by database engine to retrieve data efficiently.

  These statistics can be updated either explicitly (via `runstats` command) or automatically as and when system finds it appropriate to update the statistics.

  To enable automatic statistics updates, database configuration parameters `AUTO_MAINT` and `AUTO_RUNSTATS` should be set to `ON`.

- Data reorganization

I/O speed (read and write) is also a function of data layout on physical disk. Over a period, data becomes fragmented on disks and therefore takes longer to complete table read and write operations.

This data reorganization can be done either explicitly (via `reorg` command) or automatically as and when system finds it appropriate to reorganize the data.

To enable automatic data reorganization, database configuration parameters `AUTO_MAINT` and `AUTO_REORG` should be set to `ON`.

- Check data skew

Data skew can lead to severe imbalance in the warehouse and influence its performance downwards drastically. Large data warehouses follow shared nothing architecture (SNA) as a result each logical database partition does not shares data with any other partition. Therefore, if any single database partition is heavily loaded with data relative to other partitions, then this single partition can become a potential bottleneck in the system and degrade performance of entire system.

Data skew for a table can be found by following SQL query in IBM DB2®

```
SELECT dbpartitionnum(partition_key_column_name),
       COUNT(*)
FROM table_name
GROUP BY dbpartitionnum(partition_key_column_name);
```

In above example, `partition_key_column_name` is the name of column that is used to partition the table `table_name`.

- Identify unused objects

When a data warehouse grows larger, the growth of each database object (e.g., table, index, materialized table, aliases, and nicknames) creates additional workload for the database manager (engine). Therefore, it is very important to identify objects that are no longer in use by database users or applications. Over a period, it is very common to appear obsolete objects. Typical examples of such database objects are tables, views, indexes and materialized tables. It will make the maintenance of a warehouse easier if unused objects are removed from the system.

7.4 Backup and Restore

Backup and **restore** of database and transaction logs is a method to minimize data loss in an event of system failure or a disaster. Backup creates a replica of database by capturing all database objects (metadata, data layout, data and transaction logs) in a way that the whole database is recreated to the same state it was at the time when that image was taken. This replica is typically a data file, stored either on an external disk or tape.

Restore command uses this replica to recreate the database, fully or partially, with an option of point-in-time recovery.

IBM DB2® supports following two types of database backup.

- Offline backup

  Offline backup, also known as “cold backup”, is a backup type where users of the warehouse system cannot connect to the database at the time. Offline backup creates an image of the whole database, so it demands that every database connection is terminated prior to its start. This avoids the risk of data pages updates when the page is being backed up. But, of course, full backups require system down time.

- Online backup

  Online backup, also known as “hot backup”, is a backup mode where users of the system can execute queries (insert, update and delete) while the database backup is in progress. Data pages that are changed when the backup is in progress can be recovered by point-in-time recovery using database logs roll-forward.

IBM DB2® supports following types of database backup strategies.

- Full database backup

  Full database backup, as the name suggests, creates a database backup file that is capable of regenerating entire database (via restore command). Full database backup takes up more file system space for backup file and takes longer to create backup file for large databases, which it is generally the case of data warehouses.

  The BACKUP command includes several options. Below you see a basic BACKUP command that takes a full backup of a database named *myDB* and puts this backup image in a directory named *destinationDirectory*.

  ```
  BACKUP DATABASE myDB to destinationDirectory
  ```

- Incremental database backup

  In incremental database backup, the RDBMS backs up only those data pages that were changed since last full backup and there is no need for down time. In databases where the volume of changed data is high compared to the entire database volume, the incremental backup may not be suitable, as it will end up backing up all data pages. Therefore incremental backup strategy is useful in databases where changed data pages are comparatively smaller than the size of database. (This is generally the case of data warehouses).

  Incremental database backup can be enabled by setting database configuration parameter `TRACKMOD` to `YES`. Default value of this parameter is `NO`.
  After `TRACKMOD` is set to `YES`, one must back up the database before allowing applications to change data. In other words, one must take a full database backup so that a baseline exists against which incremental backups can be taken.
  Following command would take incremental backup of database `myDB`.

  ```
  BACKUP DB myDB INCREMENTAL
  ```

- Delta database backup

  In delta database backup, the RDBMS backs up all pages changed since the last backup of any kind (delta, incremental, or a full backup image). This backup is usually much faster than the other two backup strategies described above. But there is also riskier! Suppose you take a full backup on Monday and then take delta incremental backups on the other days. In case you need to restore a backup taken on Thursday, you will need to restore the Monday full backup and then restore the Tuesday, Wednesday and Thursday delta backups, that is four backup files. And you do need all four backup images to accomplish this task. If any of those backups are corrupted for some reason, there is no way to restore the remaining images.

  Following command will take delta backup for database `myDB`.

  ```
  BACKUP DB myDB INCREMENTAL DELTA
  ```

Once database backup is complete, it is strongly recommended to perform consistency check on backup file. Backup file may contain some corrupted data bits due to disk, I/O or application errors. IBM DB2® has an in-built backup file consistency check utility. This utility can be used as follows:

```
db2ckbkp <absolute_path_of_backup_file_name>
```

Backup file can also be used to read database objects properties in backup file in a limited way. Above command, when used with additional argument "`-a`" displays all possible information from backup file, and verifies it as well. For more information on `db2ckbkp` options, please refer to IBM DB2® 9.7 for Linux, Unix and Windows Information Center [24].

7.5 Data Archiving

Archiving is the process of moving inactive data to another storage location that can be located and accessed when needed. Data archiving is an excellent solution for the problems that are associated with ever increasing volumes of data.

Archiving is not the same as creating backups, because archive retrievals can be selective, whereas backups are not. However, archives can be a component of a backup and recovery scheme. Furthermore, archives are application oriented, whereas backups are oriented to the data store.

7.5.1 Need for Archiving

Several factors present the need for data archiving. First, the volume of data in databases is always growing. Second, much of this data is inactive but it must be maintained for business reasons. Third, it is important to maintain data as inexpensively and effectively as possible.

Data centers are concerned with ever increasing volumes of data. And although the cost of storing data is getting lower day by day, the databases are growing at an unprecedented rate. Some inactive data resides everywhere because of historical transactions, the use of data warehouses, and other reasons. Inactive data is not in use by its users; however, it is less likely to be accessed. Several reasons why we might be maintaining inactive data include:

- Compliance requirement by government
- In anticipation of the need for building future trend analysis
- Need of maintaining a complete history of customers

As the size of a database grows, the percentage of inactive data often grows as well, which can result in:

- Performance problems from additional input/output (I/O) and extensive processing that impacts users
- Manageability problems because large objects become difficult to reorganize and size limits are reached
- Extensive hardware and storage costs

An excellent solution to these problems is data archiving.

7.5.2 Benefits of Archiving

Archiving data reclaims disk space, lowers storage costs, reduces object maintenance, and improves the performance of active data.

The major benefits of archiving are interrelated. By reclaiming disk space that was consumed by inactive data, we can effectively lower our storage costs and reduce object maintenance. As a result, access to our active data becomes more efficient.

7.5.3 The importance of Designing an Archiving Strategy

An archiving strategy is a plan that ensures that we are archiving the correct data in the correct way. Archiving without an archiving strategy might result in poor overall system performance.

The most successful archiving implementations are those that are well planned and consider several important aspects. The following sections provide recommendations for creating a well-planned archiving strategy, as well as several pitfalls to avoid that can create problematic archiving schemes.

Begin by assessing our needs for archiving. Answer the following questions:

- What data do we need to archive?
- When do we need to archive this data?
- How long do we need to maintain the archived data?
- Do we need SQL access to the archived data?

- Should the archived data be stored in DB2® table archives or in flat-file archives?
- Under which circumstances and how often will we need to retrieve archived data from either table archives or file archives?
- Do we require archived data that is stored in table archives to be moved to flat-file archives (in other words, do we require a multitier archiving strategy)?

After we have answered these questions, we can begin to use DB2 Data Archive Expert [28] to build our archives. Once data is archived, it can be read again via loading onto database again. IBM Optim® [29] allows data retrieval without restoring or loading archive data on a database.

7.6 Summary

In this chapter, we briefly discussed data warehouse administration, maintenance, backup, restore and archiving. All these activities are performed by data warehouse DBAs. Administration and maintenance are very loosely connected with level business objective that a data warehouse fulfills. However, backup, restore and archiving are very tightly coupled with business objectives. In other words, warehouse administration and maintenance is driven by application that uses warehouse. Backup/restore and archiving are directly driven by the project or the organization's corporate policies. Data warehouse administration and maintenance activities will keep the system at optimum performance level. However we must consider that any system is prone to data loss (in event of software or hardware errors). To counter for the data loss possibility, backup and restore tools and process are implemented. With continuously increasing data size, it may be either not practical to keep all data accessible all the time, or it is too expansive to keep that much data online. At this point, an archiving solution is what provides a good return on investment (ROI).

8

Chapter 8 – A Few Words about the Future

In this chapter, we will present a few words about some new efforts and market trends that may affect and/or transform the way companies will use their data warehouses in the near future.

8.1 The Big Picture

Any exercise to foresee the future is always prone to error. Anyway, we have the need to look ahead and try to identify which way to go.

This is true not only for the companies who want to use their data warehouses to predict which actions may or may not be successful, but also for professionals who try to foresee how the companies will use their data in an attempt for increasing their profits.

Since the mid-90's, the market saw a boom in data warehouse and BI efforts. Nowadays even small companies know the value of better understanding the data they have and so they are also investing in data warehouses and BI systems.

Until now, data warehouses helped companies mostly by showing them how to correct and/or improve their actions to make more money. But those actions are primarily corrective ones.

In business today, companies are looking for new technologies and/or methodologies that allow them to take proactive actions instead of only corrective ones. So knowing the past (and this is primarily what we store today in data warehouses) seems not to be enough any longer.

Therefore, major players in the BI market like **IBM** are investing in analytical solutions such as **Business Analytics and Optimization** (**BAO**).

The whole idea behind BAO is that it is more effective for the companies when their executives take decisions based on analytical information instead of insights only. And these analytical information may not be based on past data only, as sales, for instance. There are several situations where the companies need different data, such as '*how many customers passed near the shelf and did not buy our product*'. So companies may need to change the way they gather information and define well-designed strategies on which data to collect.

BAO is a subject for a book by itself and the purpose here is simply to advise the reader about the trends in this market. For more information about BAO, please visit www.ibm.com/bao .

A

Appendix A – Source code and data

For detailed information about the necessary software to run all the scripts and steps described in this book, please refer to **Appendix B**. The reader can also find there information on how to use the recommended software.

Before running the scripts below, the reader must create a database or use the SAMPLE database provided by the DB2 installation process.

The sequence of the statements shown below must be obeyed otherwise some error messages will be generated when creating those tables and constraints.

It might be useful to take a look at the two diagrams shown below, representing the Attendance System (Figure A.1) and the DataWarehouse (Figure A.2).

In the following pages, the reader will find detailed information about the table creation and table population. For convenience, all SQL scripts and data files mentioned below are zipped in a single file, which is available as a separate download.

So let's get started.

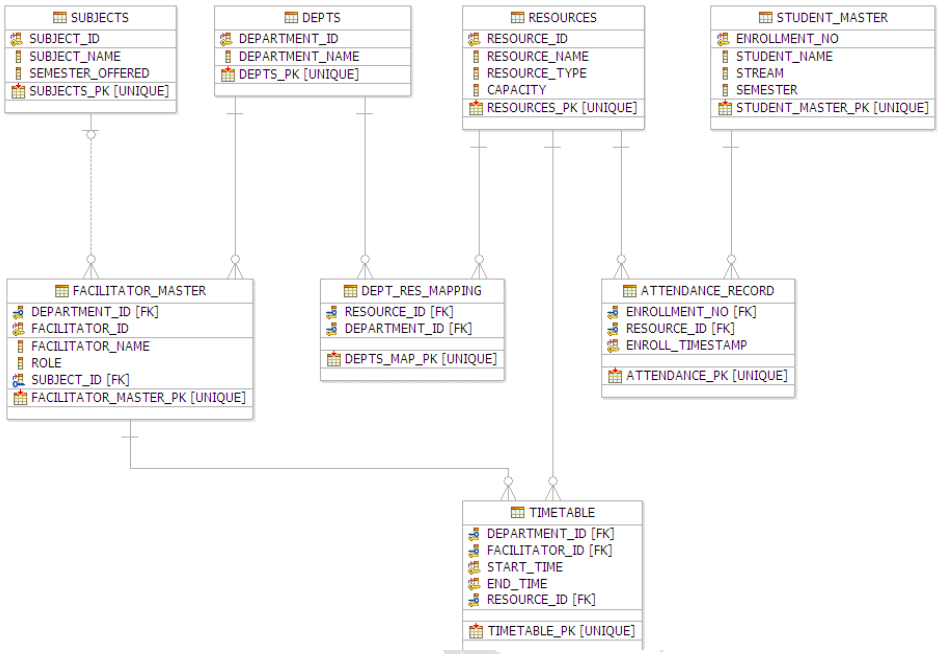


Figure A.1 Database model for the Attendance System

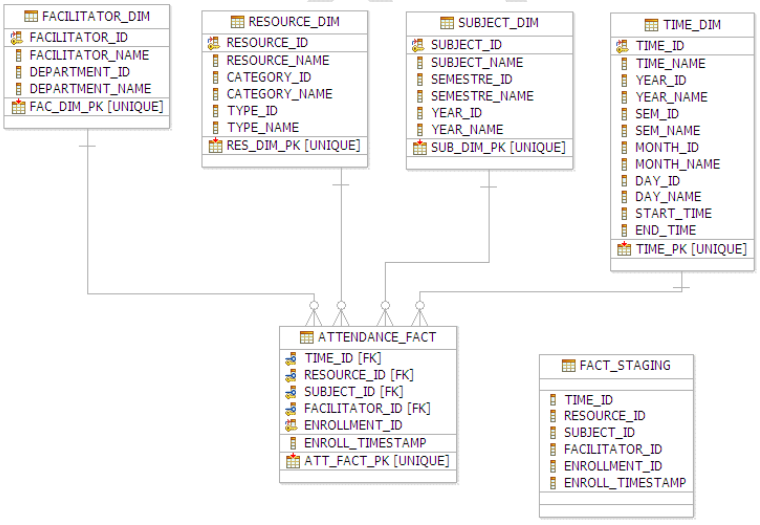


Figure A.2 Database model for the Datawarehouse

A.1 Staging Tables Creation and Data Generation

Use following SQL statements to create and generate data for staging tables. To keep the process as simple as possible, we are using SQL insert statements as much as we can. For larger tables, we use load commands. In production environments, staging tables are usually populated and updated by automated ETL jobs.

Department Table

```
CREATE TABLE staging.depts (
      department_id INTEGER NOT NULL GENERATED BY DEFAULT AS
            IDENTITY (START WITH 1 INCREMENT BY 1 MINVALUE 1
            MAXVALUE 32672 NO CYCLE NO CACHE),
      department_name CHAR(50) NOT NULL
      )
      DATA CAPTURE NONE ;
ALTER TABLE staging.depts ADD CONSTRAINT depts_pk PRIMARY KEY
(department_id);

INSERT INTO staging.depts VALUES
(DEFAULT,'COMPUTER SCIENCE'),
(DEFAULT,'CIVIL ENGINEERING'),
(DEFAULT,'LABS'),
(DEFAULT,'LIBRARY');
```

Subject Table

```
CREATE TABLE staging.subjects (
      subject_id INTEGER NOT NULL GENERATED BY DEFAULT AS
            IDENTITY (START WITH 1 INCREMENT BY 1 MINVALUE 1
            MAXVALUE 32672 NO CYCLE NO CACHE),
      subject_name CHAR(50) NOT NULL ,
      semester_offered SMALLINT NOT NULL
      )
      DATA CAPTURE NONE ;
ALTER TABLE staging.subjects ADD CONSTRAINT subjects_pk PRIMARY KEY
(subject_id);

INSERT INTO staging.subjects VALUES
(DEFAULT,'Fluid Mechanics-I', 3),
(DEFAULT,'Thermodynamics-I', 3),
(DEFAULT,'Materials', 4),
(DEFAULT,'Engineering Drawing', 2),
(DEFAULT,'Data Structures and Algorithms', 1),
(DEFAULT,'Calculus', 1),
(DEFAULT,'Solid state physics', 2),
(DEFAULT,'Computer System Architecture', 5),
(DEFAULT,'Fluid Mechanics-II', 5),
(DEFAULT,'Thermodynamics-II', 6),
(DEFAULT,'Water works', 6),
(DEFAULT,'Power plants', 5),
(DEFAULT,'Formal Languages and Automata Theory', 3),
(DEFAULT,'Elementary Computer Graphics', 2),
(DEFAULT,'Computer Communications', 4),
(DEFAULT,'System Software', 3);
```

A.2 Attendance System Metadata and Data Generation

Following set of commands/scripts will create tables, constraints and generate sample data for the attendance system that we will use for ETL, warehousing and reporting.

Resource

```
CREATE TABLE tx.resources (
      resource_id INTEGER NOT NULL GENERATED BY DEFAULT AS
            IDENTITY (START WITH 1 INCREMENT BY 1 MINVALUE 1
            MAXVALUE 32672 NO CYCLE NO CACHE),
      resource_name CHAR(20) NOT NULL,
      resource_type CHAR(20),
      capacity SMALLINT
      )
      DATA CAPTURE NONE ;
ALTER TABLE tx.resources ADD CONSTRAINT resources_pk PRIMARY KEY
(resource_id);
LOAD FROM 'c:\dwbook\tx.resources.TXT' OF DEL INSERT INTO
tx.resources;
-- the file below contains the source data to the table above
-- the first column is empty because the table uses
-- a IDENTITY column
-- the above LOAD statement assumes you place the source file in
-- folder c:\dwbook\. (You can adapt this command line as you wish)
```

Student Master Table

```
CREATE TABLE tx.student_master (
      enrollment_no INTEGER NOT NULL GENERATED BY DEFAULT AS
            IDENTITY (START WITH 1 INCREMENT BY 1 MINVALUE 1
            MAXVALUE 32672 NO CYCLE NO CACHE),
      student_name CHAR(50) NOT NULL,
      stream CHAR(10) NOT NULL,
      semester SMALLINT NOT NULL
      )
      DATA CAPTURE NONE ;
ALTER TABLE tx.student_master ADD CONSTRAINT student_master_pk
PRIMARY KEY (enrollment_no);
LOAD FROM 'c:\dwbook\tx.student_master.TXT' OF DEL INSERT INTO
tx.student_master (student_name, stream, semester);
-- the above LOAD statement assumes you place the source file in
-- folder c:\dwbook\. (You can adapt this command line as you wish)
```

Facilitator Master Table

```
CREATE TABLE tx.facilitator_master (
      facilitator_id  INTEGER NOT NULL GENERATED BY DEFAULT AS
            IDENTITY (START WITH 1 INCREMENT BY 1 MINVALUE 1
            MAXVALUE 32672 NO CYCLE NO CACHE),
      facilitator_name CHAR(50) NOT NULL,
      department_id INTEGER NOT NULL,
      role CHAR(10),
      subject_id INTEGER
      )
      DATA CAPTURE NONE ;
ALTER      TABLE      tx.facilitator_master      ADD      CONSTRAINT
facilitator_master_pk
      PRIMARY KEY (department_id, facilitator_id );
ALTER TABLE tx.facilitator_master ADD CONSTRAINT facilitato_dept_fk
      FOREIGN KEY (department_id)
      REFERENCES staging.depts (department_id)
      ON DELETE CASCADE;
ALTER TABLE tx.facilitator_master ADD CONSTRAINT facilitato_subj_fk
      FOREIGN KEY (subject_id)
      REFERENCES staging.subjects (subject_id)
      ON DELETE CASCADE;
LOAD FROM 'c:\dwbook\tx.facilitator_master.TXT' OF DEL INSERT INTO
tx.facilitator_master   (facilitator_name,   department_id,   role,
subject_id);
SET INTEGRITY FOR tx.facilitator_master IMMEDIATE CHECKED;
-- the above LOAD statement assumes you place the source file in
-- folder c:\dwbook\. (You can adapt this command line as you wish)
```

Department X Resource Mapping Table

```
CREATE TABLE staging.dept_res_mapping  (
             resource_id INTEGER NOT NULL ,
             department_id INTEGER NOT NULL ) ;
ALTER TABLE staging.dept_res_mapping ADD CONSTRAINT depts_map_pk
PRIMARY KEY (resource_id , department_id);
ALTER TABLE staging.dept_res_mapping ADD CONSTRAINT map_dept_fk
      FOREIGN KEY (department_id)
      REFERENCES staging.depts (department_id)
      ON DELETE CASCADE;
ALTER TABLE staging.dept_res_mapping ADD CONSTRAINT map_res_fk
      FOREIGN KEY (resource_id)
      REFERENCES tx.resources (resource_id)
      ON DELETE CASCADE;
LOAD FROM 'c:\dwbook\st.dept_res_mapping.TXT' OF DEL INSERT INTO
staging.dept_res_mapping ;
SET INTEGRITY FOR staging.dept_res_mapping IMMEDIATE CHECKED;
```

Timetable

```
CREATE TABLE tx.timetable (
            start_time TIMESTAMP NOT NULL,
            end_time TIMESTAMP NOT NULL,
            resource_id INTEGER NOT NULL,
            department_id INTEGER NOT NULL,
            facilitator_id INTEGER NOT NULL
      )
      DATA CAPTURE NONE ;
ALTER TABLE tx.timetable ADD CONSTRAINT timetable_pk PRIMARY KEY
(department_id,facilitator_id,start_time,end_time, resource_id);
ALTER TABLE tx.timetable ADD CONSTRAINT time_facilitato_fk FOREIGN
KEY (department_id, facilitator_id )
      REFERENCES       tx.facilitator_master       (department_id,
facilitator_id )
      ON DELETE CASCADE;
ALTER TABLE tx.timetable ADD CONSTRAINT time_resource_fk FOREIGN
KEY (resource_id )
      REFERENCES tx.resources (resource_id )
      ON DELETE CASCADE;
LOAD   FROM   'c:\dwbook\tx.timetable.TXT'   OF   DEL   INSERT   INTO
tx.timetable (start_time, end_time, department_id, facilitator_id,
resource_id);
SET INTEGRITY FOR tx.timetable IMMEDIATE CHECKED;
-- notice the LOAD statement shows a different sequence of columns
-- than the original table, due to the source file layout
-- the above LOAD statement assumes you place the source file in
-- folder c:\dwbook\. (You can adapt this command line as you wish)
```

Attendance Records Table

Executing following command once will insert one record. Execute is as many times possible or as expected by the case study (40 students per semester, 2 engineering streams, 4 classes everyday, 4 semesters running in parallel would mean 40x2x4x4 = 1280 rows per day), plus students will also visit labs and the library. Assuming average rows per day to be 1000, attendance record table should have 31000 rows for month of Aug 09. Following statement generates data for only Aug 09, one row at a time. Therefore you should execute this statement as many times as possible.

In case you prefer to use the suggested stored procedure (shown below) and you are using IBM Data Studio, please beware that procedures must be implemented in Data Studio using a special type of project object, called 'Stored Procedures'. If you try to implement it through the regular SQL Scripts object, it will generate an error message and implementation will fail.


```
CREATE TABLE tx.attendance_record (
            enrollment_no INTEGER NOT NULL,
            resource_id INTEGER NOT NULL,
            enroll_timestamp TIMESTAMP NOT NULL
      )
      DATA CAPTURE NONE ;
ALTER TABLE tx.attendance_record ADD CONSTRAINT attendance_pk
PRIMARY KEY (enrollment_no, resource_id, enroll_timestamp);
ALTER TABLE tx.attendance_record ADD CONSTRAINT attenden_stude_fk
FOREIGN KEY (enrollment_no)
      REFERENCES tx.student_master (enrollment_no)
      ON DELETE CASCADE;
ALTER TABLE tx.attendance_record ADD CONSTRAINT attendance_reso_fk
FOREIGN KEY (resource_id)
      REFERENCES tx.resources (resource_id)
      ON DELETE CASCADE;
--repeat this last statement as many times as possible
INSERT INTO tx.attendance_record VALUES (1+ int(rand()*274), 1+
int(rand()*16),timestamp('2009-08-01-00.00.00.000000')            +
```

```
int(rand()*30) day + int(rand()*24) hour + int(rand()*60) minute +
int(rand()*60) second );

-- another option is to create a procedure an run it once
-- in case you are using DATA STUDIO,
-- please verify how to create a stored procedure
CREATE PROCEDURE tx.sp_attendance_record (@NumberOfRepetitions INT)
LANGUAGE SQL
DYNAMIC RESULT SETS 1
BEGIN
WHILE @NumberOfRepetitions > 0 DO
      INSERT INTO tx.attendance_record VALUES (1+int(rand()*274),
1+     int(rand()*16),timestamp('2009-08-01-00.00.00.000000')     +
int(rand()*30) day + int(rand()*24) hour + int(rand()*60) minute +
int(rand()*60) second );
      SET @NumberOfRepetitions = @NumberOfRepetitions - 1 ;
END WHILE ;
END
-- execute procedure to insert 1000 records
CALL tx.sp_attendance_record (1000);
```

A.3 Data Warehouse Data Population

To populate data for data warehouse, SQL statements are provided at respective section in **Chapter 6** (with sample data). Please note that we need to populate dimension tables first before fact table due to foreign key constraints/references of fact table to dimension tables. Only time dimension generation is given here.

Time Dimension

```
CREATE TABLE dw.time_dim (
      time_id INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
                START WITH +1
                INCREMENT BY +1
                MINVALUE +1
                MAXVALUE +32672
                NO CYCLE
                NO CACHE
                NO ORDER ) ,
      time_name CHAR(50) NOT NULL ,
      year_id INTEGER NOT NULL ,
      year_name CHAR(50) NOT NULL ,
      sem_id INTEGER NOT NULL ,
      sem_name CHAR(50) NOT NULL ,
      month_id INTEGER NOT NULL ,
      month_name CHAR(50) NOT NULL ,
      day_id INTEGER NOT NULL ,
      day_name CHAR(50) NOT NULL ,
      start_time TIMESTAMP NOT NULL ,
      end_time TIMESTAMP NOT NULL )  ;
ALTER TABLE dw.time_dim ADD CONSTRAINT time_pk
      PRIMARY KEY (time_id);

LOAD FROM 'c:\dwbook\dw.TIME_DIM.TXT' OF DEL INSERT INTO
dw.time_dim (time_name, year_id, year_name, sem_id, sem_name,
month_id, month_name, day_id, day_name, start_time, end_time);
-- the above LOAD statement assumes you place the source file in
-- folder c:\dwbook\. (You can adapt this command line as you wish)
```

Resource Dimension

```
CREATE TABLE dw.resource_dim  (
              resource_id INTEGER NOT NULL ,
              resource_name CHAR(50) NOT NULL ,
              category_id INTEGER ,
              category_name CHAR(50) ,
              type_id INTEGER ,
              type_name CHAR(50) ) ;
ALTER TABLE dw.resource_dim ADD CONSTRAINT res_dim_pk PRIMARY KEY
      (resource_id);
-- to populate this table, run the following statement:
INSERT INTO dw.resource_dim
      (resource_id,   resource_name,   category_id,   category_name,
type_id, type_name)
SELECT
      R.resource_id,        R.resource_name,       D.department_id,
D.department_name,
      CASE R.resource_type
            WHEN 'Lecture Room' THEN 1
            WHEN 'Library' THEN 2
            WHEN 'Lab' THEN 3
            END AS type_name, r.resource_type
FROM tx.resources R
INNER JOIN staging.dept_res_mapping M
      ON R.resource_id = m.resource_id
INNER JOIN staging.depts D
      ON D.department_id = m.department_id ;
-- notice the above query could be written without using JOINs
-- but JOINs are part of SQL standard
-- and therefore it is a good idea to always use them
-- merge statement to do UPDATE/INSERT in a single declaration
MERGE INTO dw.resource_dim dest
  USING
(SELECT m.resource_id, d.department_id, d.department_name
FROM staging.dept_res_mapping m
```

```
    INNER JOIN staging.depts d
    ON m.department_id=d.department_id) src
ON (src.resource_id = dest.resource_id)
WHEN MATCHED THEN
   UPDATE SET
      dest.category_id = src.department_id,
      dest.category_name = src.department_name ;
```

Subject Dimension

```
CREATE TABLE dw.subject_dim  (
              subject_id INTEGER NOT NULL ,
              subject_name CHAR(50) NOT NULL ,
              semester_id INTEGER NOT NULL ,
              semester_name CHAR(50) NOT NULL ,
              year_id INTEGER NOT NULL ,
              year_name CHAR(50) NOT NULL )  ;
 ALTER TABLE dw.subject_dim ADD CONSTRAINT sub_dim_pk PRIMARY KEY
      (subject_id);


INSERT INTO dw.subject_dim
SELECT s.subject_id, s.subject_name, m.semester,
      CASE m.semester
            WHEN 1 THEN 'First Sem'
            WHEN 2 THEN 'Second Sem'
            WHEN 3 THEN 'Third Sem'
            WHEN 4 THEN 'Forth Sem'
            WHEN 5 THEN 'Fifth Sem'
            WHEN 6 THEN 'Sixth Sem'
            WHEN 7 THEN 'Seventh Sem'
            WHEN 8 THEN 'Final Sem'
      END AS semester_name,
      CASE m.semester
            WHEN 1 THEN 1
            WHEN 2 THEN 1
            WHEN 3 THEN 2
            WHEN 4 THEN 2
            WHEN 5 THEN 3
            WHEN 6 THEN 3
            WHEN 7 THEN 4
            WHEN 8 THEN 4
      END AS year_id,
      CASE m.semester
```

```
            WHEN 1 THEN 'Year 1'
            WHEN 2 THEN 'Year 1'
            WHEN 3 THEN 'Year 2'
            WHEN 4 THEN 'Year 2'
            WHEN 5 THEN 'Year 3'
            WHEN 6 THEN 'Year 3'
            WHEN 7 THEN 'Year 4'
            WHEN 8 THEN 'Year 4'
      END AS year_name
FROM  staging.subjects s
      INNER JOIN tx.student_master m
            ON s.semester_offered = m.semester
GROUP BY s.subject_id, s.subject_name, m.semester  ;
```

Facilitator Dimension

```
CREATE TABLE dw.facilitator_dim (
             facilitator_id INTEGER NOT NULL ,
             facilitator_name CHAR(50) NOT NULL ,
             department_id INTEGER NOT NULL ,
             department_name CHAR(50) NOT NULL ) ;
 ALTER TABLE dw.facilitator_dim ADD CONSTRAINT fac_dim_pk
      PRIMARY KEY (facilitator_id);


-- to populate this table:
INSERT INTO dw.facilitator_dim
SELECT f.facilitator_id, f.facilitator_name,
f.department_id, s.department_name
FROM tx.facilitator_master f
INNER JOIN staging.depts s
      ON f.department_id = s.department_id ;
```

Attendance Fact Table

```
CREATE TABLE dw.attendance_fact  (
              time_id INTEGER NOT NULL ,
              resource_id INTEGER NOT NULL ,
              subject_id INTEGER NOT NULL ,
              facilitator_id INTEGER NOT NULL ,
              enrollment_id INTEGER NOT NULL,
              enroll_timestamp TIMESTAMP NOT NULL )  ;
ALTER TABLE dw.attendance_fact ADD CONSTRAINT att_fact_pk
      PRIMARY KEY (time_id, resource_id, subject_id,
                  facilitator_id, enrollment_id);
ALTER  TABLE  dw.attendance_fact  ADD  CONSTRAINT  att_fact_time_fk
FOREIGN KEY (time_id)
      REFERENCES dw.time_dim (time_id)
      ON DELETE CASCADE;
ALTER  TABLE  dw.attendance_fact  ADD  CONSTRAINT  att_fact_res_fk
FOREIGN KEY (resource_id)
      REFERENCES dw.resource_dim (resource_id)
      ON DELETE CASCADE;
ALTER  TABLE  dw.attendance_fact  ADD  CONSTRAINT  att_fact_sub_fk
FOREIGN KEY (subject_id)
      REFERENCES dw.subject_dim (subject_id)
      ON DELETE CASCADE;
ALTER  TABLE  dw.attendance_fact  ADD  CONSTRAINT  att_fact_faci_fk
FOREIGN KEY (facilitator_id)
      REFERENCES dw.facilitator_dim (facilitator_id)
      ON DELETE CASCADE;

-- populate fact table using intermediate table
-- create the staging table with the same structure as fact table
-- but allowing NULL values
CREATE TABLE dw.fact_staging (
              time_id INTEGER,
              resource_id INTEGER,
              subject_id INTEGER,
```

```
            facilitator_id INTEGER,
            enrollment_id INTEGER,
            enroll_timestamp TIMESTAMP)  ;

-- populate the staging table
INSERT INTO dw.fact_staging
      (resource_id, subject_id, facilitator_id,
      enrollment_id, enroll_timestamp)
SELECT a.resource_id, m.subject_id, t.facilitator_id,
      a.enrollment_no, a.enroll_timestamp
FROM tx.attendance_record a
      INNER JOIN tx.timetable t
      ON a.resource_id = t.resource_id
      INNER JOIN tx.facilitator_master m
      ON m.department_id = t.department_id
            AND m.facilitator_id = t.facilitator_id
WHERE a.enroll_timestamp BETWEEN t.start_time AND t.end_time
      AND t.facilitator_id in
      (SELECT facilitator_id FROM dw.facilitator_dim) ;
--
MERGE INTO dw.fact_staging dest
USING
(SELECT time_id, start_time, end_time FROM dw.time_dim ) src
ON dest.enroll_timestamp BETWEEN src.start_time AND src.end_time
WHEN MATCHED THEN
     UPDATE set time_id = src.time_id;
--
-- And finally moving to fact table:
INSERT INTO dw.attendance_fact
SELECT time_id, resource_id, subject_id,
      facilitator_id, enrollment_id , enroll_timestamp
FROM dw.fact_staging;
--
-- clean up stage table
DELETE FROM dw.fact_staging;
```

B

Appendix B – Required Software

IBM DB2

It is assumed the reader will study the concepts presented in this book implementing the database objects and all SQL scripts here presented in **DB2 9.7 Express-C**, a free version of DB2 (details and/or downloads: http://www-01.ibm.com/software/data/db2/express/).

For information on how to use DB2 Express-C, visit DB2 University, where the reader can apply to a free online video course and/or download a free ebook about the software (http://www.db2university.com/courses/).

By the way, the statements shown in **Appendix A** will run seamlessly in any DB2 edition, no matter the reader chooses to run it on the mainframe, under a UNIX-like operating system or even under WINDOWS. For more information about other DB2 editions, please check the following link: http://www-01.ibm.com/software/data/db2/.

IBM DATA STUDIO

To run the SQL scripts, the reader might use DB2 Command Line Processor (CLP), which is part of the default DB2 installation. But it is recommended the user download and install **IBM Data Studio 3.1** (http://www-01.ibm.com/software/data/optim/data-studio/), the DB2's free integrated development environment (IDE).

More information about how to use Data Studio can be found at DB2 University, which provides a free online training course and an ebook (http://www.db2university.com/courses/).

IBM Cognos

Cognos is a leading BI software and as it is common in this business, it has no free edition. But you can download a trial version to study the product prior to acquiring it.

In this book, we are using **IBM Cognos 10.1 Developer Edition** as our BI solution. For more information about downloading Cognos, please refer to http://www.ibm.com/developerworks/downloads/im/cognosbi/

Prior to downloading a trial version of IBM Cognos, you need to create a free IBM ID. The reader will be guided on how to do it during the download process.

For more information on how to use the product, please refer to the documentation installed with the software. It includes important information, as you can see in Figure B.1.

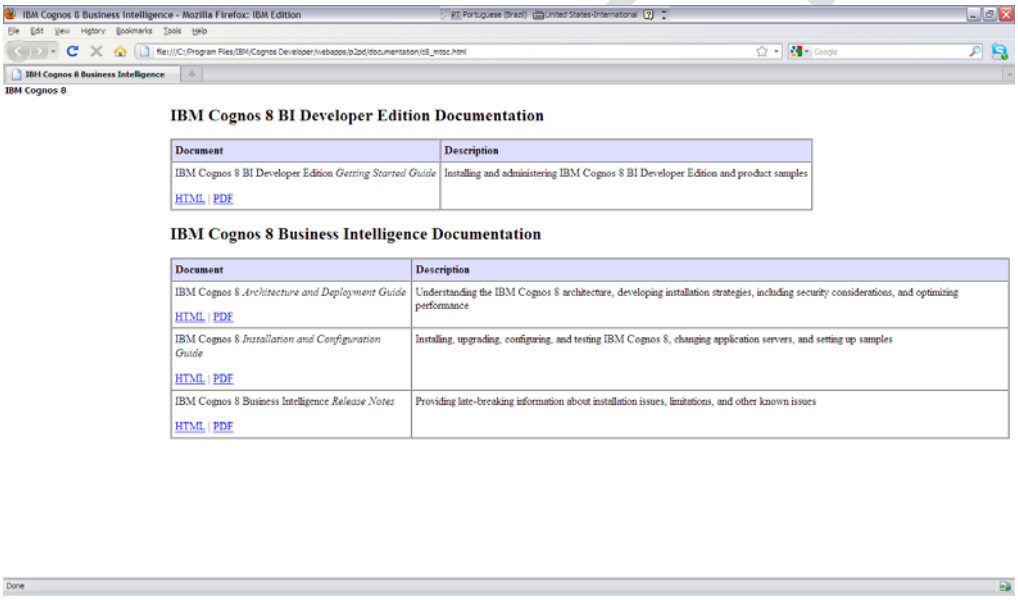


Figure B.1 IBM Cognos Developer Edition Documentation

In case you installed the product in the default path, you will find this document at : file:///C:/Program%20Files/IBM/Cognos%20Developer/webapps/p2pd/documentation/c8_mtoc.html

For any of the above software, in case you find problems with windows using a different language than the one you chose (for instance, you installed software in English, but some windows use a different language), there is a simple “fix”.

All the above software considers the language you chose in your WINDOWS Regional and Language Options (in the Windows Control Panel). You simply need to change the language in the Regional Options tab to solve this issue.

C

Appendix C – References

[1] Leveraging DB2 Data Warehouse Edition for Business Intelligence http://www.redbooks.ibm.com/abstracts/SG247274.html

[2] http://stanford.edu/dept/itss/docs/oracle/10g/olap.101/b10333/multimodel.htm

[3] Database Partitioning, Table Partitioning, and MDC for DB2 9-sg247467.pdf

[4] Physical Database Design, *Kevin Beck, Sam Lightstone*

[5] Seven essential elements to achieve high performance in information integration

http://www-01.ibm.com/software/data/infosphere/datastage/

[6] IBM Infosphere Datastage Data Flow and Job Design

http://www.redbooks.ibm.com/abstracts/sg247576.html?Open

[7] IBM WebSphere QualityStage Methodologies, Standardization, and Matching

http://www.redbooks.ibm.com/abstracts/sg247546.html?Open

[8] E. F. Codd: Relational Completeness of Data Base Sublanguages. In: R. Rustin (ed.): Database Sytems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California : (1972)

[9] E. F. Codd: Recent Investigations in Relational Data Base Systems. ACM Pacific 1975: 15-20

[10] E. F. Codd: Further Normalization of the Data Base Relational Model. IBM Research Report, San Jose, California RJ909: (1971)

[11] E. F. Codd: The Capabilities of Relational Database Management Systems. IBM Research Report, San Jose, California RJ3132: (1981)

[12] E. F. Codd: A Relational Model of Data for Large Shared Data Banks

http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[13] Ralph Kimball: Data Warehouse Toolkit 2ND Edition

[14] Ralph Kimball: Drill Down to Ask Why

[15] Ralph Kimball: Slowly Changing Dimensions

[16] Ralph Kimball: Judge Your BI Tool through Your Dimensions

[17] Ralph Kimball: Exploit Your Fact Tables

[18] Ralph Kimball: The Data Warehouse Lifecycle Toolkit

[19] Ralph Kimball: Essential Steps for the Integrated Enterprise Data Warehouse

[20] Claudia Imhoff: Mastering Data Warehouse Design: Relational and Dimensional Techniques.

[21] Multidimensional models - Constructing DATA CUBE

http://ecet.ecs.ru.acad.bg/cst04/Docs/sV/55.pdf

[22] Conceptual Multidimensional Models

http://www.dia.uniroma3.it/~torlone/pubs/idea03.pdf

[23] James D. Hamilton: Time Series Analysis, Princeton University Press

[24] IBM DB2® 9.7 for Linux, Unix and Windows Information Center

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

[25] DB2 Workload Manager for Linux, UNIX, and Windows

http://www.redbooks.ibm.com/abstracts/SG247524.html

[26] DB2 workload management best practices

http://www.ibm.com/developerworks/data/bestpractices/workloadmanagement/

[27] Using DB2 incremental backup

http://www.ibm.com/developerworks/data/library/techarticle/dm-0910db2incrementalbackup/index.html

[28] IBM Data Archive Expert

http://publib.boulder.ibm.com/infocenter/mptoolic/v1r0/topic/com.ibm.db2tools.aeu.doc.ug/ahxudzicoverviewpage.htm

[29] Optim Integrated data management

http://www-01.ibm.com/software/data/data-management/optim-solutions/

[30] IBM Data Studio

http://www-01.ibm.com/software/data/optim/data-studio/

[31] Getting Started with Framework Manager

http://publib.boulder.ibm.com/infocenter/rentrpt/v1r0m1/index.jsp?topic=%2Fcom.ibm.swg.im.cognos.ug_fm.8.4.1.doc%2Fug_fm_i_FrameworkManager.html

//////////////////

OLAP system with Redundancy

Hence, different from the HADR solution of OLTP systems, the standby node here does not have its own storage at all. The HA solution in OLAP is configured so that in the event of a failover of any primary, its entire storage gets auto-mounted on the standby node.

In order to build such a system, this is the procedure to follow:

1. Since the HA setup uses shared storage, a SAN switch becomes mandatory, which allows all the respective primary file systems to be visible to corresponding standby nodes (in case more than one standby) when an extra SAN zones are created.
2. All the networks that exist amongst the primary nodes (FCM, public etc.) should all be made available to the standby nodes.
3. All the file systems that exist on the primary nodes must be auto-mountable on corresponding standby node.

4. The database instance is made highly available across the primary and standby nodes.

The figure 1.3 illustrates a typical HA (N+1) setup in an OLAP system.

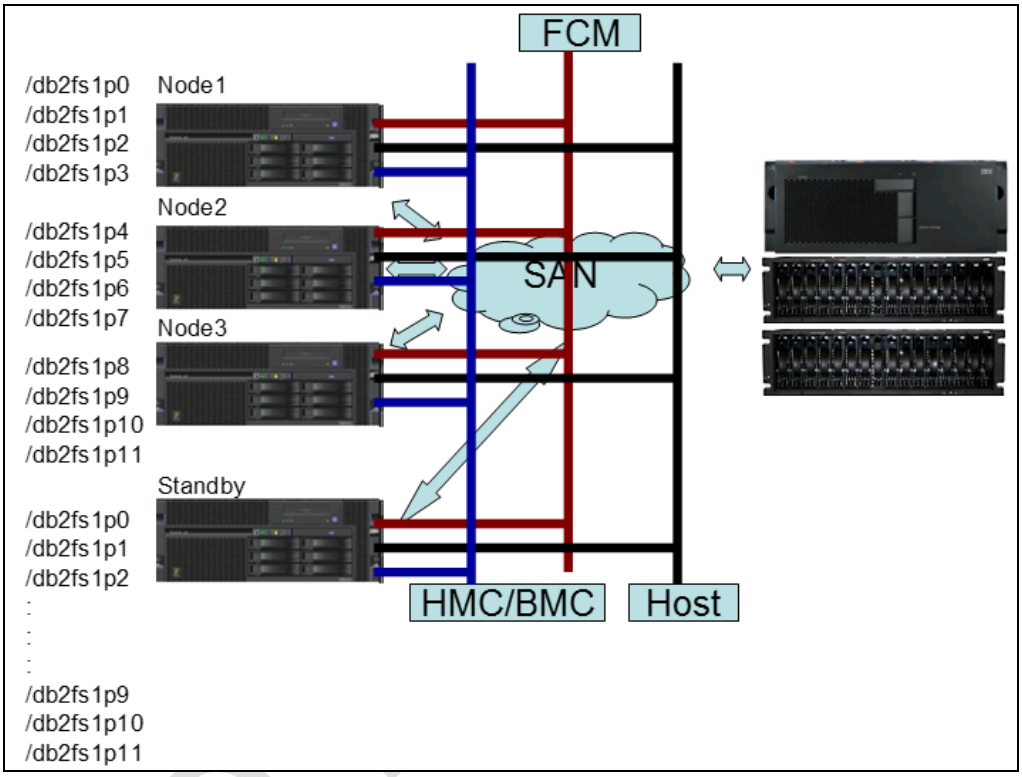


Figure 1.3 An HA (N+1) setup of an OLAP system

As you can see in figure 1.3, each of the primary nodes has four file systems, and all of these are visible and mountable on the standby node through SAN. The standby node can be configured to take over one or more nodes in the event of a failure.