WEB ENABLED DATA WAREHOUSE
BANKING SYSTEM CASE STUDY

By

Mturi Elias
mtrsel@hotmail.com

&

Camilius Sanga
camiliusanga@yahoo.com

Under supervision of
Prof N.L. Mohan
*****

PROJECT REPORT
SUBMITTED TO OSMANIA UNIVERSITY
IN PARTIAL FULLFILMENT
FOR THE AWARD OF DEGREE OF
MASTERS OF SCIENCE
IN COMPUTER SCIENCE

University College of Science
OSMANIA UNIVESRSITY
HYDERABAD – 500 007 INDIA

MAY 2004

University College of Science(A)

Department Of Mathematics
Osmania University

Certificate

This is to certify that the dissertation entitled "**Web Enabled Data Warehouse – Banking System Case Study**" was successfully carried out by **Mr. Mturi Elias** bearing Hall Ticket No. 11-02-646 and **Mr. Camilius Sanga** bearing Hall Ticket No. 11-02-645 in partial fulfillment of the requirements leading to award of Masters Degree in M.Sc (Computer Science) of Osmania University during year 2003-2004.

External Examiner | **Project Supervisor** | **Head of Department**

Prof. D. RAMA MURTHY
Head Dept. of Mathematics
Osmania University
Hyderabad - 500 007 INDIA

University College of Science(A)

Department Of Mathematics
Osmania University

Certificate

This is to certify that the dissertation entitled "**Web Enabled Data Warehouse – Banking System Case Study**" was successfully carried out by **Mr. Mturi Elias** bearing Hall Ticket No. 11-02-646 and **Mr. Camilius Sanga** bearing Hall Ticket No. 11-02-645 in partial fulfillment of the requirements leading to award of Masters Degree in M.Sc (Computer Science) of Osmania University during year 2003-2004.

External Examiner

Project Supervisor

Head of Department

Prof. D. RAMA MURTHY
Head Dept. of Mathematics
Osmania University
Hyderabad - 500 007 INDIA

CENTER FOR EXPLORATION GEOPHYSICS

University College of Science
Osmania University
Hyderabad - 500 007 –India

Prof. N. L Mohan
Head, Dept. Of Geophysics
Osmania University
Hyderabad - 500 007

CERTIFICATE

The project entitled **"WEB ENABLED DATA WAREHOUSE- Banking System Case Study"** submitted in partial fulfillment for the award of degree of Masters of Computer Science is a record bonafied project carried out by Mr. Mturi Elias and Camilius Sanga under my supervision and no part of the report has been submitted for any other degree.

Supervisor

NLM

Prof. N. L Mohan
Project supervisor

DECLARATION

We, Mturi Elias and Camilius Sanga do hereby solemnly declare that the work contained herein has been produced by our personal effort and no way a replica of any material presented as a project either in part or in full in any university for the award of any degree.

Date: MAY 2004 Signature : _____

Mturi Elias

Date: MAY 2004 Signature : _____

Camilius Sanga

Place: University College of Science
OSMANIA UNIVERSITY
HYDERABAD – 500 007 INDIA

ACKNOWLEDGEMENTS

I would like to thank my wife Selestina and my parents Elias and Akechi for their consistent and unflagging support throughout my two years of study.

It also gives me a great pleasure to thank and express my gratitude to my co-workers at University of Dar Es Salaam, Department of Computer Science, Tanzania for their support while I was doing this thesis.

I am indebted to **Prof. N. L Mohan**, Head, Centre for Exploration Geophysics, Osmania University, Hyderabad, for his scholastic guidance and keen interest in this project that could not have been possible without his supervision.

I would like to thank all the interviewees for their time and interest, as well as those who gave some documents from their banks especially Manager of Global Trust Bank Intl and State Bank of Hyderabad.

During my project I also received extra assistance from Dr. L.A. Babu to whom I want to express my great appreciation.

Thank you Mr. Prashanth (System administrator of our department) for doing such a capable job of downloading some software from the Internet

Finally I would like to thank all of those that I have met throughout my study, and that have assisted me in various ways especially Head of departments and other faculties of Mathematics Department, College of Science, Osmania University.

I would first like to express my gratitude to my family and my co-workers at Sokoine University of Agriculture, Directorate of Computer Centre, Tanzania for their support while I was doing this thesis.

I am indebted to **Prof. N. L Mohan**, Head, Centre for Exploration Geophysics, Osmania University, Hyderabad, for his scholastic guidance and keen interest in this project that could not have been possible without his supervision.

I would also like to thank all the interviewees for their time and interest, as well as those who gave some documents from their banks especially Managers of Global Trust Bank International, Sec'Bad Branch and State Bank of Hyderabad, O.U Branch.

During my project I also received extra assistance from Dr. L.A. Babu to whom I want to express my great appreciation.

Thank you Mr. Prashanth (System administrator of our department) for doing such a capable job of downloading some software from the Internet

Finally I would like to thank all of those that I have met throughout my study, and that have assisted me in various ways especially Head of departments and other faculties of Mathematics Department, College of Science, Osmania University.

Abstract

This thesis is an explorative study of the early implementations of the corporate data warehouse that was intended for managers, executives, business analysts, and a few other high-level employees as a tool for analysis and decision-making. Information from the data warehouse was delivered to this group of users in a client/server environment. But today's data warehouses are no longer confined to a select group of internal users. Under present conditions, corporations need to increase the productivity of all the members in the corporation's value chain. Useful information the corporate data warehouse must be provided not only to the employees but also to customers, suppliers and all other business partners.

So today's business climate, you need to open your data warehouse to the entire community of users in the value chain, perhaps also to the general public. This is a tall order. How can you accomplish this requirement to serve information to thousands of users in 24*7 mode? How can you do this without incurring exorbitant costs for information delivery? The Internet along with Web technology is the answer. The web will be your primary information delivery mechanism.

When you bring your data warehouse to the Web, from the point of view of the users, the key requirements are: **self-service data access, interactive analysis, high data quality, high availability and performance, zero-administration client** (thin client technology such as Java applets), **tight security** and **unified metadata.**

WEB ENABLED DATA WAREHOUSE

(BANKING SYSTEM CASE STUDY).

Table of Contents:

Chapter 1: INTRODUCTION.

In early implementations, the corporate data warehouse was intended for managers, executives, business analysts, and a few other high-level employees as a tool for analysis and decision-making. Information from the data warehouse was delivered to this group of users in a client/server environment. But today's data warehouses are no longer confined to a select group of internal users. Under present conditions, corporations need to increase the productivity of all the members in the corporation's value chain. Useful information of the corporate data warehouse must be provided not only to the employees but also to customers, suppliers and all other business partners.

So in today's business climate, you need to open your data warehouse to the entire community of users in the value chain, perhaps also to the general public. This is a tall order. How can you accomplish this requirement to serve information to thousands of users in 24*7 mode? How can you do this without incurring exorbitant costs for information delivery? The Internet along with Web technology is the answer. The web will be your primary information delivery mechanism.

When you bring your data warehouse to the Web, from the point of view of the users, the key requirements are: **self-service data access, interactive analysis, high data quality, high availability and performance, zero-administration client** (thin client technology such as Java applets), **tight security** and **unified metadata.**

Chapter 1: INTRODUCTION.

In early implementations, the corporate data warehouse was intended for managers, executives, business analysts, and a few other high-level employees as a tool for analysis and decision-making. Information from the data warehouse was delivered to this group of users in a client/server environment. But today's data warehouses are no longer confined to a select group of internal users. Under present conditions, corporations need to increase the productivity of all the members in the corporation's value chain. Useful information of the corporate data warehouse must be provided not only to the employees but also to customers, suppliers and all other business partners.

So in today's business climate, you need to open your data warehouse to the entire community of users in the value chain, perhaps also to the general public. This is a tall order. How can you accomplish this requirement to serve information to thousands of users in 24*7 mode? How can you do this without incurring exorbitant costs for information delivery? The Internet along with Web technology is the answer. The web will be your primary information delivery mechanism.

When you bring your data warehouse to the Web, from the point of view of the users, the key requirements are: **self-service data access, interactive analysis, high data quality, high availability and performance, zero-administration client** (thin client technology such as Java applets), **tight security** and **unified metadata**.

In the context of e-commerce, analyzing the behavior of customer, a product, or a company consists in monitoring one or several activities (commercial or medical pursuits, patent deposits, etc.). The objective of multidimensional analysis, particularly OLAP, is to analyze such activities under the form of numerical data. The information is summarized and is presented as relevant information (i.e.. knowledge), this connecting OLAP to other analysis tools such as KDD (Knowledge Discovery in Databases) techniques (namely, data mining) whose objectives are to understand and predict the behavior of one or several activities.

To be efficient in terms of quality and response time, analysis tools need their input data to be properly structured, acquired, and prepared in a previous step. These data are typically stored in databases aimed at decision support (such as data warehouses) that we call Decision Databases (DDBs). These databases can necessitate external data sources. For instance, a company willing to support competitive monitoring cannot merely analyze only data from its own production databases. The Web is a prevalent data delivery and data source in this context. However, the data broadcasted on this medium are very heterogeneous, which makes their conceptualization in a data-warehousing framework difficult. Nonetheless, the concepts of data warehousing remain valid in this approach. Measures, though not necessarily numerical, remain the indicators for analysis, and analysis is still performed following different perspectives represented by dimensions. Large data volumes and their dating are other arguments in favor of this approach.

The Objectives of the study include:

- To use the Web as our primary information delivery mechanism for Data Warehouse.
- To integrate the OLAP technology and Data Mining techniques to analyze data in an application manner.
- To make clearly why data quality is critical in a data warehouse and how to improve the quality of data in the data warehouse.
- How to improve the performance of the Data warehouse.

Organization

Chapter 1 introduces the scene and problem statement/specification of the project.

Chapter 2 introduces the Enterprise Architecture and Data Warehouse concepts, the basis of the reasons for writing this report.

Chapter 3 of this report focuses on the analysis phase of SBH Bank Web Enabled Data Warehouse. This section is devoted to addressing the primary requirements identified as a result of the interview conducted in the SBH bank.

Chapter 4 presents data modeling of the SBH bank for data warehousing. In data warehousing project, the logical data model is built based on user requirements, and then it is translated into the physical data model.

Chapter 5 of this report focuses on the data cubes and OLAP aspect of data warehousing.

It is common for the data warehouse to be on the bottom of the nightly batch load, and after the loading of the data warehouse, there usually isn't much time remaining for the OLAP cube to be refreshed. As a result, it is worthwhile to experiment with the OLAP cube generation paths to ensure optimal performance.

Chapter 6 of this report opens a window to the main objective of this project (Web-Enabled Data Warehouse) and the future of the data warehouse. This section is devoted to addressing why and how the Data Warehouse can be adapted to the Web.

Chapter 7 reinforces the importance of data quality in a data warehouse. It describes why data quality is critical in a data warehouse, observe the challenges posed by corrupt data and address the methods to deal with and appreciate the benefit of quality data.

Chapter 8 provides a detailed description of the techniques, which improves the performance in a data warehouse.

Chapter 9 is a step-by-step guide for developing and implementing ETL process, OLAP and the Front-end application for our Data Warehouse.

Chapter 10 is the Conclusion, Evaluation and Further Work. It provides the details of what we have achieved, give a critical appraisal (evaluation) of our work - how could the work be taken further (perhaps by another student next year)?

Chapter 2. BACKGROUND

In retrospect, it is easy to see how computing has shifted its focus from operational to decisional concerns. The differences in operational and decisional information requirements presented new challenges that old computing practices could not meet. Below, we elaborate on how this change in computing focus became the impetus for the development of data warehousing technologies.

The Business Cycle shows us that any enterprise must operate at three levels: operational (i.e., the day-to-day running of the business); tactical (i.e., the definition of policy and the monitoring of operations); and strategic (i.e., the definition of organization's vision, goals and objectives).

Much of the effort and money in computing has been focused on meeting the *operational* business requirements of enterprises. After all, without the OLTP applications that record thousands, even millions, of discrete transactions each day, it would not be possible for any enterprise to meet customer needs while enforcing business policies consistently. Nor would it be possible for an enterprise to grow without significantly expanding its manpower base.

The Data Warehouse Defined

What is a data warehouse? Data warehouses are a special type of database that are built for the specific purpose of *getting information out* rather than *putting data in,* which is the purpose of most application databases.

William H. Inmon in *Building the Data Warehouse* defines a data warehouse as "a collection of integrated, subject-oriented, time-variant and nonvolatile databases designed to supply the information required for decision-making."

Subject-Oriented: Organized around major subjects, such as customer, product, sales

Integrated: Constructed by integrating multiple, heterogeneous data sources, such as relational databases, flat files, on-line transaction records.

Time Variant: The data warehouse contains slice of data across different periods of time.

Non-Volatile: Data in the data warehouse is ready-only.

A more thorough look at the above definition yields the following observations

The Dynamic, Ad Hoc Report

The most ideal scenario for enterprise decision-makers (and for IT professionals) is to have a repository of data and a set of tools that will allow decision-makers to create their own set of dynamic reports. The term *dynamic report* refers to a report that can be quickly modified by its user to present either greater or less detail, without any additional programming required. Dynamic reports are the only kind of reports that provide true, ad-hoc reporting capabilities.

Data Warehouse Architecture

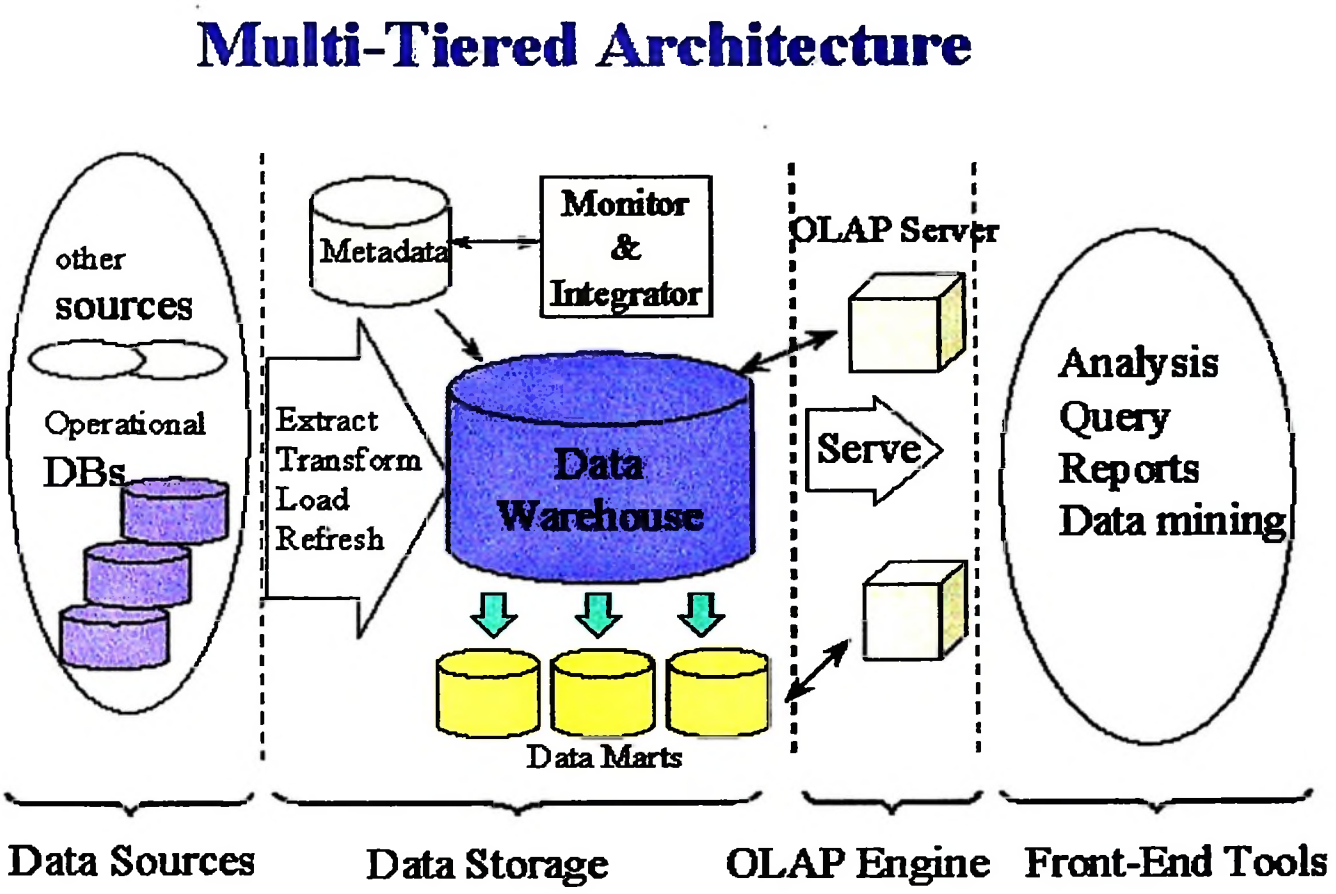


Fig.2.1

Extraction, Transformation and Loading (ETL)

In order to load the data warehouse regularly, data from one or more operational systems needs to be extracted, transformed and loaded into the data warehouse.

❖**Extraction:**

During extraction, the desired data has to be identified and extracted from many different sources, including databases systems and application.

❖**Transformation:** the process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data. It involves:

- Cleaning data
- Eliminating inconsistencies
- Adding elements
- Merging data
- Integrating data

❖**Loading:** moves the data into the warehouse.

- Transportation – involves moving the data from source data stores or an intermediate staging area to the data warehouse.
- Loading – loads the data into the target warehouse database in the target system server.

A Word About Data Marts

A discussion of data warehouses is not complete without a note on data marts. Unlike data warehouses, which contain large quantities of data from key operational systems in an enterprise, a data mart typically contains only a subset of the data that would have been stored in an enterprise data warehouse. Data mart data are selected to meet the specific needs of a subset of the organization. It is not unusual to find a data mart developed and implemented for a department, a division, or a geographical location.

Data marts are often preferred by enterprises as a first step to building a data warehouse, since these can be used as a "proof of concept." Initial success with the data mart can be used to convince skeptics in the enterprise and loosen the enterprise's purse strings

OLAP

OLAP is an acronym for **On Line Analytical Processing**, A category of software technology that enables analysts, mangers and executives to perform ad hoc data access and analysis based on its dimensionality. This form of multidimensional analysis provides business insight through fast, consistent, interactive access to a wide variety of possible views of information.

OLAP is becoming the fundamental foundation for Intelligent Solutions including Business Performance Management, Planning, Budgeting, Forecasting, Financial Reporting, Analysis, Simulation Models, Knowledge Discovery, and Data Warehouse Reporting.

A Web-Enabled Data warehouse

A Web-enabled data warehouse uses the Web for information delivery and collaboration among users. The figure below indicates an architectural configuration for a Web-enabled data warehouse. The convergence of the Web and data warehousing is of supreme importance to every corporation doing business in the 21st century.

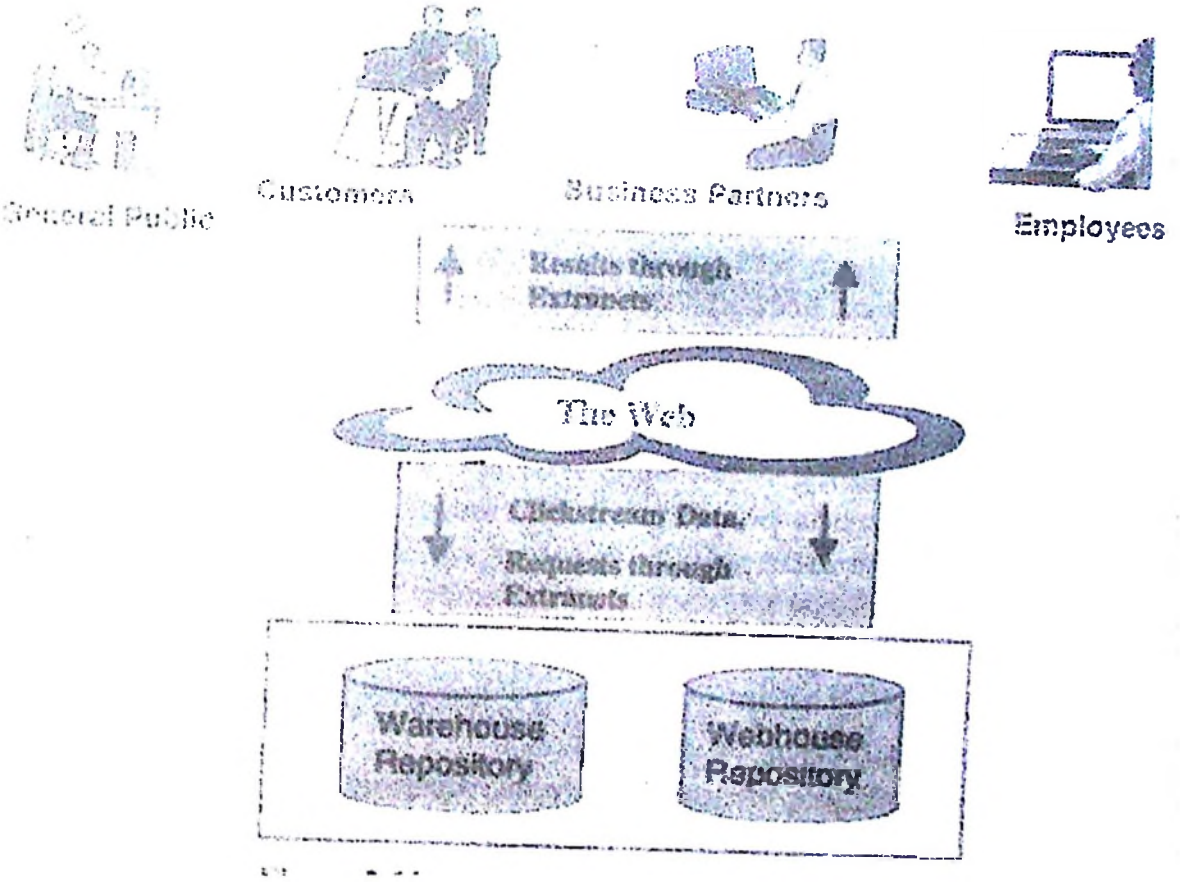


Fig.2.2 Simplified Web Enabled Configuration

Data Mining

Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories. ***We are drowning in data, but starving for knowledge.*** **The solution is *data mining.***

An information extraction activity whose goal is to discover hidden facts contained in databases. Using a combination of machine learning, statistical analysis, modeling techniques and database technology, data mining finds patterns and subtle relationships in data and infers rules that allow the prediction of future results. Typical applications include market segmentation, customer profiling, fraud detection, evaluation of retail promotions, and credit risk analysis.

The Purposes of a Data warehouse

At this point, it is helpful to summarize the typical reasons enterprises undertake data warehousing initiatives.

To Provide Business Users with Access to Data

The data warehouse provides access to integrated enterprise data previously locked away in unfriendly, difficult-to-access environments. Business users can now establish, with minimal effort, a secure connection to the warehouse through their desktop Personal Computer. Security is enforced either by the warehouse front-end application, by the server database, or both.

Because of its integrated nature, a data warehouse spares business users from the need to learn, understand, or access operational data in their native environments and data structures.

Data Warehouse Cost-Benefit Analysis/Return on Investment

Senior management typically requires a cost-benefit analysis (CBA) or a study of return on investment (ROI) prior to embarking on a data warehousing initiative. Although the task of calculating ROI for data warehousing initiatives is unique to each enterprise, it is possible to classify the type of benefits and costs that are associated with data warehousing.

Benefits:

Data warehousing benefits can be expected from the following areas:

- **Redeployment of staff assigned to old decisional systems.** The cost of producing today's management reports is typically undocumented and unknown within an enterprise. The quantification of such costs in terms of staff hours and erroneous data may yield surprising results. Benefits of this nature, however, are typically minimal, since warehouse maintenance and enhancements require staff as well. At best, staff will be redeployed to more productive tasks.
- **Improved productivity of analytical staff due to availability of data.** Analysts go through several steps in their day-to-day work: locating data, retrieving data, analyzing data to yield information, presenting information, and recommending a course of action. Unfortunately, much of the time (sometimes up to 40 percent) spent by enterprise analysts on a typical day is devoted to locating and retrieving data. The availability of integrated, readily accessible data (in the data warehouse) should significantly reduce the time that analysts spend with data collection tasks and increase the time they have available to actually analyze the data they have collected. This leads either to shorter decision cycle times or improvements in the quality of the analysis.
- **Business improvements resulting from analysis of warehouse data.** The most significant business improvements in warehousing result from the analysis of warehouse data, especially if the easy availability of information yields insights heretofore unknown to the enterprise. The goal of the data warehouse is to meet

decisional information needs; it therefore follows naturally that the greatest benefits of warehousing are obtained when decisional information needs are actually met and sound business decisions are made both at the tactical and strategic level. Understandably, such benefits are the most significant and, therefore, the most difficult to project and the most difficult to quantify.

Chapter 3. ANALYSIS

The State Bank of Hyderabad offers a significant portfolio of services, which are

- Checking accounts. (Current account)
- Savings accounts.
- Mortgage loans.
- Investment loans.
- Personal loans.
- Credit card
- Safe deposit boxes.

Our goal is to build a household data warehouse where we can track all the accounts owned by the bank, and see all the individual account holders, as well as the residential and commercial household groupings to which they belong. A major goal of the bank is to market more effectively to its households, and offer additional services to households that already have one or more accounts.

As result of conducting user interviews with managers and analysts around the State Bank of Hyderabad (SBH), we have developed the following set of special requirements:

Functional Requirements

i. We want to see five years of historical data on every account. For all prior months it will be sufficient to see the end-of-month snapshot.

ii. For the current month, we want a valid snapshot as of yesterday. We don't need the other prior days in the current month.

iii. Every type of accounts has a primary balance. There is a significant need to group different kinds of accounts in the same analyses and compare primary balance.

iv. Every type of account has a list of custom dimension attributes and custom numeric facts that tend to be quite different from account type to account type.

v. Every account is deemed to belong to a household. Upon studying the historical production data, we conclude that accounts and the individual who own the accounts come and go from households as much as several times per year for each household.

vi. Since some of accounts were created many years ago, and by different production systems, we find that our records of the individual account holders' names and addresses differ from account in many cases.

vii. In addition to the household identification, we are very interested in demographic information as it pertains to both the individual accounts account holders and the households. We also capture and store behavior scores relating to the activity in each of the accounts.

Information Delivery Requirements

i. Queries

- **Query Definition**. Make it easy to translate the business need into the proper query.
- **Query Recasting**. Even simple-looking queries can result in intensive data retrieval and manipulation. Therefore, provide for parsing incoming queries and recasting them to work more efficiently.
- **Ease of Navigation.** Use of metadata to browse through the data warehouse, easily navigating with business terminology and not technical phrases.
- **Query Execution.** Provide ability for the user to submit the query for execution without any intervention.
- **Results Presentation.** Present results of the query in a variety of ways.
- **Aggregate Awareness.** Query processing mechanisms must be aware of aggregate fact tables and, whenever necessary, redirect the queries to the aggregate tables for faster retrieval.
- **Query Governance.** Monitor and intercept runaway queries before they bring down the data warehouse operations.

downtime in the set-up due to a security breach or application failure could result in serious business implications.

Security Requirements

The main security requirements for State Bank of Hyderabad are Authentication of business partners and customers, data integrity, confidentiality, continuous availability of data and non-repudiation of transactions.

The following assets of the State Bank of Hyderabad that must be monitored because they have potential threat:

- Web Complex – Router, Web servers, Application servers, Database servers, Exchange servers and other systems.
- Web Data – Data stored in the various database servers within the DMZ.
- Data Transmission – Data that is in transited between the client and the bank institution, and vice versa.
- Business Transactions – Transactions that can be carried out through the application(s).
- Corporate Resources – Back-end supporting systems and internal users, News, other enterprise systems and applications. Also, the application should be able to communicate securely with several systems in real-time within the De-Militarized Zone (DMZ), corporate network and third party network. This emphasizes the need for transactional integrity across system boundaries. Security for this should not impede application performance, availability or accessibility. The applications are mission critical which requires a 24*7 uptime and any

ii. Reports

- **Set of preformatted reports.** Provide a library of preformatted reports with clear descriptions of the reports. Make it easy for users to browse through the library and select the reports they need.
- **Parameter-driven predefined report.** These give the users more flexibility than the preformatted ones. Users must have the capability to set their own parameters and ask for page breaks and subtotals.
- **Easy-to-use report development.** When users need new reports in addition to preformatted or predefined reports, they must be able to develop their own reports easily with a simple report-writer facility.

Performance Requirements

i. The system must be available 24 hours a day, seven days a week, and that if a user or administrator is accessing a given account, they will have exclusive privileges to it until their transaction has been completed.

ii. In order to be fully interactive, the system will have to be able to handle queries and display a message on the screen within a maximum of 15 seconds. Otherwise, the system will not be very realistic or usable.

Information Delivery System

In all of our deliberations up to now, we have come to realize that there are four underlying methods for information delivery. We may be catering to the need of any class of users. We may be constructing the information delivery system to satisfy the requirements of users with simple needs or those of power users. Still the principal means of delivery are the same. These includes:

- Queries
- Reports
- Analysis
- Applications

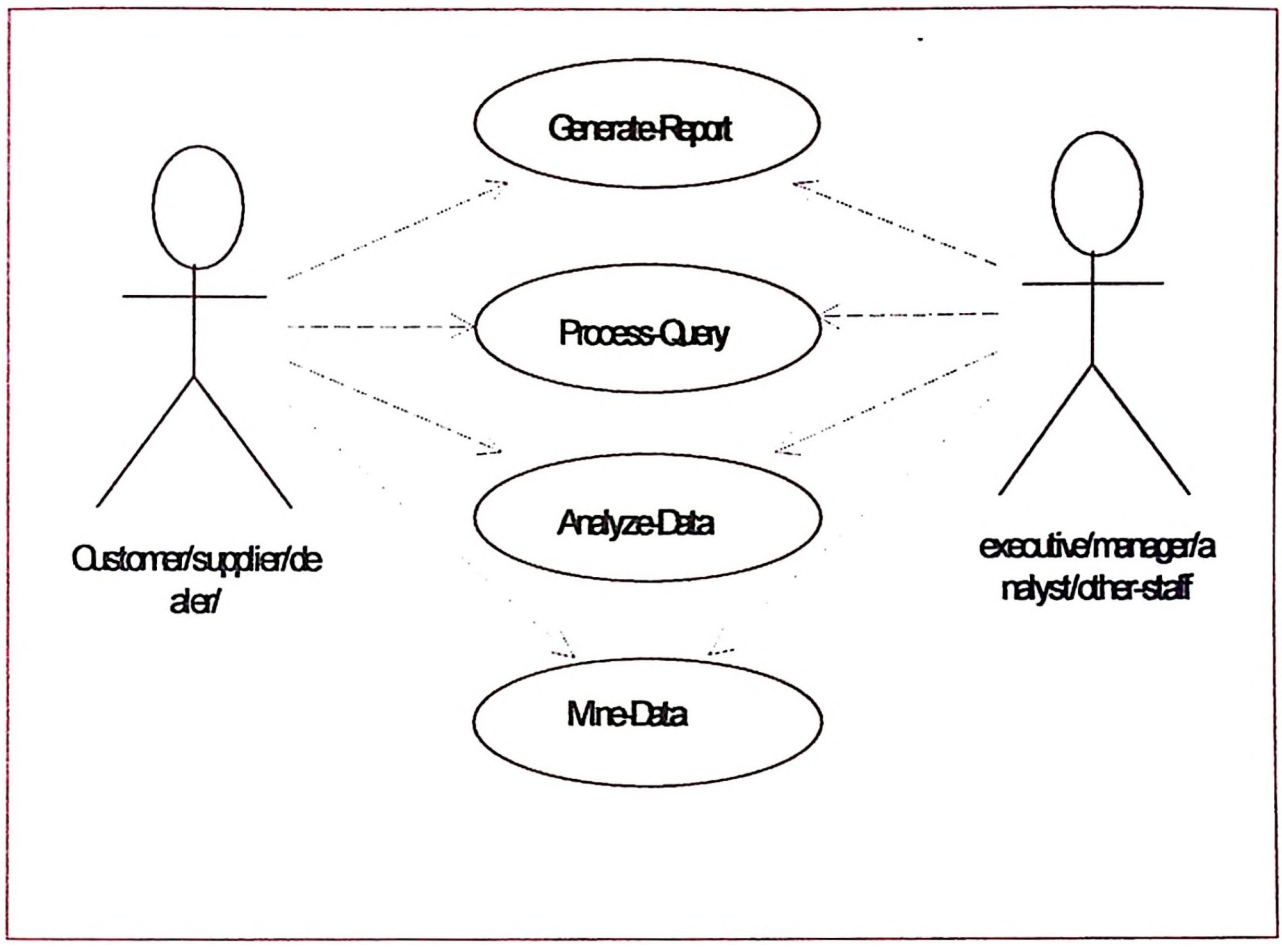


Fig.3.1

1. Queries

In a data warehouse, query processing is the most common method for information delivery.

Features of a managed query environment:

- Query initiation, formulation, and results presentation are provided on the client machine.
- Metadata guides the query process.

- Ability for the users to navigate easily through the data structures is absolutely essential.
- Information is pulled by the users, not pushed to them.
- Query environment must be flexible to accommodate different classes of users.

2. Reports

The method of information delivery through reports is carry-over from operational systems.

Features of the reporting environment:

- The information is pushed to the user, not pulled by the user as in the case of queries. Reports are published and the user subscribes to what he or she needs.
- Compared to queries, reports are flexible and predefined.
- Most of the reports are preformatted and therefore, rigid.
- The user has less control over the reports received than the queries he or she can formulate.
- A proper distribution system must be established.
- Report production normally happens on the server machine.

3. Analysis

Users interested in analysis include Business strategists, market researchers, planners and analysts. Because of its rich historical data content, the data warehouse is very well suited for analysis. It provides these users the means to search for trends and find correlations and discern patterns. In one sense, analysis session is nothing but a session of a series of

related queries. Analysis can become extremely complex, depending on what the explorer is after. Complex analysis falls in the domain of online analytical processing (OLAP).

4. Applications

A decision support application in relation to the data warehouse is any downstream system that gets data feed from the data warehouse. In addition to letting the users access the data content of the warehouse directly, specialized application for specific group of users is created. A more recent development is data mining, a major type of application that gets data from the data warehouse.

Data mining deals with knowledge discovery.

Chapter 4. DATA MODELING.

This is a very important step in the data warehousing project. Indeed, it is fair to say that the foundation of the data warehousing system is the data model. A good data model will allow the data warehousing system to grow easily, as well as allowing for good performance.

In data warehousing project, the logical data model is built based on user requirements, and then it is translated into the physical data model. There are three levels of data modeling. They are conceptual, logical, and physical. This section will explain the difference among the three, the order with which each one is created, and how to go from one level to the other.

Conceptual Data Model

Features of conceptual data model include:

- Includes the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified.

Logical Data Model

Features of logical data model include:

- Includes all entities and relationships among them.
- All attributes for each entity are specified.

- The primary key for each entity specified.
- Foreign keys (keys identifying the relationship between different entities) are specified.
- Normalization occurs at this level.

At this level, the data modeler attempts to describe the data in as much detail as possible, without regard to how they will be physically implemented in the database.

In data warehousing, it is common for the conceptual data model and the logical data model to be combined into a single step (deliverable).

The steps for designing the logical data model are as follows:

1. Identify all entities.
2. Specify primary keys for all entities.
3. Find the relationships between different entities.
4. Find all attributes for each entity.
5. Resolve many-to-many relationships.
6. Normalization.

Physical Data Model

Features of physical data model include:

- Specification all tables and columns.
- Foreign keys are used to identify relationships between tables.
- Denormalization may occur based on user requirements.

- Physical considerations may cause the physical data model to be quite different from the logical data model.

At this level, the data modeler will specify how the logical data model will be realized in the database schema.

The steps for physical data model design are as follows:

1. Convert entities into tables.
2. Convert relationships into foreign keys.
3. Convert attributes into columns.
4. Modify the physical data model based on physical constraints / requirements.

4.1 Detail Operation Description

i. Data Requirements.

- The bank is organized into branches.
- Bank customers are identified by their customer_id values.
- Bank employees are identified by their employee_id values.
- The bank offers two types of accounts.
- A loan originates at a particular branch and can be held by one or more customers. A unique loan number identifies a loan.

  In real banking enterprise, the bank keeps track of deposits and withdrawals from current and current account, just as it keeps track of payments to loan accounts.

ii. Entity Sets Designation

Identify sets and their attributes.

- The branch entity sets with attributes branch-name, branch-city and assets.
- The customer entity set with attributes customer-id, customer-name, customer-street, customer-city and banker-name.
- The employee entity set with attributes employee-id, employee-name, telephone-no, salary and manager.
- Two account entity sets: - savings-account and checking account / current with the common attributes of account-no and balance; in addition, savings-account has the attribute interest-rate and current-account has the attribute overdraft-amount.

- The loan entity set with the attributes loan-number, amount and originating-branch.
- The loan-payment with attributes payment-no, payment-date and payment-amount.

iii. Relationship Sets Designation

Specify sets relationship and mapping.

- Borrower, a many-to-many relationship set between customer and loan.
- Loan-branch, a many-to-one relationship set that indicates in which branch a loan originated.
- Loan-payment, a one-to-many relationship attribute from loan to payment, which documents that a payment is made on a loan.
- Depositor, with relationship attribute access-date, many-to-many relationship set between customer and account, indicating that a customer owns an account.
- Cust-banker, with relationship attribute type, many-to-one relationship set expressing that a customer can be advised by a bank employee and that a bank employee can advise one or more customers.
- Works-for, a relationship set between employee entities with role indicators manager and worker; the mapping cardinalities express that an employee works for only one manager and that a manager supervises one or more employees

Detail Class Diagram

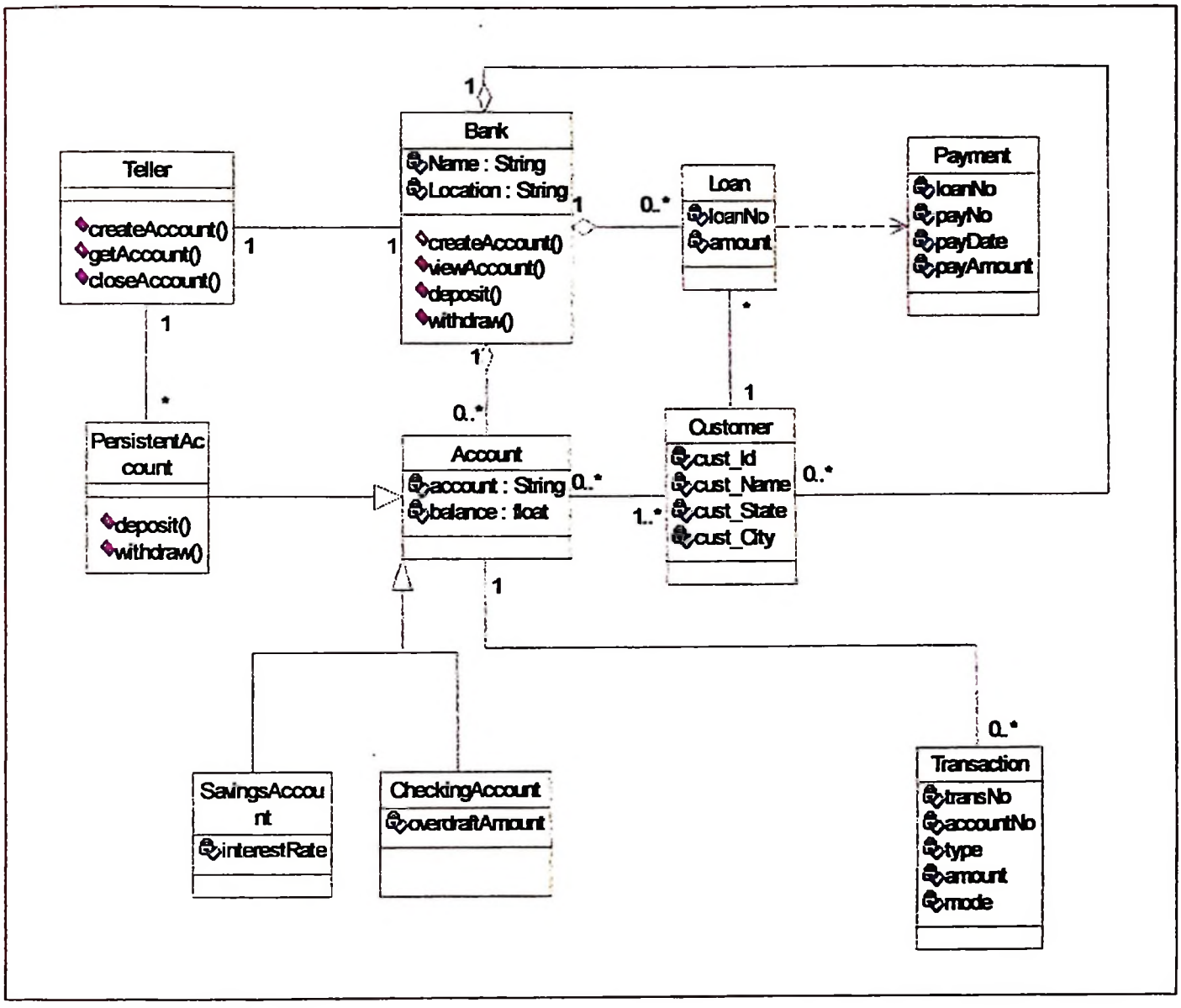


Fig.4.1

4.2 Dimensional Data Modeling

Dimensional data model is most often used in data warehousing systems. This is different from the 3rd normal form, commonly used for transactional (OLTP) type systems. As you can imagine, the same data would then be stored differently in a dimensional model than in a 3rd normal form model.

To understand dimensional data modeling, let's define some of the terms commonly used in this type of modeling:

Dimension: A category of information. For example, the time dimension.

Attribute: A unique level within a dimension. For example, Month is an attribute in the Time Dimension.

Hierarchy: The specification of levels that represents relationship between different attributes within a hierarchy. For example, one possible hierarchy in the Time dimension is Year --> Quarter --> Month --> Day.

Fact Table: A fact table is a table that contains the measures of interest. For example, sales amount would be such a measure. This measure is stored in the fact table with the appropriate granularity. For example, it can be sales amount by store by day. In this case, the fact table would contain three columns: A date column, a store column, and a sales amount column.

Lookup Table: The lookup table provides the detailed information about the attributes. For example, the lookup table for the Quarter attribute would include a list of all of the quarters available in the data warehouse. Each row (each quarter) may have several fields, one for the unique ID that identifies the quarter, and one or more additional fields that specifies how that particular quarter is represented on a report (for example, first quarter of 2001 may be represented as "Q1 2001" or "2001 Q1").

A dimensional model includes fact tables and lookup tables. Fact tables connect to one or more lookup tables, but fact tables do not have direct relationships to one another. Dimensions and hierarchies are represented by lookup tables. Attributes are the non-key columns in the lookup tables.

In designing data models for data warehouses / data marts, the most commonly used schema types are **Star Schema** and **Snowflake Schema**.

Star Schema: In the star schema design, a single object (the fact table) sits in the middle and is radially connected to other surrounding objects (dimension lookup tables) like a star. A star schema can be simple or complex. A simple star consists of one fact table; a complex star can have more than one fact table.

Snowflake Schema: The snowflake schema is an extension of the star schema, where each point of the star explodes into more points. The main advantage of the snowflake schema is the improvement in query performance due to minimized disk storage requirements and joining smaller lookup tables. The main disadvantage of the snowflake schema is the additional maintenance efforts needed due to the increase number of lookup tables.

Steps in the dimensional data modeling process.

i. **Choose a business process to model.**

A business process is a major operational process in our banking system that is supported by some kind of legacy system (or systems) from which data can be collected for the purposes of the data warehouse. In our case, we have the following business processes:

- Account administration. And
- Loan Processing.

ii. **Choose the grain of the business process.**

The grain is the fundamental atomic level of data to be represented in the fact table for the identified processes.

In our case, the grain for account administration is account by month while for loaning is loan by month.

iii. **Choose the dimension that will apply to each fact table record.** In our case study, we have the following dimensions:

- Account.
- Customer.
- Branch.
- Loan.
- Status.
- Time.
- Payment.

One of the design principles is that:

The dimension tables must not be normalized but should remain as flat tables. Normalized dimension tables destroy the ability to browse. Disk space savings gained by normalizing the dimension tables are typically less than one percent of the total disk space needed for the overall schema.

iv. **Choose the measured facts that will populate each fact table record**. Typical measured facts are numeric additive quantities.

In our case study, in account core fact table we have the following measured facts:

- Transaction count
- Balance.
- Amount.

- Fees charged.
- Payment amount.

While, in **savings account custom fact table** we have:

- Deposits.
- Withdraws.
- Interest earned.
- Balance.
- Service charges.

While, in **checking account custom fact table** we have:

- ATM transaction.
- Drive-up Transaction.
- Walk-in Transaction.
- Deposits.
- Checks paid.
- Overdraft.

Another design principle is that:

The fact table in a dimensional schema is naturally highly normalized.

We have used **Star Schema** Model in dimensional modeling (DM) for our data warehouse and data marts due to the following reasons:

1. It is easy to understand by the users because the structure is so simple and straightforward.

2. Provides fast response to queries with optimization and reductions in the physical number of joins required between fact tables and dimension tables.
   (Most suitable for query-centric banking environments).
3. Contains simple metadata.
4. Many front-end tools support it.

Also we have extended the design to deal with the major issue of heterogeneous products. A central problem of our Web enabled banking data warehouse is the highly varied nature of the financial service products.

- In data warehouse where a dimension must describe a large number of heterogeneous items, the recommended technique is to create a core fact and create a custom fact table for querying each individual type in depth.

In account administration process and loan processing, we have

Account Core Fact table,

Savings Account Custom fact table and

Checking account Custom Fact table.

- Multiple fact tables are needed when a business has heterogeneous products that have naturally different facts but a single customer.

4.2.1.Data Mart Matrix.

Data Mart Matrix Showing the Data Warehouse Bus Architecture

The data mart matrix shows the relationship between the possible data marts and dimensions. Any dimension (column) with more than one X implies that this dimension must be conformed across multiple data marts in order to fit into the Data Warehouse Bus Architecture. A brief description of each data mart and dimension follows the matrix. ** Indicates data marts that are detailed in the Dimensional Model Document.

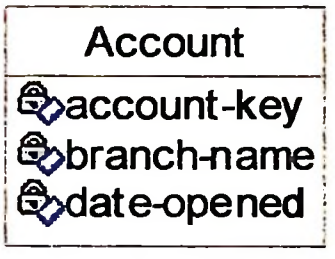
Data Mart	Status	Customer	Time	Branch	Loan	Account	Payment
AccountAdmin	x		x	x	x	x	x
LoanProc	x	x	x	x		x	x

Fig.4.2

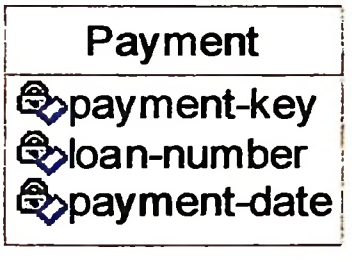
4.2.2. Logical Data Model.

Dimensions:

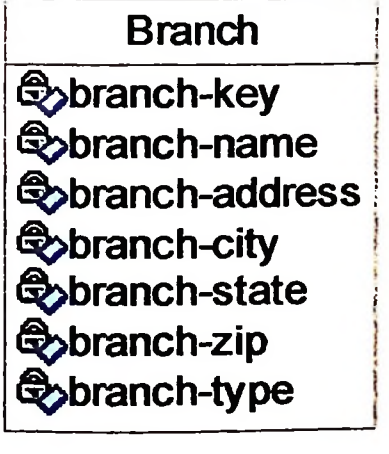
Account Dimension.



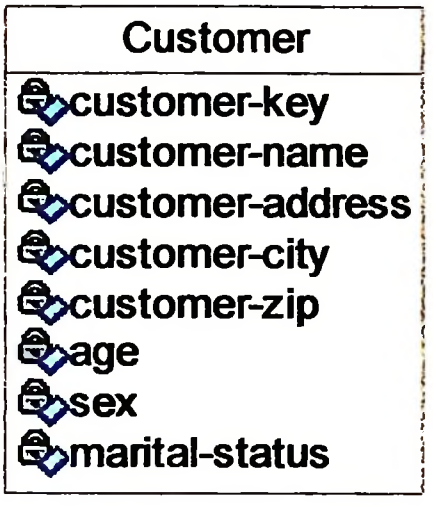
Payment Dimension.



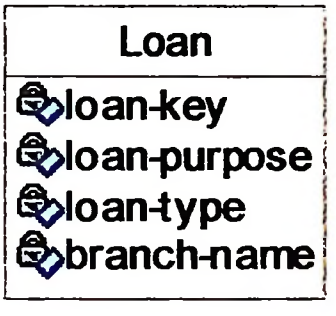
Branch Dimension.



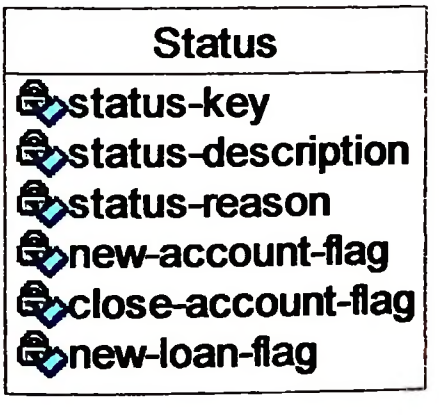
Customer Dimension.



Loan Dimension.



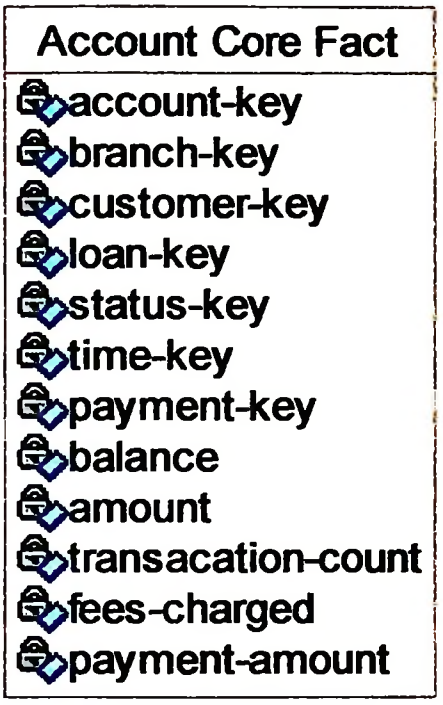
Status Dimension.



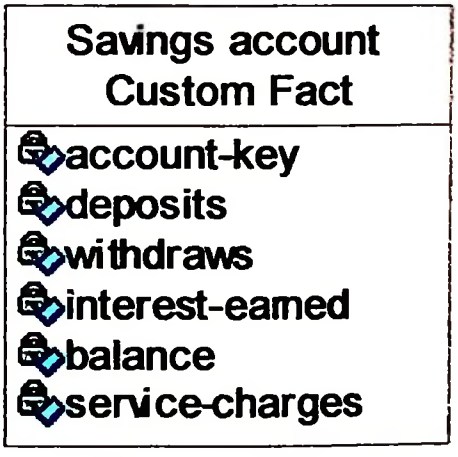
Time Dimension.

Facts Tables.

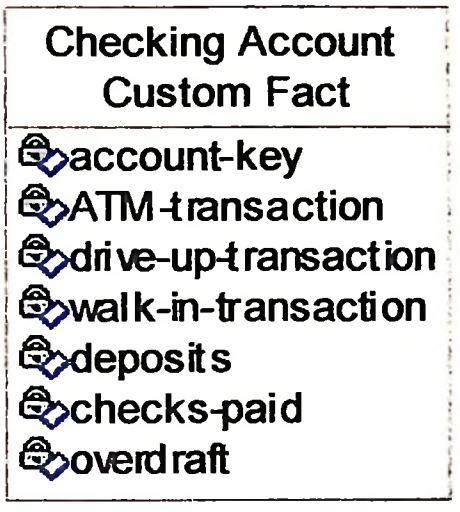
Account Core Fact Table.



Savings Account Custom Fact Table.



Checking account Custom Fact Table.



Star Schema.

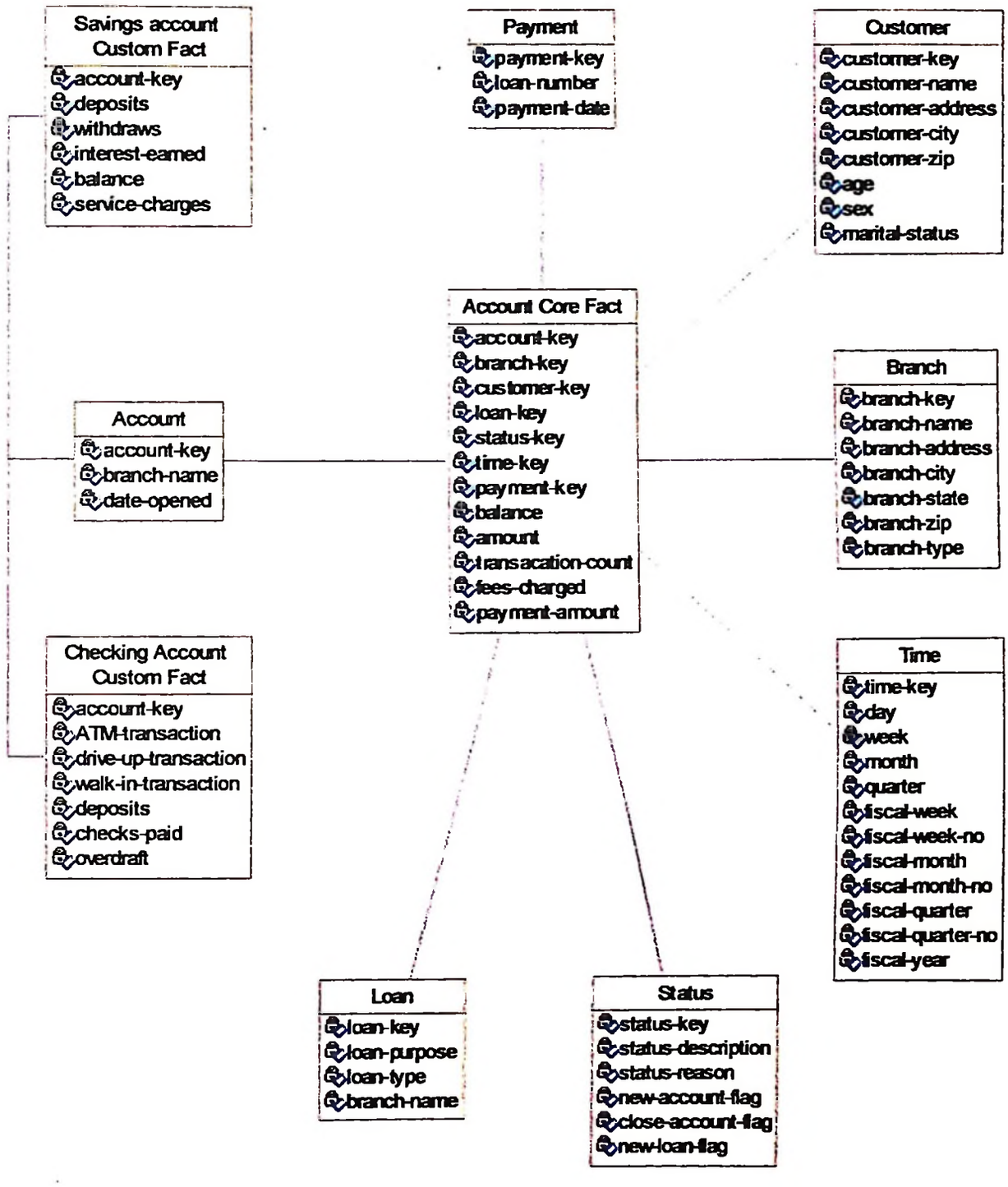


Fig.4.3

4.2.3. Physical Database design.

ACCOUNT ADMINISTRATION AND LOAN PROCESSING:

Dimension Descriptions.

Dimension Name	Dimension Description
Time	Contains attributes about the time when an activity occurred
Branch	The business group within the company and other organizational relationships.
Customer	Describe individuals who are members of the same account.
Account	A statement of money paid or owed with a bank.
Status	Describes the condition of the account.
Payment	Amount to be paid depend on the loan taken.
Loan	A sum of money being borrowed by a customer.

Dimension details.

Customer Dimension.

Attribute Name	Attribute Description
Customer-name	The name of the customer
Customer-address	The street address for this customer location. Note: the address number, suite/box, street name will be split apart for complete flexibility.
Customer-city	The name of the city where this customer location is.
Customer-state	The name of the state where this customer location is.
Customer-zip	The digit zip code for this customer location.
age	The age of the customer.
sex	Define sex either male or female.
Marital-status	The marital status of the customer.

Data Mart Name	Data Mart description
AccountAdmin	Describes all the accounts owned by the bank and individual account holders.
LoanProc	Describe all the loans borrowed by customers from the bank.

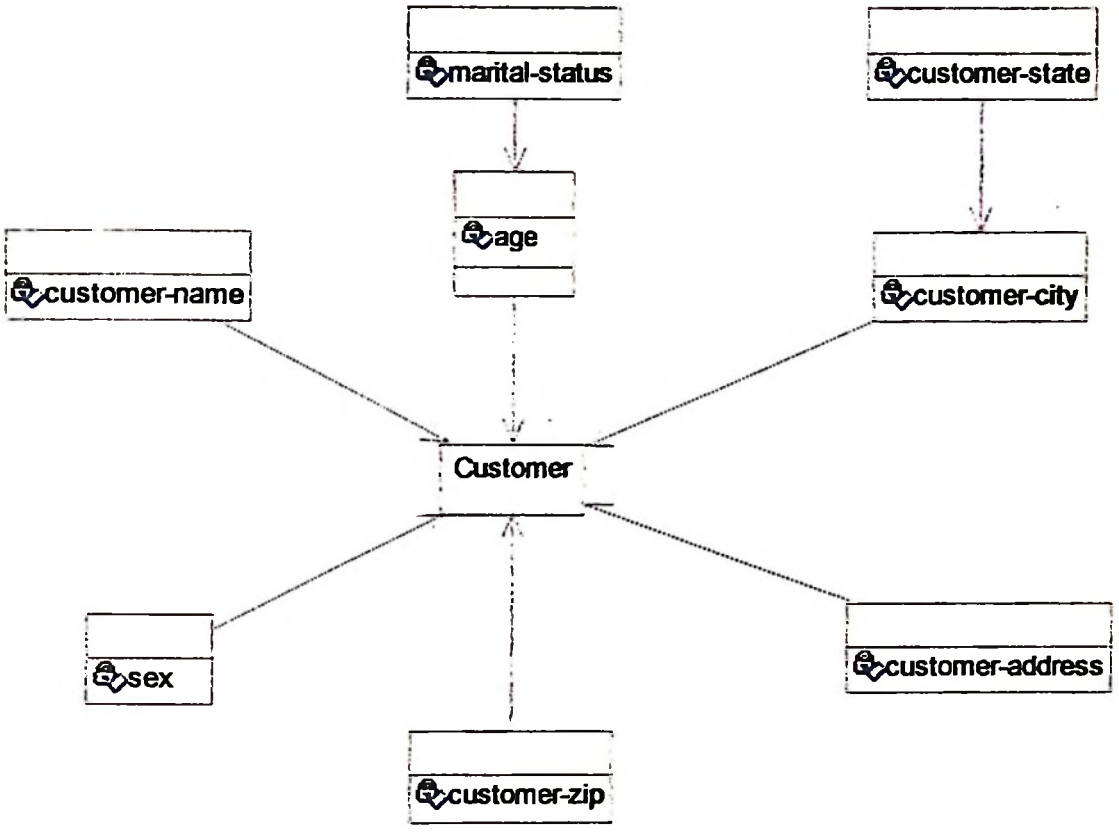


Fig.4.4

Branch Dimension.

Attribute Name	Attribute Description
Branch-name	The name of the branch.
Branch-address	The street address for this branch location. Note: the address number, suite/box, street name will be split apart for complete flexibility.

Attribute Name	Attribute Description
Branch-city	The name of the city where this branch location is.
Branch-state	The name of the state where this branch location is.
Branch-zip	The digit zip code for this branch location.
Branch-type	The type of the branch for example headquarter, corporate, normal or branch.

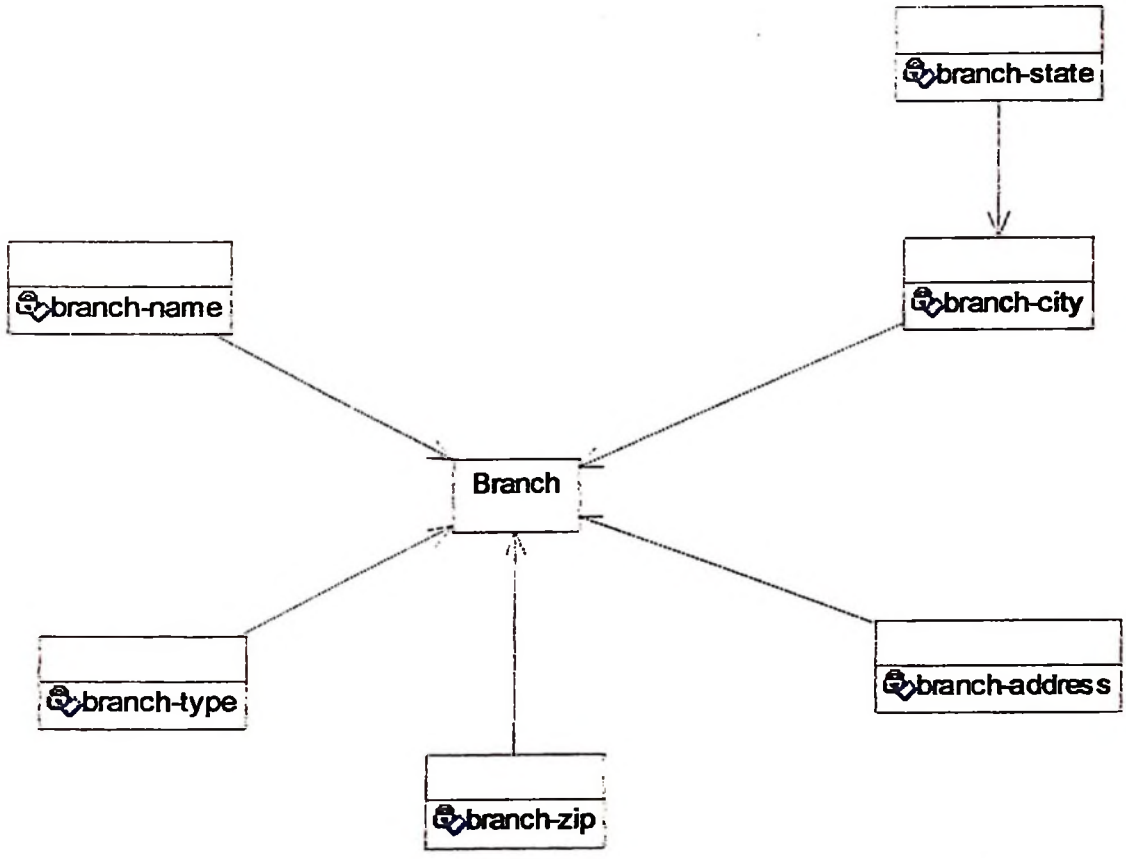


Fig.4.5

Loan Dimension.

Attribute Name	Attribute Description
Loan-purpose	The name of the branch.
Loan-type	The street address for this branch location. Note: the address number, suite/box, street name will be split apart for complete flexibility.
Branch-name	The name of the branch.

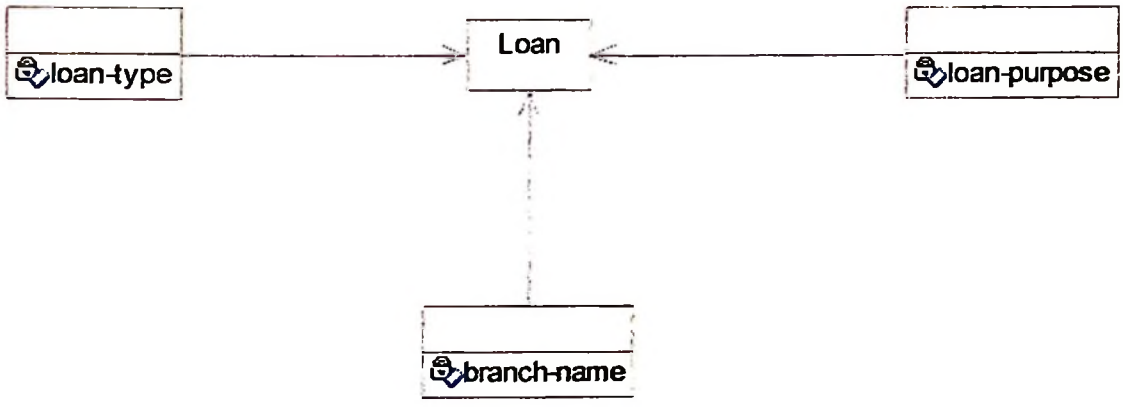


Fig.4.6

Status Dimension.

Attribute Name	Attribute Description
Status-description	This is used to record the condition of account.
Status-reason	This records whether the account is inactive and reason for account being closed.

Attribute Name	Attribute Description
New-account-flag	This is used to mark new accounts .
Close-account-flag	This is used to mark accounts that have just been closed.
New-loan-flag	This is used to mark accounts that have taken loan.
Paid-loan-flag	This is used to mark accounts that have paid loan.

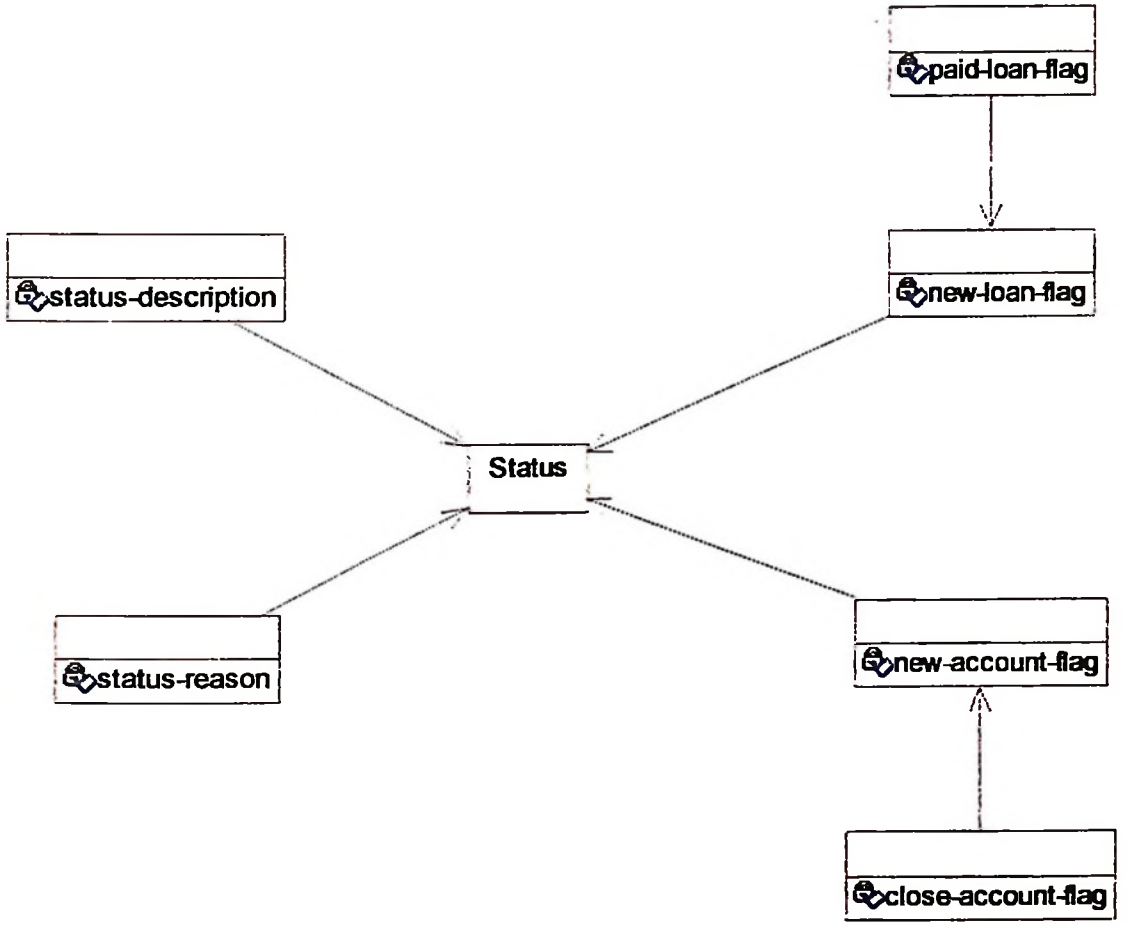


Fig.4.7

Time Dimension.

Attribute Name	Attribute Description
Month	One of the twelve divisions of the calendar year
Year	A period of time occupying a regular part of a calendar year that is used for some particular activity
Fiscal-quarter	A quarter within 12-month period for which an organization plans the use of its funds. This period may be a calendar year but can be any 12-month period.

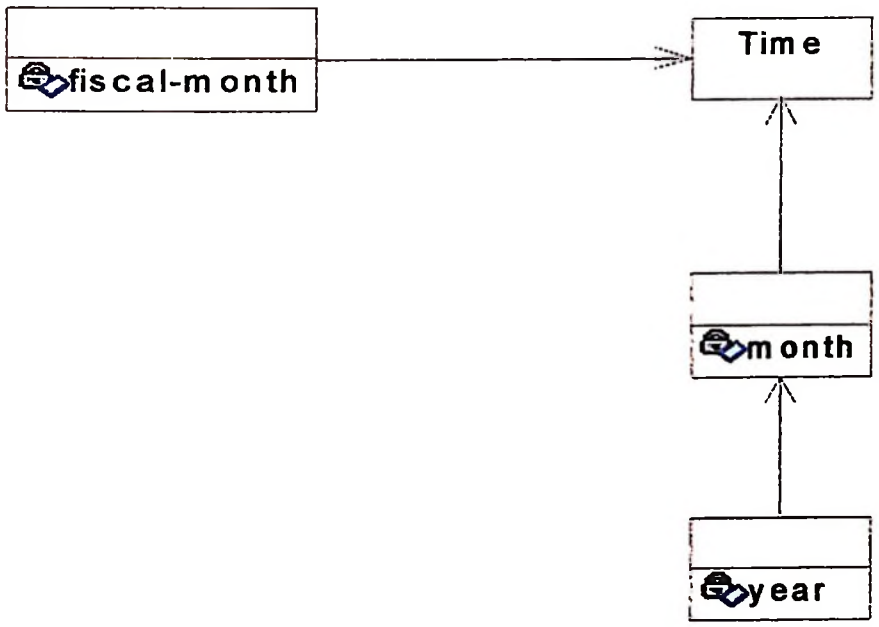


Fig.4.8

Payment Dimension.

Attribute Name	Attribute Description
Payment-date	Date at which loan paid to the bank by a customer.

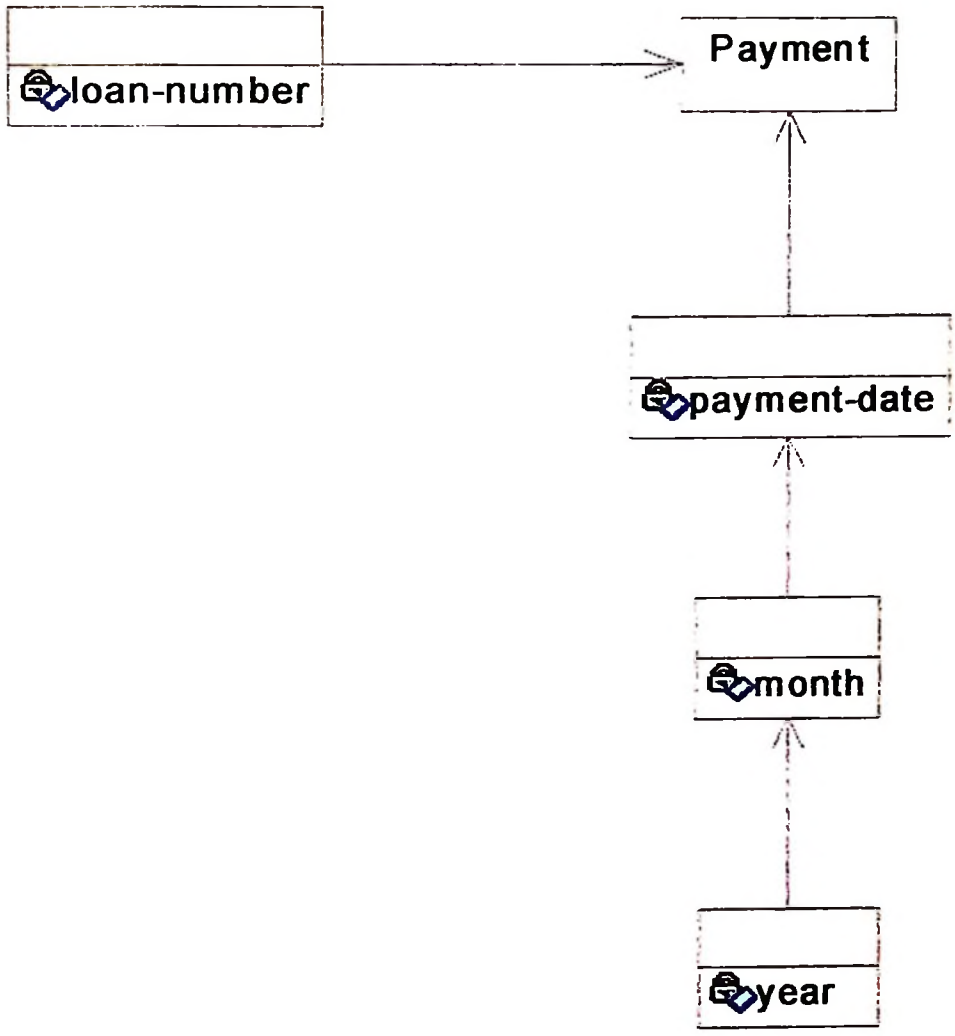


Fig.4.9

Account Dimension.

Attribute Name	Attribute Description
Branch-name	This is the name of the branch.
Date-opened	Date of opening an account.

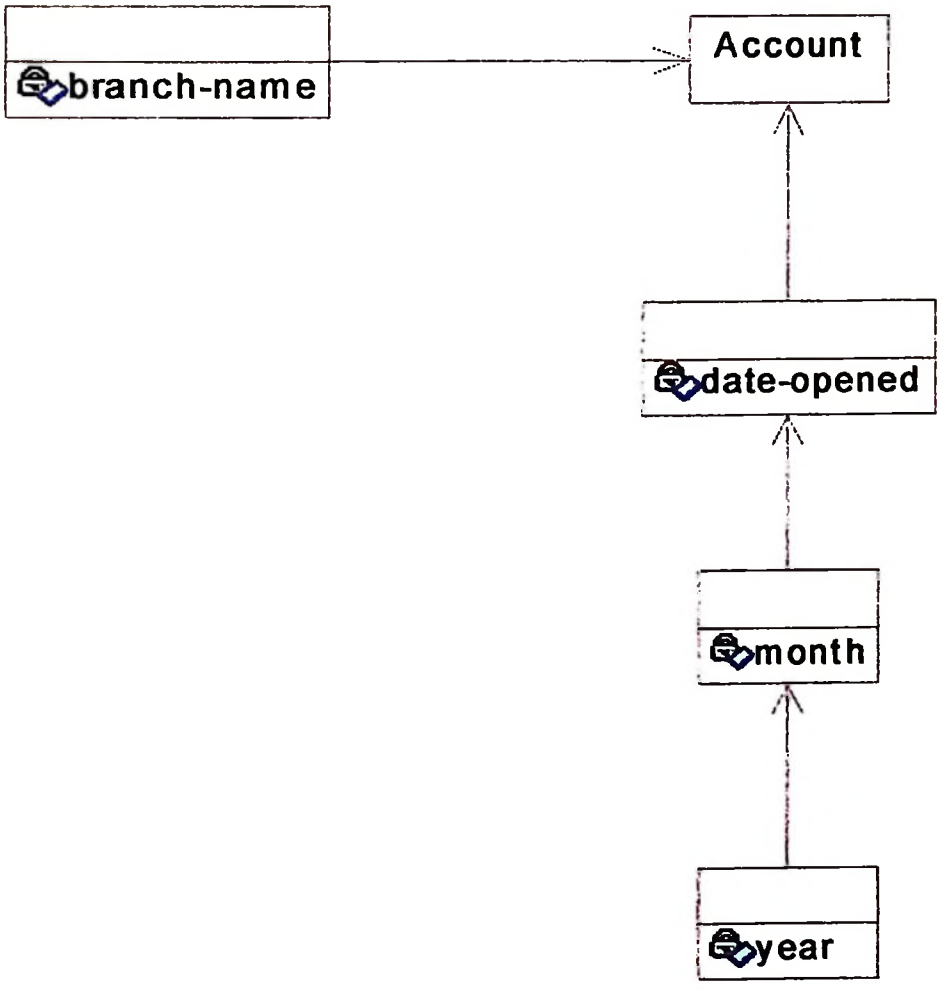


Fig.4.10

FACT DETAILS.

Account Core fact Table.

Fact Name	Fact Description
Balance	The difference between the total amounts of money coming into and going out of an account in a particular period of time.
Amount	A quantity of money
Transaction-count	Represents the total number of transaction performed during a sample period
Fees-charged	Fee charge to a saving account customer with less balance than the allowable minimum amount. Also means a fixed charge for borrowing money
Payment-amount	Total amount paid depending to loan taken by customer.

Savings account Custom Fact Table.

Fact Name	Fact Description
Deposits	An amount of money paid into a bank account.

Fact Name	Fact Description
Withdraws	An amount of money taken from a bank account.
Interest-earned	The fee paid or earned for the use of money
Balance	The difference between the total amounts of money coming into and going out of an account in a particular period of time.
Service-charges	An amount of money paid for service

Checking account Custom fact table.

Fact Name	Fact Description
ATM transaction	Transaction done through automatic teller machine by a customer.
Deposits	An amount of money paid into a bank account.
Checks-paid	A printed form that use to pay for something instead of using money.
Overdraft	An arrangement between a bank and a customer, allowing them to take out more money from their checking account.

4.3 Slowly Changing Dimensions.

Compared to the fact table, the dimension tables are more stable and less volatile. However, unlike the fact table, which changes through the increase in the number of rows, a dimension table does not change just through the increase in the number of rows, but also through changes to the attributes themselves.

From the consideration of changes to the dimension tables, we can derive the following principles:

- Most dimensions are generally constant over time.
- Many dimensions, though not constant over time, change slowly.
- The customer key of the source record does not change.
- The description and other attributes change slowly over time.
- In the source OLTP systems, the new values overwrite the old ones.
- Overwriting of dimension table attributes is not always the appropriate option in a data warehouse.
- The ways changes are made to the dimension tables depend on the types of changes and what information must be preserved in the data warehouse.

The usual changes to dimension tables in our web enables data warehouse may be classified into three distinct types:

1) Type 1 changes: correction of errors.

The solution to type 1 is to overwrite the dimension record with the new values, thereby losing history.

- The type 1 response to slowly changing dimension is used whenever the old value of the attribute has no significance or should be discarded.

2) Type 2 changes: preservation of history.

The solution to type 2 is to create a new additional dimension record using a new value of the surrogate key.

- The type 2 response to slowly changing dimension requires the use of surrogate key because we assume that the production customer key has not been allowed to change.

3) Type 3 changes tentative soft revisions.

The solution to type3 is to create an old field in the dimension record to store the immediate previous attribute value.

- The type 3 response to slowly changing dimension is used when a change is soft or tentative or when we wish to keep tracking history with the old value of attribute as well as new.

4.3.1. Nature of type 1 changes.

In our case study, we found the customer name s and marital status fall under this category. For example, suppose a spelling error in the customer name is corrected to read as Rose Diana from the erroneous entry of Rose Dyna and marital status changed from single to married.

Also, suppose the customer name for another customer is changed from Daniel Emma to Daniel Emma and marital status changed from single to married.

Consider the changes to the customer name in both cases. There is no need to preserve the old values. In case of Rose Diana the old one is erroneous and need s to be discarded. When the users need to find all the transactions from Rose Diana, the users will use the correct name. The same principles apply to the change in customer name for Daniel Emma.

But the change in the marital status is slightly different. This change can be handled in the same way as the change in customer name only if that change is a correction of error. Otherwise, you will cause problems when the users want to analyze orders by marital status.

INCREMENTAL LOAD

-TYPE 1 CHANGE

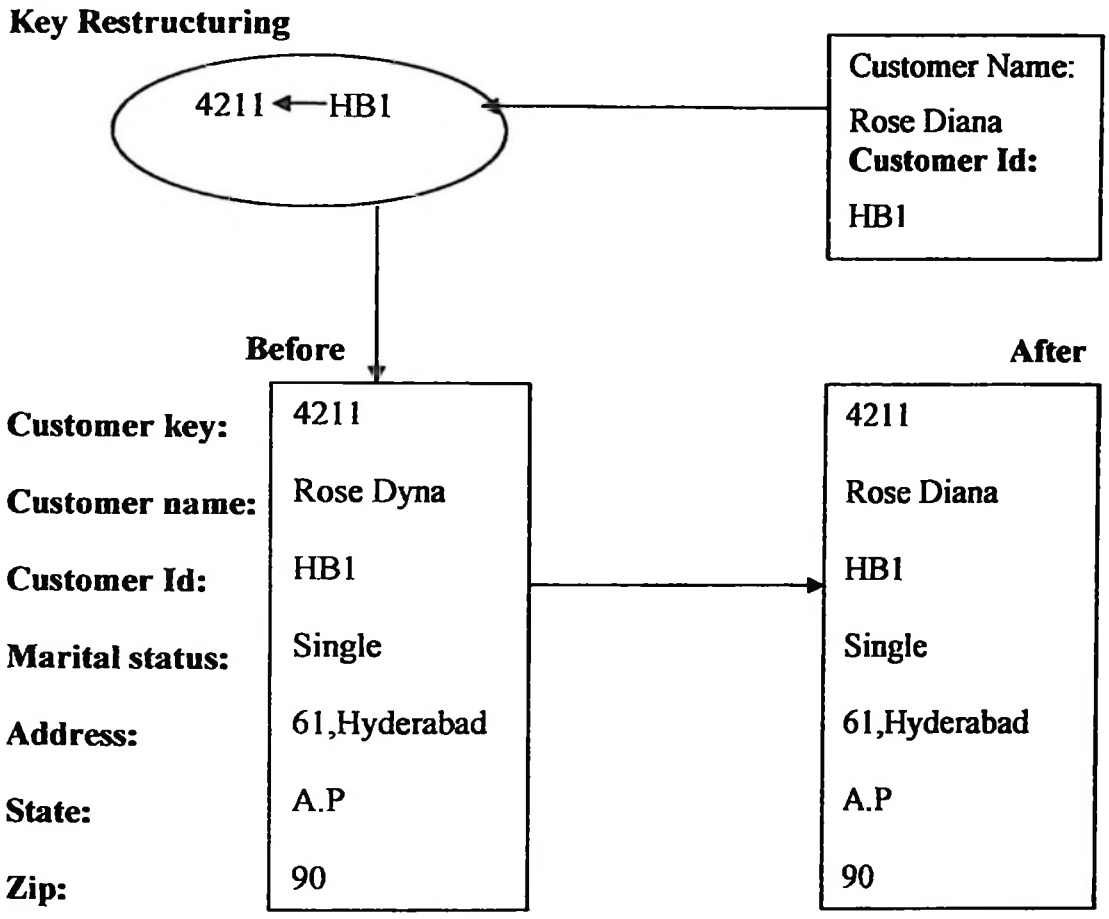


Fig.4.11: Applying Type 1 changes to the data warehouse.

By looking to the above Fig: [4.11] showing the application of Type 1 changes to the customer dimension table. The method for applying Type 1 changes is:

- Overwrite the attribute value in the dimension table row with the new value.
- The old value of attribute is not preserved.
- No other changes are made in the dimension table or any other key values are not affected.
- This type is easiest to implement.

4.3.2. Nature of type 2 changes.

In our web enabled data warehouse one of the essential requirements is to track the customer transactions and his/her business behaviors by marital status in addition of tracking using other attributes like state and address.

If the change to marital status happened on February 2,2004,all transactions from rose Diana before that date must be included under marital status: Single, and all transactions on or after April 4,2004,should be included under marital status: Married. Also she moved from her old address in A.P on February 2,2004,to U.P, therefore all transactions done prior to February 2,2004, must go under state A.P.

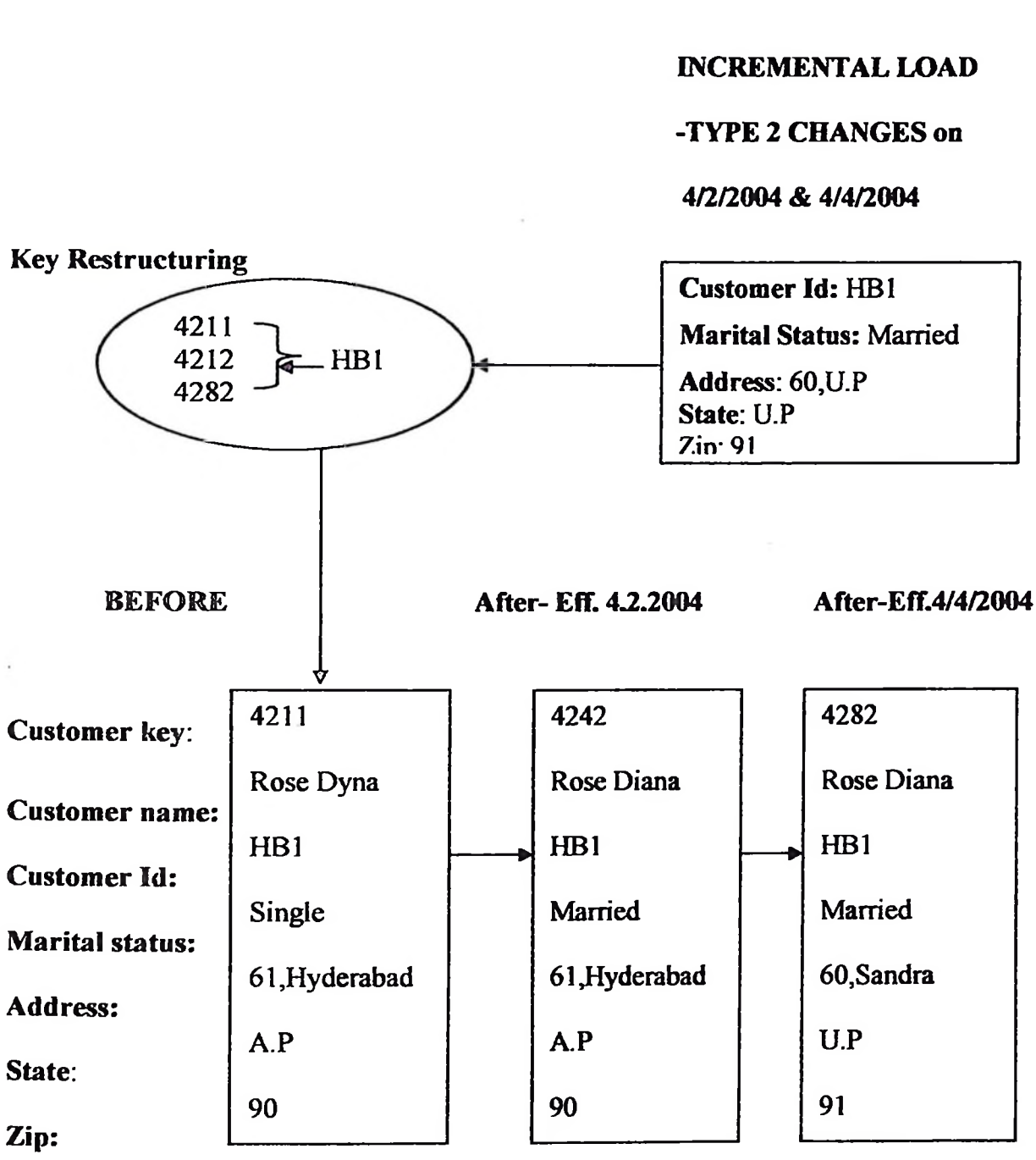


Fig.4.12: Applying Type 2 changes to the data warehouse.

The types of changes shown above from marital status and customer address are type 2 changes. The general principles for this type of change are:

- They usually relate to true changes in source systems.
- There is a need to preserve history in the data warehouse.
- This type of changes partitions the history in the data warehouse.
- Every change for the same attribute must be preserved.

Applying type 2 changes to the data warehouse.

The method for applying type 2 changes is:

- Add a new dimension table row with the new value of the changed attribute.
- An affective date (we have used record_Updated) field may be included in the dimension table.
- There are no changes to the original row in the dimension table.
- The key of the original row is not affected.
- The new row is inserted with a new surrogate key.

4.3.3. Nature of type 3 changes.

Sometimes, though rarely, there is a need to track both the old and new values of changed attributes for a certain period, in both forward and backward directions. These types of changes are Type 3 changes. As an example you can see Fig. [4.12] below.

The general principles for Type 3 changes:

- They usually relate to soft or tentative changes in the source systems.

- There is a need to keep track of history with old and new values of changed attribute.
- They are used to compare performances across the transition.
- They provide the ability to track forward and backward.

Applying Type 3 changes to the Data warehouse.

The methods for applying Type 3 changes are:

- Add an old field in the dimension table for the affected attribute.
- Push down the existing value of attribute from the current field to the old field.
- Keep the new value of the attribute in the current field.
- Also, you may add a current effective date field for the attribute.
- The key of row is not affected.
- No new dimension is needed.
- The existing queries will seamlessly switch to the current value.
- Any queries that need to use the old value must be revised accordingly.
- The technique works best for one soft change at a time.
- If there is a succession of changes, more sophisticated techniques must be devised.

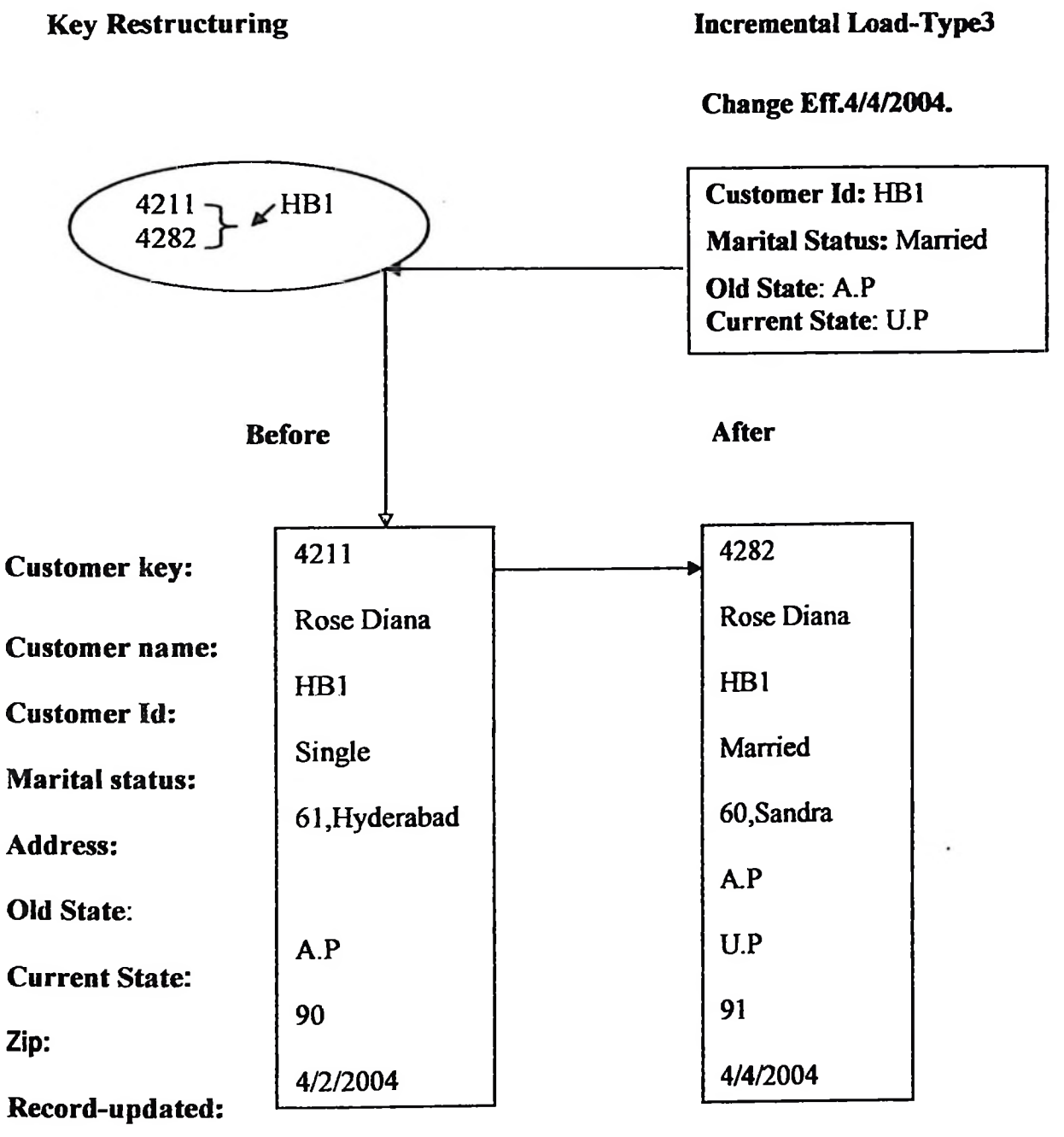


Fig.4.13: Applying Type 3 changes to the data warehouse.

Sequence Diagram

Sequence Diagram: Get Balance

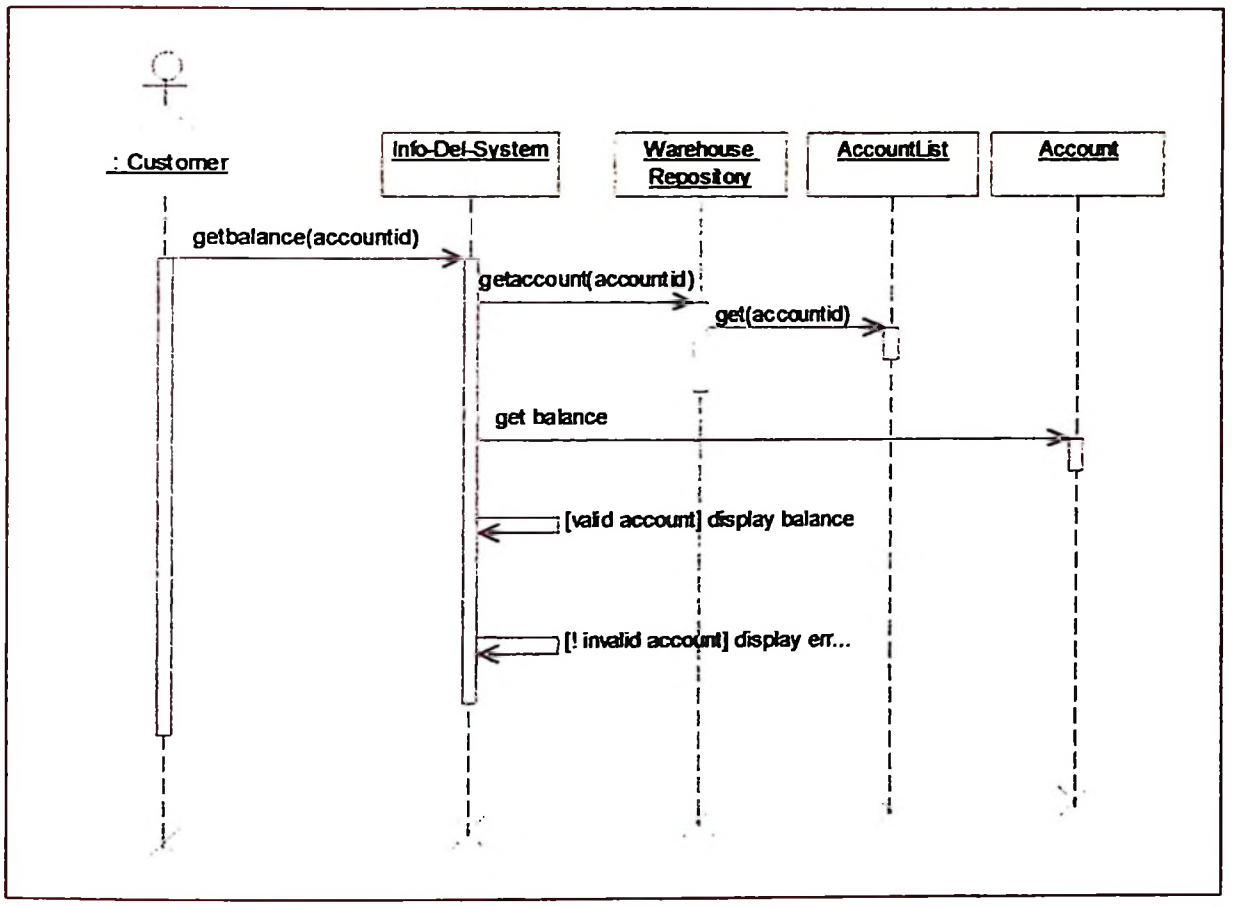


Fig.4.14

Sequence Diagram: generate account statistics

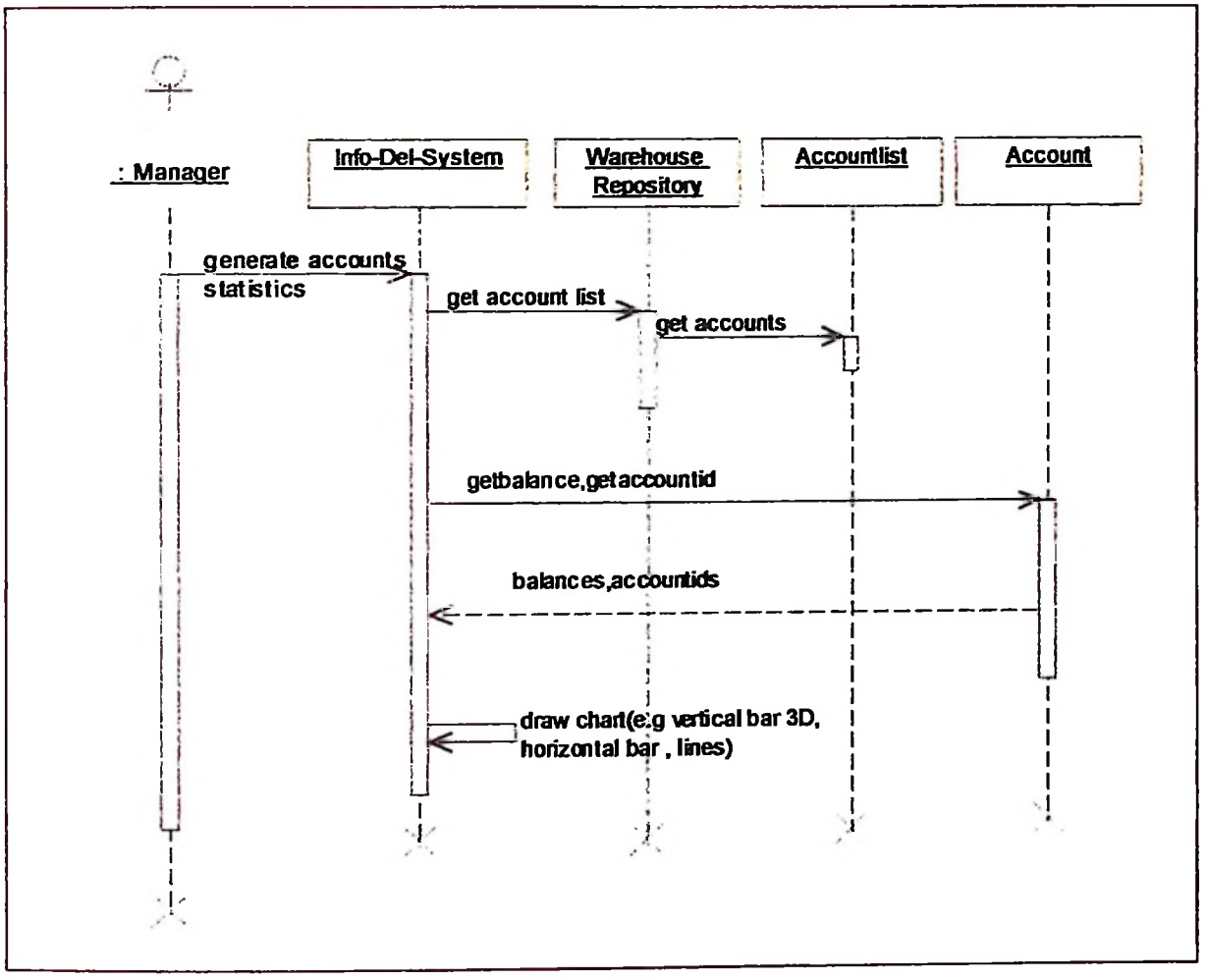


Fig.4.15

Sequence Diagram: list customers

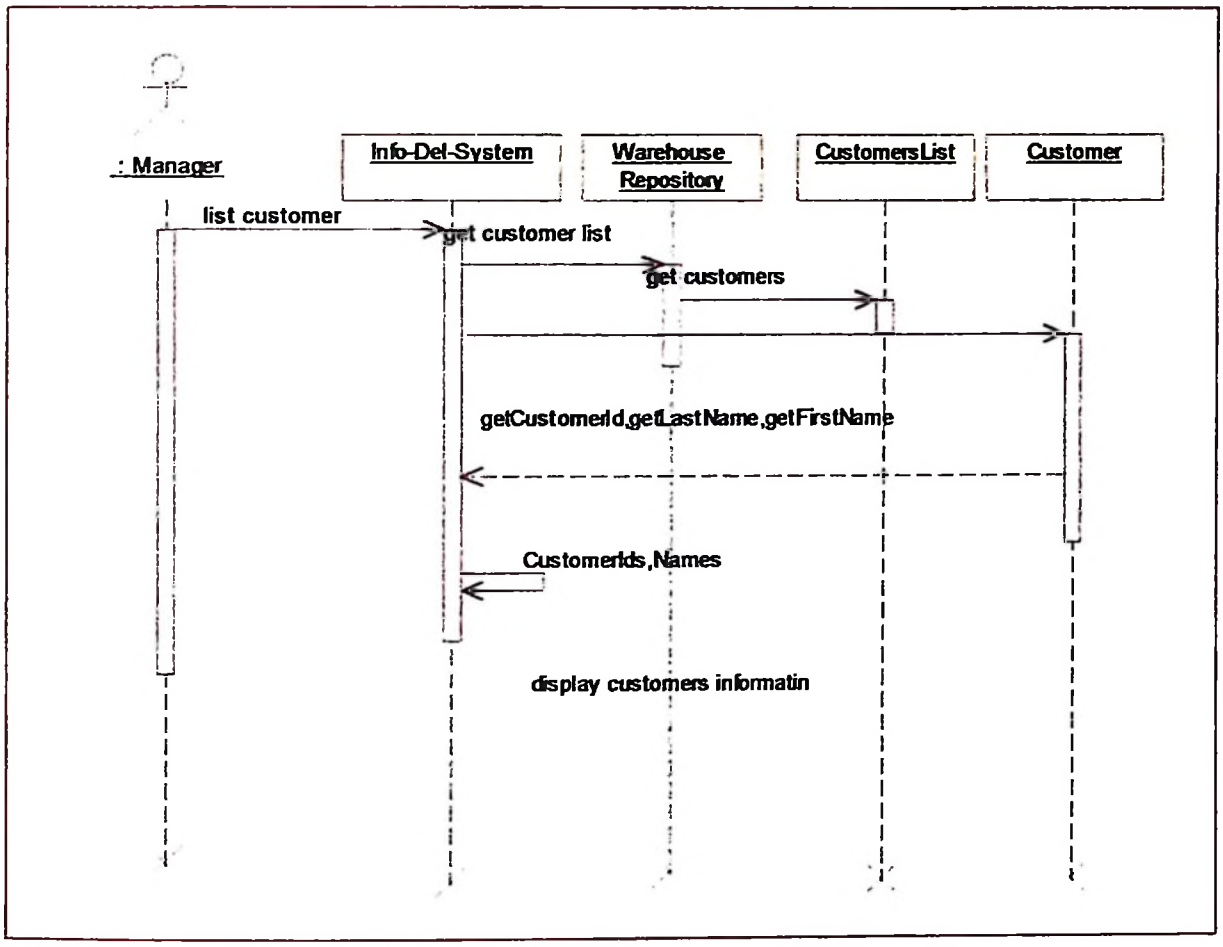


Fig.4.16

Collaboration Diagram: get balance

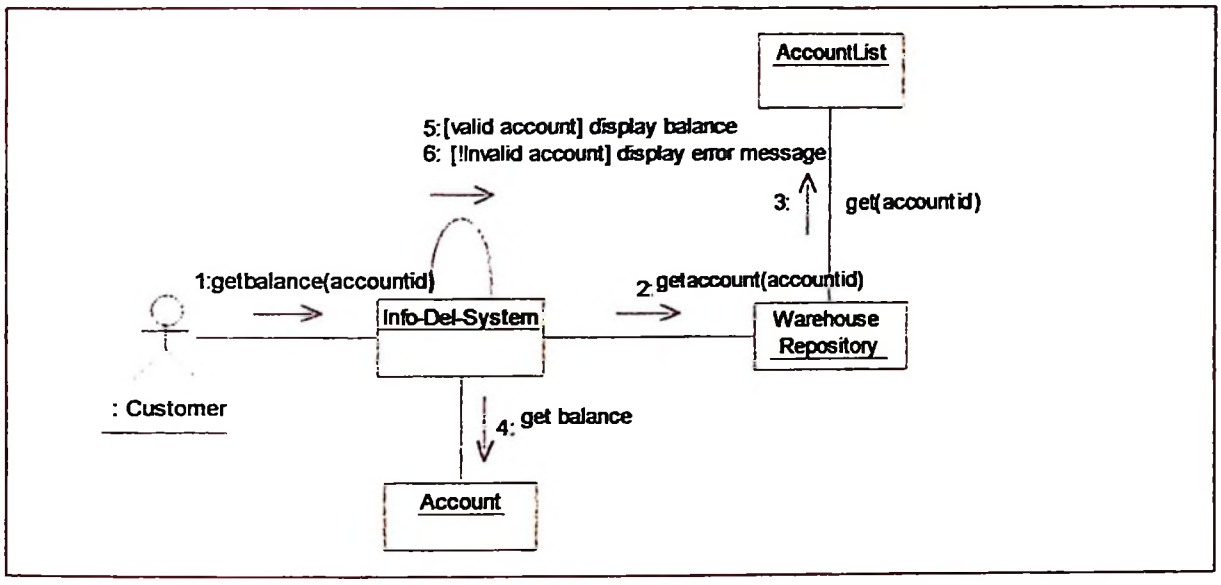


Fig.4.17

Collaboration Diagram: generate account

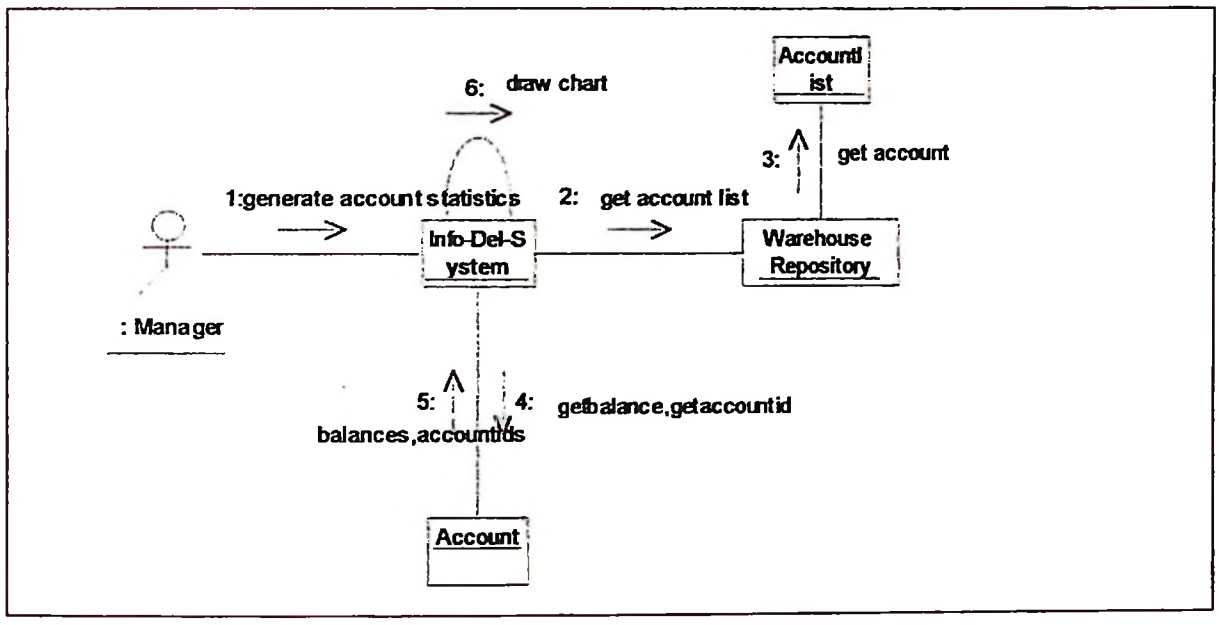


Fig.4.18

Collaboration Diagram: list customer

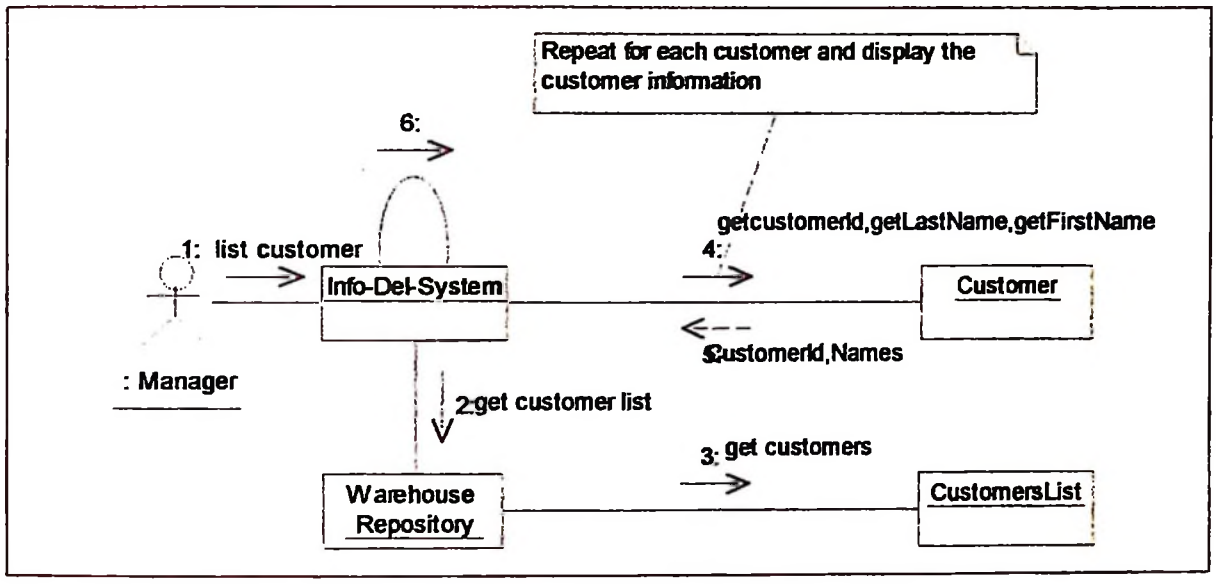


Fig.4.19

Chapter 5. OLAP CUBE DESIGN TECHNIQUES

OLAP is an acronym for **On Line Analytical Processing**, and it is becoming the fundamental foundation for Intelligent Solutions including Business Performance Management, Planning, Budgeting, Forecasting, Financial Reporting, Analysis, Simulation Models, Knowledge Discovery, and Data Warehouse Reporting.

OLAP perform multidimensional analysis of enterprise data and provides the capabilities for complex calculations, trend analysis and very sophisticated data modeling.
OLAP enables end-users to perform ad hoc analysis of data in multiple dimensions, thereby providing the insight and understanding they need for better decision-making.
It is FAST, EASY, SIMPLE, CONTROLLABLE access to the intelligence locked in your data.

On-line analytical processing (OLAP) requires efficient processing of complex decision support queries over very large databases. It is well accepted that pre-computed data cubes can help reduce the response time of such queries dramatically. A very important design issue of an efficient OLAP system is therefore the choice of the right data cubes to materialize. We call this problem the data cube schema design problem. In this chapter we show that the problem of finding an optimal data cube.

5.1 OLAP Cube Architecture

The primary OLAP object is the cube, a multidimensional representation of detail and summary data. A cube consists of a data source, dimensions, measures, and partitions. We design cubes based on the analytical requirements of users. A data warehouse can support many different cubes such as a Sales cube, an Inventory cube, and so on.

A cube's data source identifies and connects to the database containing the data warehouse data that is the source of data for the cube.

Dimensions map data warehouse dimension table information into a hierarchy of levels, such as a Geography dimension with levels of Continent, Country, State-Province, and City. Dimensions can be independently created and shared among cubes for ease of cube construction and to ensure consistency of analysis data summarization. For example, if a shared dimension is used for a product hierarchy in all appropriate cubes, the organization of summarized product information will be consistent among the cubes that use the dimension.

A virtual dimension is a special type of dimension that maps the properties of members of another dimension into a dimension that can then be used in cubes. For example, a virtual dimension of an account type property enables a cube to summarize data such as new-account-flag by type by balance, such as closed-account-flag by type by balance. Virtual dimensions and member properties are evaluated as necessary for queries and they require no physical cube storage.

Measures identify the numerical values from the fact table that are summarized for analysis such as balance, or amount.

Partitions are the multidimensional storage containers that hold cube data. Each cube contains at least one partition, and a cube's data can be combined from multiple partitions. Each partition can take its data from a different data source and can be stored in a separate location. A partition's data can be updated independently of other partitions in a cube. For example, a cube's data can be divided by time, with a partition for current year's data, another partition for the previous year's data, and a third partition for all data prior to the previous year.

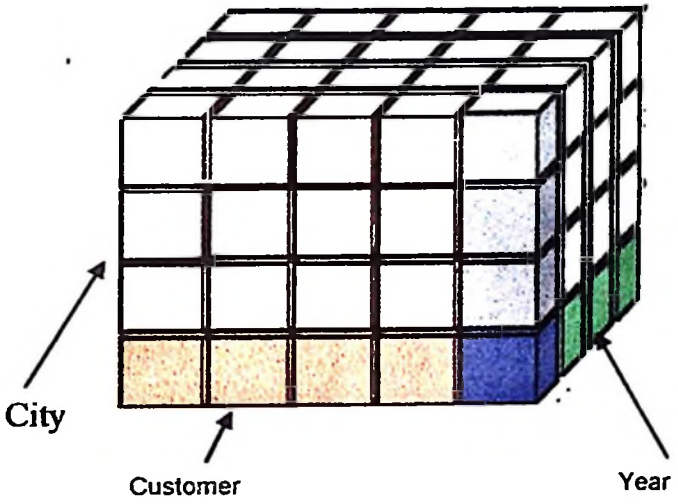
A cube's partitions can be independently stored in different storage modes with different degrees of summarization. Partitions are invisible to the user, to whom the cube appears to be a single object, yet they provide the administrator with a wide variety of options to manage the underlying OLAP data.

5.2 The Data Cube

Thus far in the Overview section, we have written informally about a multidimensional structure called the data cube. In short we have described it as a data abstraction that allows one to view aggregated data from a number of perspectives. Conceptually, the cube consists of a core or base cuboid, surrounded by a collection of sub-cubes/cuboids that represent the aggregation of the base cuboid along one or more dimensions. We refer

to the dimension to be aggregated as the *measure* attribute, while the remaining dimensions are known as the *feature* attributes.

The figure below depicts a small, practical data cube example. This particular data cube has three feature attributes — customer, city and year --- and a single measure attribute -- - balance. (Balance is computed with the sum function). By selecting cells, planes, or subcubes from the base cuboid, we can analyze balances figures at varying granularities. Such queries form the basis of OLAP functions like roll-up and drill-down.



A simple data cube

Fig.5.1

In total, a d-dimensional base cube is associated with 2^d cuboids. Each cuboid represents a unique view of the data at a given level of granularity.

Not all these cuboids need actually be present, however, since any cuboid can be computed by aggregating across one or more dimensions in the base cuboid. Nevertheless, for anything but the smallest data warehouses, some or all of these cuboids may be computed so that users may have rapid responses at run time.

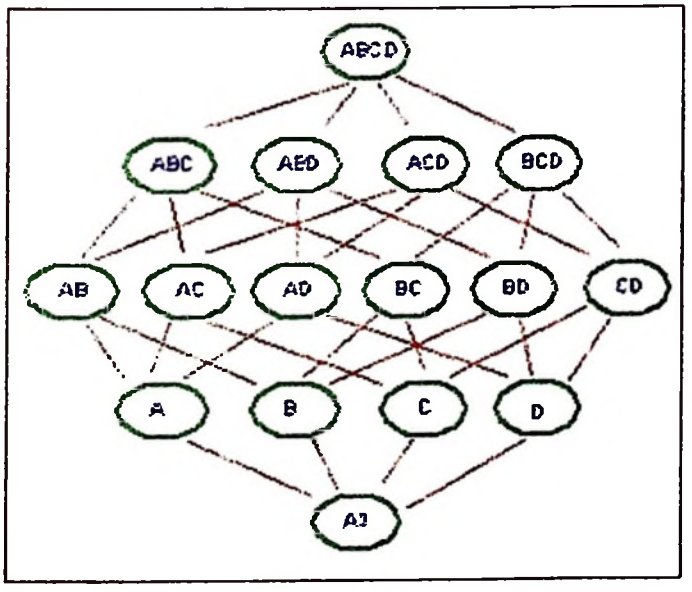
One final note is in order at this point. We have described the data cube as a conceptual model. This is certainly true. However, in the case of a MOLAP server, it is also the physical model, as MOLAP stores the cube structure directly as a multi-dimensional array. ROLAP servers must map this representation to a relational design.

5.3 Data Cube Algorithms

OLAP is multi-dimensional data. That being said, one might legitimately ask, "How does one prepare data for multi-dimensional analysis, particularly if some or all of the 2^d cuboids are required?" Strictly speaking, no special operators or SQL extensions are required to take a raw data set, composed of detailed transaction-level records, and turn it into a data structure, or group of structures, capable of supporting subject-oriented analysis. Rather, the SQL *group-by* and *union* operators can be used in conjunction with d sorts of the raw data set to produce all cuboids. However, such an approach would be tedious to program and immensely inefficient, given the obvious inter-relationships between the various view. Consequently, in 1995, Jim Gray et al. proposed the data cube operator as a means of simplifying the process of data cube construction.

Subsequent to the publication of the seminal data cube paper, a number of independent research projects began to focus on designing efficient algorithms for the computation of the complete cube.

Most were based upon the exploitation of the data cube *lattice*, a directed graph that depicts the relationships between all 2^d cuboids in a given d-dimensional space. (See the figure below for a graphical illustration.) Starting with the base cuboid — containing the full complement of dimensions — the lattice branches out by connecting every parent node with the set of child nodes/views that can be derived from its dimension list. In general, a parent containing *d* dimensions can be connected to d views at the next level in the lattice.



The lattice

Fig.5.2

It should be clear from the lattice depiction that many views share common dimension values and that any efficient computational mechanism for producing group-bys, whether it be sort-based or hash-based, must exploit these relationships. For example, a three-dimensional cuboid can be viewed as the parent of three two-dimensional cuboids, each of which contains a distinct combination of two dimensions of the parent. Clearly, it should not be necessary to independently compute all four views since the parent and one or more of the children may be able to share some portion of the aggregation workload.

Though a number of algorithms have been proposed, the techniques of primary importance can be roughly divided into three categories.

- **Top Down**. The top down methods work directly from the lattice to compute smaller group-bys from parents. For example, the parent view ABCD might be used to generate ABC, AB and A. What sets the top down methods apart is the means by which they share the computation costs across views. Perhaps the best-known methods in this class are the *PipeSort* and *PipeHash.* The basic premise of both algorithms is that a minimum spanning tree should be generated from the original lattice such that the cost of traversing edges — and thereby building cuboids — will be minimized.

- **Bottom Up.** As the dimensions increase, the high-dimension cuboids become increasingly *sparse.* Because child views are now almost as big as their parents, *bottom up* methods were proposed. Bottom up methods work by first aggregating (usually with a sort) on a single dimension, then recursively partitioning the current attribute in order to aggregate at successively finer degrees of granularity. Since the recursive sorting is performed on smaller and smaller partitions, most of the external memory sorting is avoided, restricted mainly to the first or second dimensions.

- **Array-based.** The algorithms presented above all correspond to the ROLAP model in that they operate on multi-dimensional tables. Given that many vendors (and customers) are attracted to the perceived performance benefits of the MOLAP model, it is also important to explore array-based approaches that directly support multi-dimensional data structures. In short, such methods structure the raw data set in a d-dimensional array that is typically stored on disk as a sequence of *chunks.* (A chunk is a means by which a large d-dimensional array can be partitioned into smaller d-dimensional sub-arrays). On dense sets, there is little questions that array-based algorithms are very efficient. In sparse, high dimensional environments, however, various performance-draining compromises must be employed in order to access the array.

In this chapter we describe a new approach to efficiently generate partial datacubes based on a parallel version of **Pipesort** and a new **greedy algorithm** to select intermediate views.

5.4 Generating Partial Datacubes

In the following section we present a high-level outline of our coarse-grained parallel partial datacube construction method. This method is based on sequential Pipesort and a parallel version of Pipesort described in.

The key to going from these methods for computing ***complete*** datacubes to a method for computing ***partial*** datacubes is Step 2 of the following algorithm - the greedy method for computing an efficient schedule tree for the partial datacube generation problem.

A Parallel Algorithm for Generating Partial Datacubes

1. **Build a Model:**

Construct a lattice for all 2^d views and estimate the size of each of the views in the lattice.

To determine the cost of using a given view to directly compute its children, use its estimated size to calculate

- the cost of scanning the view and
- the cost of sorting it.

2. **Compute a schedule tree using the model:**

Using the bipartite matching technique presented in Pipesort, reduce the lattice to a spanning tree that identities the appropriate set of prefix-ordered sort paths. Prune the spanning tree to remove any nodes that cannot possibly be used to compute any of the selected nodes. Run a greedy algorithm using the pruned tree to identify useful intermediate nodes. The tree built by the greedy algorithm contains only selected nodes and intermediate nodes and is called the schedule tree as it describes which views are best computed from which other views.

3. **Load balance and distribute the work:**

Partition the schedule tree into $s \times p$ sub-trees (s = oversampling ratio). Distribute the sub-trees over the p compute nodes. On each node use the sequential Pipesort algorithm to build the set of local views.

Given that finding the optimal schedule tree is NP-Complete, we need to find a method that takes a manageable amount of time to find a reasonable schedule.

In computing the schedule tree we propose starting from the spanning tree that is derived from Pipesort. Clearly there are many other approaches that could be taken. We chose this approach for our initial try at generating partial cubes because the Pipesort tree has proven to be effective in the generation of complete datacubes and therefore appears to be a good starting point for a schedule for partial datacubes. This choice is indeed supported by our experimental findings.

In the following sections we will describe exactly how the Pipesort tree is pruned, as well as the greedy algorithm for selecting intermediate nodes/views.
For a description of how the model is built and the details of the load balancing algorithm see.

The Pruning Algorithm.

Before passing the Pipesort tree to the greedy algorithm, we want to ensure that is has been pruned of any unnecessary nodes.
Quite simply, we remove any node from the tree whose attributes are not a superset of at least one selected node. The pseudo code can be written as follows:

Input: Spanning tree T and Subset S

Output: Pruned (spanning) tree T

```
for every node i in T - S
    for all nodes j of S
        if there is no node j whose
            attributes are a subset of the
                attributes of i delete node i
                from T
```

The operation of this simple quadratic time algorithm is illustrated in Figure 5.3.

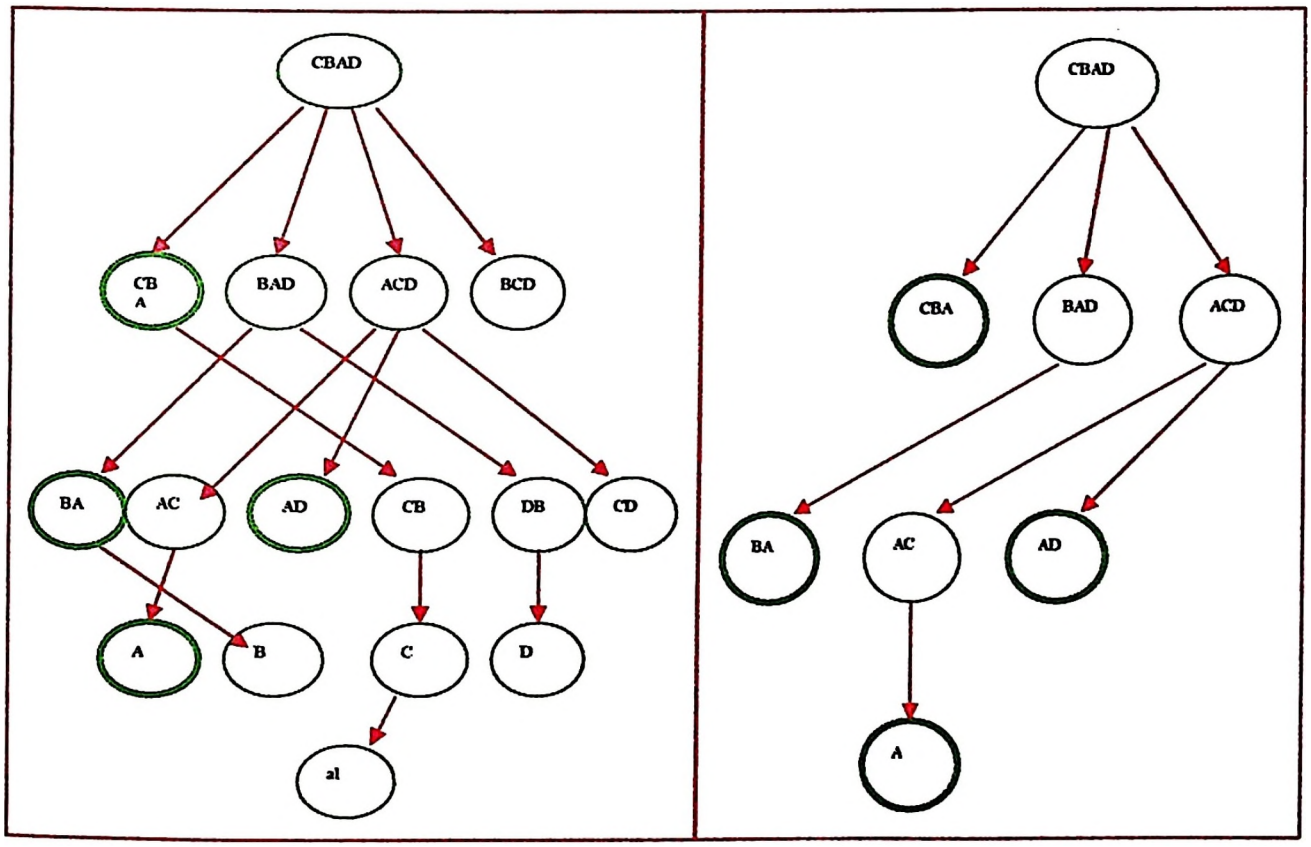


Fig.5.3

Fig 5.3. Graph Pruning. Left hand side: Spanning Tree of the lattice as created by Pipesort with selected nodes in hold. Right hand: Pruned Tree.

The GreedyAlgorithm.

The greedy algorithm takes as input a spanning tree T of the lattice that has been pruned and a set S of selected nodes representing those views to be materialized as part of the partial datacube.

The algorithm begins by assuming that its output, the schedule tree T', will consist only of the selected nodes organized into a tree based on their relative positions in T.
In other words, if a selected node a is a descendant of a selected node b in T, the same relationship is true in the initial schedule tree T'. The algorithm then repetitively looks for intermediate nodes that reduce the total cost of the schedule. At each step, the node from T that most reduces the total cost of T' is added. This process continues until there are no nodes which provide a cost improvement.

Figure 5.4 shows an example of an initial schedule tree T' for a five dimensional cube with attributes A, B, C, D, and E and selected views A, AB, BC, CD, DE, and DAE, as well as a possible final schedule tree, T'.
The pseudo code for the greedy algorithm is as follows:

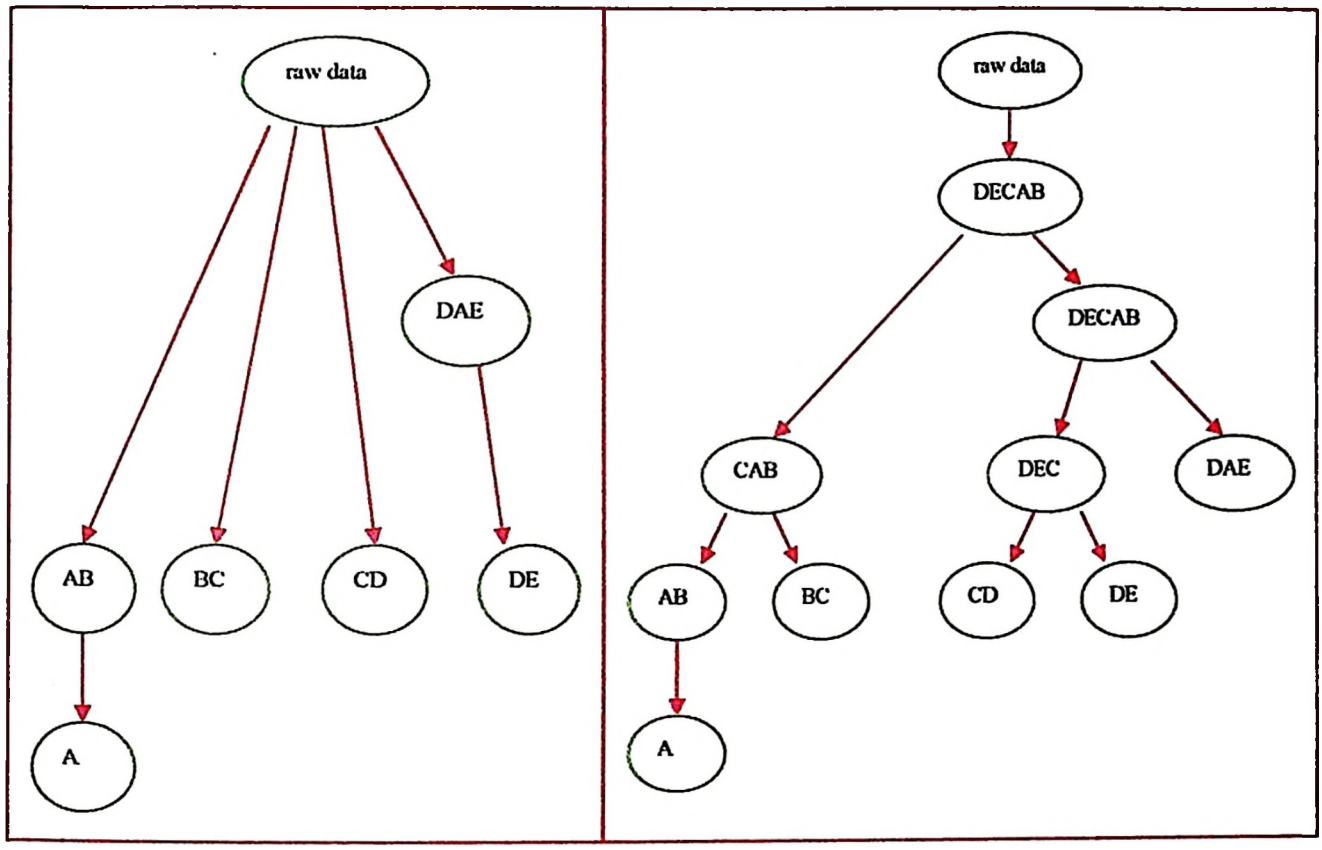

raw data
AB
BC
CD
DE
DAE
A
raw data
DECAB
DECAB
CAB
DEC
DAE
AB
BC
CD
DE
A

Fig .5.4

Figure 5.4. The Schedule tree. Left hand: The initial schedule tree T' containing only selected nodes. Right hand: An example final containing both selected and intermediate nodes.

```
Input: Spanning tree T and Subset S

Output: Schedule tree T'


Initialize T' with nodes of S
    each node of S is connected to its
    immediate predecessor in S edges are
    weighted accordingly.
while global_benefit >= 0
for each node i in T - S
/*compute the benefit of including node i*/
    for every node j in S
        if attributes of j are a prefix of i
            local_benefit += current_cost of
                j - cost of scanning i
```

```
        else if attributes are a subset of i
    local_benefit += current_cost of j - cost
    of resorting i
local_benefit -= cost of building i
if local_benefit > global_benefit
        global_benefit = local_benefit
        best node = i
if global_benefit > 0
add node i to T'
```

Chapter 6. DATA WAREHOUSE AND THE WEB

A Web-enabled data warehouse uses the Web for information delivery and collaboration among users. As months go by, more and more data warehouses are being connected to the Web. Essentially, this means an increase in the access to information in the data warehouse. Increase in information access, in turns, means increase in the knowledge level of the enterprise. It is true that even before connecting the Web, you could give access for information to more of your users, but with much difficulty and a proportionate increase in communication costs. The Web has changed all that. It is now a lot easier to add more users. The communications infrastructure is already there. Almost all of your users have Web browsers. No additional client software is required. You can leverage the Web that already exists. The exponential growth of the Web, with its networks, servers, users, and pages, has brought about the adoption of the Internet, intranets and extranets as information transmission media. The Web-enabled data warehouse takes center stage in the Web revolution.

Why the Web?

It appears to be quite natural to connect the data warehouse to the Web. Why do we say this? For a moment, think of how users view the Web. First, they view the Web as a tremendous source of information. They find the data content useful and interesting. Your internal users, customers, and business partners already use the Web frequently. They know how to get connected. The Web is everywhere. The sun never sets on the Web. The

only client software needed is the Web browser, and almost everyone, young and old, has learned how to launch and use the browser.

Now consider your data warehouse in relation to the Web. Your users need the data warehouse for information. Your business partners can use some of the specific information from the data warehouse. What do all of these have in common? Familiarity with the Web and ability to access it easily.

Three information delivery mechanisms that can adapt based on Web technology. In each case, users access information with Web browsers.

Internet. The first medium is, of course, the Internet, which provides low-cost transmission of information. You may exchange information with anyone within or outside the company. Because the information is transmitted over public networks, security concerns must be addressed.

Intranet. From the time the term, "intranet" was coined in 1995, this concept of a private network has gripped the corporate world. An intranet is a private computer network base on the dada communications standards of the public Internet. The applications posting information over the intranet all reside within the firewall and, therefore, are more secure. You can have all the benefits of the popular Web technology. In addition, you can manage security better on the intranet.

Extranet. The Internet and the intranet have been followed by the extranet. An extranet is not completely open like the Internet, nor it is restricted just for internal use like an intranet. An extranet is an intranet that is open to selective access by outside parties. From your customers, suppliers, and business partners.

The figure below illustrates how information from the data warehouse may be delivered over these information delivery mechanisms.

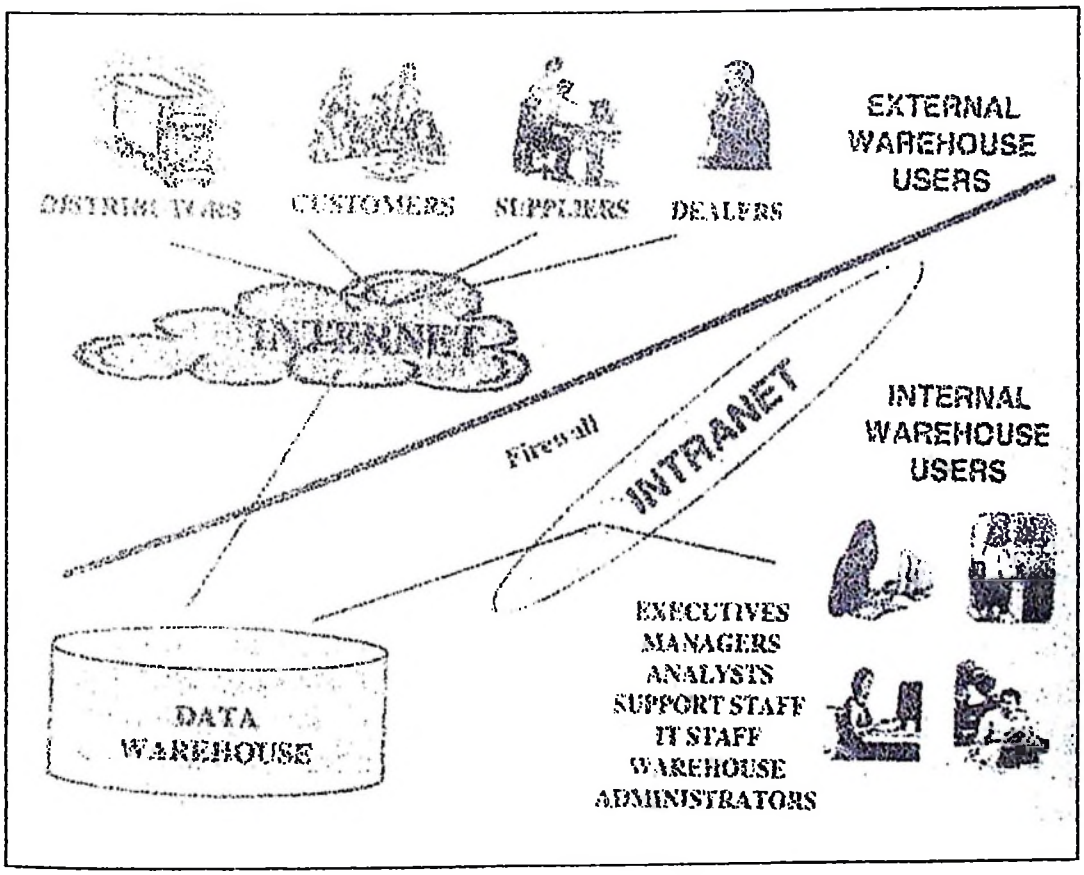


Fig. 6.1

This is how your data warehouse may be deployed over the Web.

If you choose to restrict your data warehouse to internal users, then you adopt the intranet.

If it has to be opened up to outside parties with proper authorization, you go with the extranet. In both cases, the information delivery technology and the transmission protocols are the same.

The intranet and the extranet come with several advantages. Here are a few:

- With a universal browser, your users will have a single point of entry for information.
- Minimal training is required to access information. Users already know how to use a browser.
- Universal browsers will run on any systems.
- Web technology opens up multiple information formats to the users. They can receive text, images, charts, even video and audio.
- It is easy to keep the intranet/extranet updated so that there will be one source of information.
- Opening up your data warehouse to your business partners over the extranet fosters and strengthens the partnerships.
- Deployment and maintenance costs are low for Web-enabling your data warehouse. Primarily, the network costs are less. Infrastructure costs are also low.

6.1 Adapting the Data warehouse for the Web

Much is expected of a Web-enabled data warehouse. That means we have to reinvent our data warehouse. A number of tasks must be carried to adapt the data warehouse for the Web.

The requisites for adapting the data warehouse to the Web.

- **Information "Push" Technique.** The data warehouse was designed and implemented using the "pull" technique. The information delivery system pulls information from the data warehouse based on requests, and then provides it to the users. The Web offers another technique. The Web can "push" information to users without their asking for it every time. Your data warehouse must be able to adopt the "push" technique.

- **Ease of Usage.** With the availability of click stream data, you can very quickly check the behavior of the user at the site. Among other things, click stream data reveals how easy or difficult it is the users to browse the pages. Ease of use appears at the top of the list of requirements.

- **Speedy Response.** Some data warehouses allow jobs to run long to produce the desired results. In the Web model, speed is expected and cannot be negotiated or compromised.

- **No Downtime.** The Web model is designed so that the system is available all the time. Similarly, the Web-enabled data warehouse has no downtime.

- **Multimedia Output.** Web pages have multiple data types-textual, numeric, graphics, sound, video, animation, audio, and maps. These types are expected to show as outputs in the information delivery system of the Web-enabled data warehouse.

- **Market of One.** Web information delivery is tending to become highly personalized, with dynamically created XML pages replacing static HTML coding, Web-enabled data warehouses will have to follow suit.

- **Scalability.** More access, more users, and more data-these are the results of Web-enabling the data warehouse. Therefore, scalability becomes a primary concern.

6.2 OLAP and the Web

Large amounts of time and money are invested in building data warehouses in the hope that the enterprise will get the information it needs to make strategic decisions of lasting value. For maximizing the value potential, you need to cater to as large a user group as possible to tap into the potential of the warehouse. This includes the extension of OLAP capabilities to a larger group of analysts.

6.2.1 Web-OLAP Approaches

The underlying combination for a successful implementation is comprised of the Web technology, the data warehouse with its OLAP system, and a thin-client architecture. How do you implement OLAP in such an environment? How will the OLAP system work in your Web-enabled data warehouse? What kind of client and Web architecture will produce the optimum results? These questions may be approached in different ways.

1. **Browser Plug-ins.** In the first approach, you use plug-ins or browser extensions. This is just a slightly modified version of fat-client Windows implementation except that the client configuration is more towards that of a thin client. Support issues creep in and this approach has scalability problems.

2. **Precreated HMTL Documents.** In the next approach, you provide precreated HTML documents along with the navigation tools to find these. The documents are results sets of analytical operations. This approach takes advantage of Web technology and thin-client economy, but users are confined to using predefined reports. The approach is devoid of on-demand analysis; users cannot do typical online analytical processing.
3. **OLAP in the Server.** The best approach is to use the server to do all online analytical processing and prevent the results on a true thin-client information interface. This approach realizes the economic benefits of the Web and thin-client architecture. At the same time, it provides an integrated server environment irrespective of the client machines. Maintenance is minimized because applications and logic are centralized on the server. Version control is also consistent. Everyone shares the same components server, metadata, and reports. This approach works well in production environments.

6.2.2 OLAP Engine Design

When the data warehouse is web-enabled and the level of OLAP operations raised, the design of the OLAP engine determines possibilities for scaling up. In the OLAP product you choose for your Web-enabled data warehouse, OLAP engine design ranks high in criticality. A property designed engine produces a performance curve that stays linear as the number of current users increases.

Consider the following options:

Dependence on the RDBMS. The OLAP engine relies completely on the RDBMS to perform multidimensional processing, generating complex, multipass SQL to access summary data Joins, aggregations, and calculations are all done within the database, posing serious problems for Web-enabled systems. A large number of temporary tables become necessary. The overhead for creating, inserting, dropping, allocating disk space, checking permissions, and modifying system tables for each calculation is enormous. Just five concurrent users may bring the OLAP system to its knees.

Dependence on the Engine. Here the engine generates SQL to access summary data and performs all processing on a middle tier. You will observe two problem areas in this approach. Heavy network traffic and large memory requirements make this approach undesirable. You may get a linear performance curve, but the curve is likely to be too steep because the potential of the DBMS is not being used.

Intelligent OLAP Engine. The engine has the intelligence to determine the type of request and where it will be performed optimally. Because of its intelligence, the engine is able to distribute the joins, aggregations, and calculations between the engine component and the RDBMS. In this model, you are able to separate the presentation, logic, and data layers both logically and physically. Therefore, system processing is balanced and network traffic is optimized. Currently, this seems to be the best approach, achieving a performance curve that remains linear with a gradual inclination as the number of concurrent users increases.

6.3 Security Issues

Without a doubt, when you open up your Web-enabled data warehouse to users throughout the enterprises via intranet and to business partners outside via an extranet, you tend to maximize the value. Depending on your organization, you may even drive more value when you take the next step and open the warehouse to the public on the Internet. But these actions raise serious security issues. You may have to impose security restrictions at different levels.

At the network level, you may look into solutions that support data encryption and restricted transfer mechanisms. Security at the network level is just one piece in the protection scheme. Carefully institute a security system at the application level. At this level, the security system must manage authorizations on who is allowed to get into the application and what each user is allowed to access.

Chapter 7. DATA QUALITY.

7.1. What Is Quality Data?

Quality data is, simply put, data that meets business needs. Quality data don't have to be perfect, just accurate, complete, consistent, timely, and flexible enough to meet business needs.

The data quality processes are not utilized just during the population of data warehouse, they are also employed before the warehouse is even designed and after the warehouse is loaded. The goal is to have data free of duplicates, misspellings, omissions and unnecessary variations and to have the data conform to the defined structure. Simply put, the data quality addresses the problem cynically but precisely summed in "garbage in – garbage out".

According to Andrew Ippilito, data has a number of quality characteristics:

Accuracy

1) The measure or degree of agreement between a data value (or set of values) and a source assumed to be correct.

2) A qualitative assessment of freedom from error.

Completeness

1) The degree to which values are present in the attributes that require them.

Consistency

1) Data are maintained so they are free from variation or contradiction.

2) The measure of the degree to which a set of data satisfies a set of constraints.

Timeliness

1) The extent to which a data item or multiple items are provided at the time required or specified.

2) A synonym for currency, the degree to which specified values are up to date.

Uniqueness

1) The ability to establish the uniqueness of a data record (and data key values).

Validity

1) The quality of the maintained data is rigorous enough to satisfy the acceptance requirements of the classification criteria.

2) A condition where the data values pass all edits for acceptability, producing desired results.

Data must be properly labeled and defined to be meaningful. (That is to say, the data describing the data - **metadata** - must be accurate.) For example, if a field is labeled as "Address," it ought not to have a telephone number in it. It has been said that metadata is a description of information that planners and designers wish would be put into the database - it often happens that data fields are used for a variety of unintended purposes.

Another characteristic is:

Flexible — the data definitions are understood so that data can be analyzed in a number of ways.

Therefore, when establishing data quality standards, it's vital to understand the end use of the data.

7.2.The Cost of Poor-Quality Data

As more banks implement web enabled data warehouses, they encounter the high costs imposed by poor-quality data.

7.2.1 Business Process Costs

When inaccurate information is entered into a web enabled data warehouse (banking system), standard processes fail. At Banking Institutions, process failures due to inaccurate information range from customers not being registered for the correct accounts, to inaccurate loans details, to customer security assets.

7.2.2 Cost to Rework Information

When data is entered incorrectly or is considered unreliable, data users must

i. Collect the same information from other sources.

ii. Correct errors.

iii. Verify the data's accuracy.

All of these activities consume banking staff time and add cost.

7.2.3 Lost and Missed Opportunities

As the banking sector becomes more competitive, institutions need quality data to support decision-making and assist in delivering services to customers. Poor-quality data can result in substandard customer service (for example, potential customers not receiving monthly reports due to inaccurate address information). Poor-quality data can also result in ineffective decision-making and potentially a loss of reputation.

Although data cleansing and business practice change are extremely important, you could justify the expenditure of money and effort by counting the costs of not having or using quality data. Banking staffs are the ones who can really do cost estimates because the estimates are based on forecasts of lost opportunities and possible bad decisions.

The following is a list of categories for which cost estimates can be made.

i. Bad decisions based on routine analysis.

ii. Lost business opportunities because of unavailable or dirty data.

iii. Strain and overhead on source systems because of corrupt data causing reruns.

iv. Fines from governmental agencies for noncompliance or violation of regulations.

v. Redundant data unnecessary using resources.

vi. Inconsistent reports.

7.3 Improving Data Quality

The first step in improving bank institutional data quality is realizing that problems exist. Data quality issues usually become evident when a bank institution starts planning a data warehouse project. For many institutions, such projects force them to start reviewing data and addressing quality problems.

To improve data quality, a bank institution must be prepared to

- ✓ Take a hard look at their business practices related to the collection and recording of data.
- ✓ Establish a data custodian (owner) for every piece of information they collect and make that custodian responsible for their data quality.
- ✓ Perform a thorough and objective analysis of existing data quality.

- ✓ Correct existing data quality problems.
- ✓ Make permanent changes to business practices to improve future data.
- ✓ Make the review and improvement of data quality an ongoing process.

Quality Data — An Improbable Dream?

Four step process to help banks institutions identify data quality issues and establish an ongoing data quality improvement process.

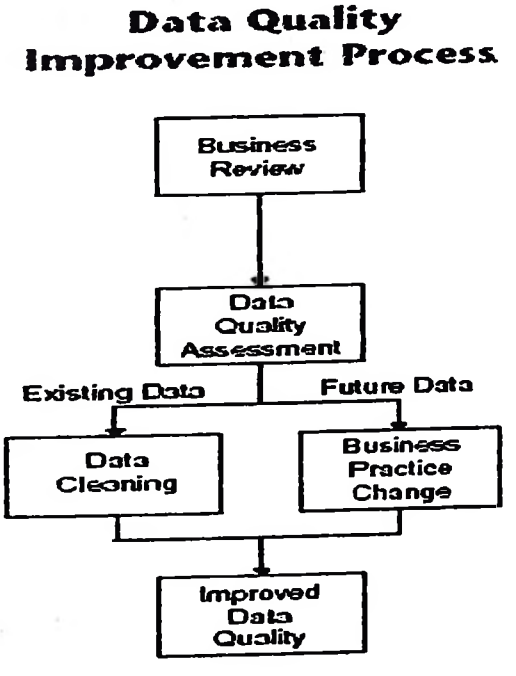


Fig.7.1: Data Quality Improvement Process

7.3.1 Business Review

The first stage in the project's process for improving data quality is to perform a thorough review of the bank institution's business practices related to the collection and recording of data.

- ✓ Correct existing data quality problems.
- ✓ Make permanent changes to business practices to improve future data.
- ✓ Make the review and improvement of data quality an ongoing process.

Quality Data — An Improbable Dream?

Four step process to help banks institutions identify data quality issues and establish an ongoing data quality improvement process.

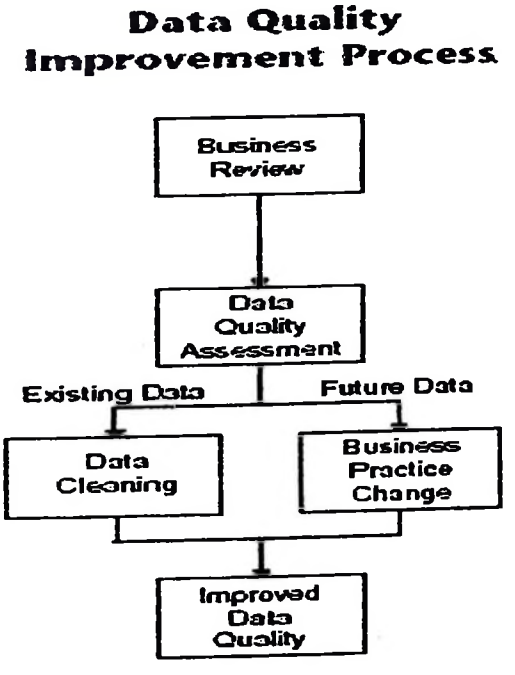


Fig.7.1: Data Quality Improvement Process

7.3.1 Business Review

The first stage in the project's process for improving data quality is to perform a thorough review of the bank institution's business practices related to the collection and recording of data.

The review process asks

- How and when are data collected?
- Where and how in the institution's administrative system(s) are the data recorded?
- Are data recorded in more than one system? If so, how are they reconciled?
- What quality checks ensure accurate data capture?
- Who is the custodian of each data element?
- Who creates the data? (Who collects and enters data?)
- Who uses the data?
- What types of reporting are required?

Our project involves as many stakeholders from the bank institutions as possible. For the banking system analysis and design we conducted interviews to the offices of the branch Manager, Management of Information System department, (IT department), and any other offices responsible for creating or using data recorded in the banking system for State Bank of Hyderabad (SBH, Osmania Branch).

The results of the business review include

- √ A comprehensive understanding of the institution's business practices that impact data quality.
- √ Identification of individuals responsible for data collection and quality control.
- √ Identification of all data users and their requirements.
- √ Preliminary metrics to use in a quality assessment.
- √ Initial identification of problem business practices.

The business review can take anywhere from several days to several weeks, depending on the complexity of the data set being considered and the number of stakeholders involved.

7.3.2 Data Validation and Quality Assessment

Once the business review has ended, the next task is to assess existing data quality and completeness. A representative sample of data is extracted from the banking system and placed in a staging area where it can be reviewed using a variety of software tools. The Web enabled data warehouse (Banking case study) Project currently uses Oracle Discoverer for data validation and quality assessment. We build a series of queries to view every data element and compare the results to the established metrics. We found the software easy to learn and inexpensive (since it is free software), and it integrates well into our Oracle environment. A number of other software tools automatically perform data validation, but we found their cost prohibitive in a higher education environment.

The data validation and quality assessment process

→ Establishes metrics to assess the validity of the data extracted from the banking system.

The metrics may include

- Acceptable date ranges for elements such as applicant birth date, opening date, and achievement date.

- Acceptable syntax and values for elements such as customer ID codes and account ID codes.

- Estimates of counts of total customers opened account and customers records.

- Business rules with which the data must comply. For example, a banking system allows withdraws if and only if the account has more than the minimum amount.

→ Systematically reviews all the data elements, considering factors such as range of values, number of null records, duplicate records, compliance with business rules, and inaccurately recorded information.

→ Provides a summary of data problems and a strategy for data cleansing.

→ Tracks data problems to their source.

→ Makes recommendations for business practice changes to improve data quality.

The data quality assessment can be a humbling and time-consuming experience for a bank institution, particularly if it's the first time they've looked critically at their own data.

Typical problems found with banking system data include

- Customers more than 2000 years old.
- Customers not yet born.
- Customer transaction that took place before the bank was established.
- Customer transaction that ended before they started normal working time.
- Duplicate records, such as – Customers with more than one ID number with different names.

For every quality problem revealed, the institution identifies the source of the error, the magnitude of the problem (does it affect one record or thousands?), corrective action, and a strategy to prevent this error from occurring

7.3.3 QUALITY ASSESSMENT.

7.3.3.1 Data Cleansing

Once data quality problems have been identified, the bank institution proceeds with data cleansing. Data cleansing goes to the source system and corrects errors and other problems identified during quality assessment. In our Project, we prefer to do as much data cleansing as possible at the source — in the banking system. This ensures that the data is correct in the system of record and means that all future users of that data will have accurate information.

However, it's not always possible or practical to perform all cleansing in the banking system. In these cases, the project also performs data cleansing in a staging area prior to its transfer into the data warehouse. Unfortunately, if data cleansing takes place after extraction of data from the source system, cleansing will have to be repeated every time the data is re-extracted.

Typical data cleansing includes

- Correcting data entry errors.
- Removing or correcting nonsensical dates.
- Deleting "garbage" records – records that don't contain valid data.
- Combining and/or deleting duplicate records.

Data cleansing is time consuming and therefore expensive. It's usually necessary to perform extensive data cleansing as part of implementing a data quality improvement process. Still, the ultimate goal is to reduce the amount of data cleansing required over time. Changing business practices, combined with regular quality reviews, should help limit the amount of routine cleansing required.

7.3.3.2. Changing Business Practices

During the preceding stages in the quality improvement process, a bank institution will have identified a number of business practices that negatively affected their data quality.

The third stage in the project's quality improvement process involves

- Implementing changes to business practices that will improve data quality and

- Adopting an ongoing process to regularly review and improve data quality.

Implementing business practice changes within a banking environment can be a challenge. It's important to involve the stakeholders who participated in the initial business review. It also helps to solicit executive support if the needed business practice changes are significant. Business practice changes that improve data quality will range from simply reviewing data collection and entry practices with responsible staff to a complete reengineering of business processes.

The types of business practice changes typically seen in our project include

- Educating data entry staff on the importance of accurate data entry.
- Updating code sets.
- Centralizing the creation of new codes
- Ensuring that data is entered in the correct location in the banking system.
- Consolidating data collection into one banking system.

- Implementing validation routines within the administrative system.

For a bank institution to have continued quality data, it must also implement a regular process for reviewing and improving data quality. A continuing data quality process should

- Appoint someone within the institution responsible for monitoring overall data quality.
- Review the quality of data at regular intervals.
- Establish a regular updating cycle for all code sets.
- Provide a procedure for addressing data quality issues as they are identified
- Establish data custodians who will be responsible for the quality of their data.
- Provide education on the importance of quality data to data custodians, creators, and data users.
- Communicate data quality improvements to the users.

7.3.4. Improved data quality.

Data quality is a data warehouse is critical, more so than in an operational system. Strategic decisions made on the basis of information from the data warehouse are likely to be more far reaching in scope and consequences.

Improved data quality:

- Boosts confidence in decision-making.
- Enables better customer service.
- Increases opportunity to add better value to the services.
- Reduces risk from disastrous decisions.
- Reduces costs, especially of marketing campaigns.
- Enhances strategic decision-making.
- Improves productivity by streamlining processes and
- Avoids compounding effects of data contamination.

Note that not all errors are preventable — simple keying errors are difficult to prevent, for example. Data will never be perfect. Upon completion of the data validation and quality assessment process, the bank institution will have a complete picture of its current data quality and will have developed some initial strategies to resolve existing problems and prevent future ones.

Chapter 8. PERFOMANCE ENHANCEMENT.

One of the main challenges within data warehousing is to recognize that fact and detail tables will grow incredibly large and to manage that growth successfully. Query performance continues to present challenges as these fact tables grow.

8.1. DATABASE DESIGN TO ENHANCE PERFORMANCE.

The techniques which improves the performance in a data warehouse are:

i. Indexing.

Indexes are database objects associated with database tables and created to speed up access to data within the tables.

Indexing is used for the following reasons:

- It is huge cost saving, greatly improving performance and scalability.
- It can replace a full table scan by a quick read of the index followed by a read of only those disk blocks that contain the rows needed.

Types of indexing:

a Bitmap Indexes

Bitmap indexes are widely used in data warehousing applications, which have large amounts of data and ad hoc queries but a low level of concurrent transactions. For such applications, bitmap indexing provides:

- Reduced response time for large classes of ad hoc queries

- A substantial reduction of space usage compared to other indexing techniques
- Dramatic performance gains even on hardware with a relatively small number of CPUs or small amount of memory
- Very efficient maintenance during parallel DML and loads

Benefits for Data Warehousing Applications

Bitmap indexes are not suitable for OLTP applications with large numbers of concurrent transactions modifying the data. These indexes are primarily intended for decision support (DSS) in data warehousing applications where users typically query the data rather than update it.

Cardinality:

The advantages of using bitmap indexes are greatest for low cardinality columns: that is, columns in which the number of distinct values is small compared to the number of rows in the table. A gender column, which only has two distinct values (male and female), is ideal for a bitmap index.

b. B-tree Indexes

A B-tree index is organized like an upside-down tree. The bottommost level of the index holds the actual data values and pointers to the

corresponding rows, much like the index in a book has a page number associated with each index entry.

c. Join Index.

A join index is built by translating restrictions on the column value of a dimension table to restrictions on a large fact table. The index is implemented using rowed or bitmap, depending on the cardinality of the indexed column. A bitmap representation, which is called Bitmap join index, is used with the low cardinality data while a row id representation is used with a high cardinality.

d. Clustered Indexes.

A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore, the table can have only one clustered index. The leaf nodes of a clustered index contain the data pages. Clustered tables combine the data segment and the index segments; the two segments are one. Data is the index and the index is the data.

Clustered tables improve performance because in one read you get the index and the data segments.

ii. Data Partitioning.

Partitioning means breaking tables down into smaller, more manageable units, thus addressing the problems of supporting large tables and indexes (which

are inherent in data warehouses). A large table is broken into many smaller physical tables or views and then they are pulled together again for query actions that access data from more than one of the tables or views.

There are two broad criteria for splitting a table into partitions:

i. Vertically.

In vertical partitioning, you break tables up on a column-by-column basis.

You can use vertical partitioning when:

a. It would improve the speed of query and update actions.

b. Users require access to specific columns. It is useful if queries are specifically on a small number of columns rather than a whole row, or you want to control visibility to sensitive data.

c. Some data is changed infrequently. You can keep the infrequently changed data in a separate partition.

d. Descriptive dimension text may be better moved away from the dimension itself. Usually, wide dimension tables are candidates for vertical partitioning.

ii. Horizontally.

Horizontal partitioning is commonly used in warehouse environments because it enables you to store a very large table in

smaller tables. It gives the database administrator control over the rows that goes into each table. The advantage, when querying data, is that full table scans are reduced.

Warehouse partitioning can be based on different criteria. Partitioning by time is most common, because most of the information you need for analysis is based on time periods. Partitioning by time is also effective for loading and archiving tasks.

Other criteria include:

- Sales region or person.
- Geography.
- Organization.
- Line of business.

Partitioning is an effective technique for storage management and improving performance. The following are benefits of partitioning

- A query needs to access only the necessary or they may explicitly request an individual partition. Queries run faster when accessing smaller amounts of data.
- An entire partition may be taken off-line for maintenance. You can separately schedule maintenance of partitions. Partitions promote concurrent maintenance operations.
- Index building is faster.
- Loading data into the data warehouse is easy and manageable.

- Data corruption affects only a single partition. Backup and recovery on a single partition reduces downtime.
- The input-output load gets balanced by mapping different partitions to various disk drives.

iii. Data clustering.

In data warehouse, many queries require sequential access of huge volumes of data. The technique of data clustering facilitates such sequential access. Clustering foster sequential prefetch of related data. You achieve data clustering by physically placing related table close to each other in storage. When you declare a cluster of tables to the DBMS, the tables are placed in neighboring areas on disks

iv. Parallel Processing.

Parallelism is the ability to apply multiple CPU and I/O resources to the execution of a single SQL command. Simply expressed, parallelism is the idea of breaking down a task so that, instead of one process doing all of the work in a query, many processes do part of the work at the same time.

Parallel processing techniques may be applied to data loading and data reorganization. Parallel processing techniques work in conjunction with data partitioning schemes. You have to assess propositions like placing two partitions on the same storage device if you need to process them in parallel.

v. Summary Levels.

The data warehouse needs to contain both detailed and summary data. Select the levels of granularity for the purpose of optimizing the input-output operations. Also, rolling summary structures are especially useful in a data warehouse. Suppose in your data warehouse you need to keep hourly data, daily data, weekly data and monthly summaries. Create mechanisms to roll the data into the next higher levels automatically with the passage of time.

Summaries are created in Oracle by using a schema object called a materized view. In a data warehouse, you can use materized views to precompute and store aggregated data such as the sum of sales. They can also be used to precompute joins with or without aggregations. A materized view eliminates the overhead that is associated with the expensive joins and aggregations for large or important class of queries. Having direct access to a summary table containing precomputed data reduces the disk I/O and CPU sort and memory swapping requirements. Materized views within the data warehouse are transparent to the end user or to the database application.

vi. Referential Integrity checks.

As you know, referential integrity constraints ensure the validity between two related tables. The referential integrity rules in the relational model govern the values of the foreign key in the child table and the primary key in the parent table. Every time a row is added or deleted, the DBMS verifies that the referential integrity is preserved. This verification ensures that parent rows are not deleted while child rows exist and that child rows are not added without parent rows. Referential integrity verification is critical in the OLTP systems, but reduces performance.

Now consider the loading of data into the data warehouse. By the time the load images are created in the staging area, the data structures have already gone through the phases of extraction, cleansing and transformation. The data ready to be loaded has already been verified for correctness as far as parent and child rows are concerned. Therefore, there is no further need for referential integrity verification while loading the data. Turning off referential integrity verification produces significant performance gains.

vii. Data arrays.

What are data arrays? Suppose in a financial data mart you need to keep monthly balances of individual line accounts. In a normalized structure, the monthly balances for a year will be found in twelve separate table rows. Assume that in many queries the users request for the balances for all the months together. How can you improve the performance? You can create a

data array or repeating group with twelve slots, each to contain the balance for one month.

Although creating arrays is a clear violation of normalization principles, this yields tremendous performance improvement.

8.2 Environment Performance.

As mentioned earlier, system performance is a critical success factor for large data warehouse initiatives. Aside from having the ability to meet current processing needs, the data warehouse environment should also be scalable to meet future growth needs.

8.2.1 Scalability for Data Warehouse Solutions

A truly "scalable" system is one that can consistently respond to increased demand with the addition of CPU, memory, or storage resources. BI organizations need to anticipate the environment growth before it happens, and the data warehouse must reside on a scalable architecture that will not limit any foreseeable increases in the amount of data or number of users.

To ensure reliable performance levels across the BI environment, it is essential that scalability be addressed for all of the functional components.

8.2.2 Planning for Performance

Planning a data warehouse environment should take into account the performance requirements for information movement, usage, and growth within an organization. Below are some functional areas specific to BI initiatives where performance requirements should be addressed in the environment plan:

8.2.2.1 Extraction, Transformation, and Loading

The core of every data warehouse initiative is the extraction, transformation, and loading (ETL) component. This is an extensive piece of the system, and it involves the collection, cleansing, validation, organization, and storage of the subject matter data.

Regardless of whether the data comes from flat files or direct transfers from other systems, the ETL process must handle the data transformation and loading functions for the data warehouse. Data cleansing and translation tasks make up the transformation function, and this piece usually consumes the majority of ETL processing time. Once transformation is complete, the ETL process then works with the database platform to load the data. The loading process can drain system resources, especially when database indexes and constraints add additional overhead. In order to achieve acceptable ETL processing durations, BI environment planners need to factor in the transformation and loading processes when preparing for a large-scale environment. When time frames are limited, such as less than 8 hours, then more CPU's become critical to ETL workloads.

8.2.2.2 Standard Query Processing

When the data warehouse becomes available to all users, it is critical that the system be able to handle the total query demand, regardless of the number or complexity of the queries. Creating a well designed data warehouse or "reporting database" is essential to meeting this requirement. Additionally, hardware configurations should coincide with the characteristics of the system usage. If the data warehouse system is expected to perform a large number of CPU intensive tasks (Such as heavy computations), processor resources must be sufficient to maintain reasonable process durations.

Typically, data warehouses handle both standardized and ad-hoc queries. Standard reports may be scheduled to process nightly, and the system must be able to have the results ready within the nightly batch window. Although the standard queries may be the most commonly run query type, ad-hoc queries are what really test the environment. BI reporting tools allow end-users to build custom queries on demand. Because of the random nature of ad-hoc queries, no amount of database tweaking can guarantee that every query will perform within the required timeframe. CPU, I/O, and memory resources should be planned according to anticipated query demand.

8.2.2.3 Analytical Processing

For advanced OLAP applications, the evolution of BI tools and technologies has provided new structures or "analytical databases" to enable extensive analysis across enterprise data. Reporting databases may provide the granular data for basic analysis and reporting, however veteran OLAP users learned early on that optimized data structures would be needed to facilitate quick analytical queries.Relational aggregate tables were created to store summarizations of the granular reporting data. These tables could then be queried to quickly return pre-calculated aggregations.

For more advanced analytical processing, multidimensional cubes emerged as optimized structures residing outside of the relational database. Cubes provide customized aggregations across multiple business dimensions such as customer and geography. Because of their multidimensional structure, cubes enable analytical queries across business dimensions that are cumbersome or impossible to perform with relational databases. With the emergence of these analytical database structures, execution times for many analysis queries improved dramatically. This advancement led to the separation of reporting and analysis databases. Without analysis structures in place, basic analytical queries such as year-to-date comparisons across multiple years could take several hours or longer for large data volumes.

Aggregate tables and multidimensional cubes are created through the analytical processing of granular data. Summaries of the data called "aggregations" are pre-calculated to enable almost instant response to analytical queries such as those associated with weekly transactions totals.

Although query execution time has improved dramatically with these structures, the performance gain comes at a price. Additional CPU and storage resources are required to provide the processing and storage of aggregations before the analysis structure can be queried. BI environment planning should take into account the extent of these additional structures within the environment. On systems with enormous amounts of transactional data, processing the analysis structures can consume a great deal of time and storage resources depending on the data volume, update frequency, and level of aggregation. Nevertheless, analysis databases have brought on new levels of analytical query performance, and BI environments must facilitate the creation and management of these structures.

8.2.2.4 Data Mining

Using information collected in their reporting and analysis databases, BI organizations make use of data mining to uncover and identify patterns in their business data. Statistical analysis is performed on data collections to help decision makers predict certain business conditions as well as conduct what-if business scenarios. The results of data mining initiatives help analysts to identify data relationships, efficiency issues, and areas for potential revenue growth.

Over the past several years, data mining tools and techniques have become more accessible to all decision makers. Applications now embed data mining operations into their functionality so that the actual analytical complexity is hidden from the user. Regardless of their ease-of-use, data mining tools make use of formal analysis methods to explore the business data.

Chapter 9 IMPLEMENTATION

9.1 BACKEND – ETL DEVELOPMENT

The ETL (Extraction, Transformation, Loading) process typically takes the longest to develop, and this can easily take up to 60% of the data warehouse implementation cycle or longer. The reason for this is that it takes time to get the source data, understand the necessary columns, understand the business rules, and understand the logical and physical data models.

Tool Used for ETL

We have used Oracle9i Warehouse Builder for developing and implementing ETL process our Data Warehouse.

Oracle9*i* Warehouse Builder is a business intelligence tool that provides an integrated solution for designing and deploying enterprise data warehouses, data marts, and business intelligence applications. It solves the complex problem of data integration between dispersed data sources and targets. In addition, Warehouse Builder provides all the necessary functionality to maintain the life cycle of the system you develop.

Warehouse Builder is a complete design and implementation tool for building and managing data warehouses and business intelligence systems.

It combines the key components of both an extraction, transformation, and loading (ETL) tool and a design tool into one product. In addition, Warehouse Builder leverages Oracle database technology. It is the central point of integration of the Oracle Business Intelligence tools suite and provides integration with ad-hoc query tools as well as OLAP and relational database features.

The architecture of Warehouse Builder is comprised of two components, the design environment and the runtime environment. Each of these components handles a different aspect of the system. The design environment manages the metadata, while the runtime environment handles the physical data.

Creating a business intelligence application is a complex process. It involves various steps and phases, which may span a great number of systems, resources, and functional areas. Warehouse Builder reduces this complexity as it enables you to manage these tasks from a single interface while ensuring scalability, reliability and flexibility by leveraging the latest Oracle database technology.

Key Warehouse Builder functions include:

- Importing source data definitions.
- Designing and creating target database schema.
- Defining data movement and transformation between sources and targets.
- Assigning dependencies between ETL processes.
- Managing and updating source definitions.
- Deploying, upgrading, and managing target schemas.
- Designing and creating an ad-hoc query tool environment.
- Designing and creating an OLAP environment

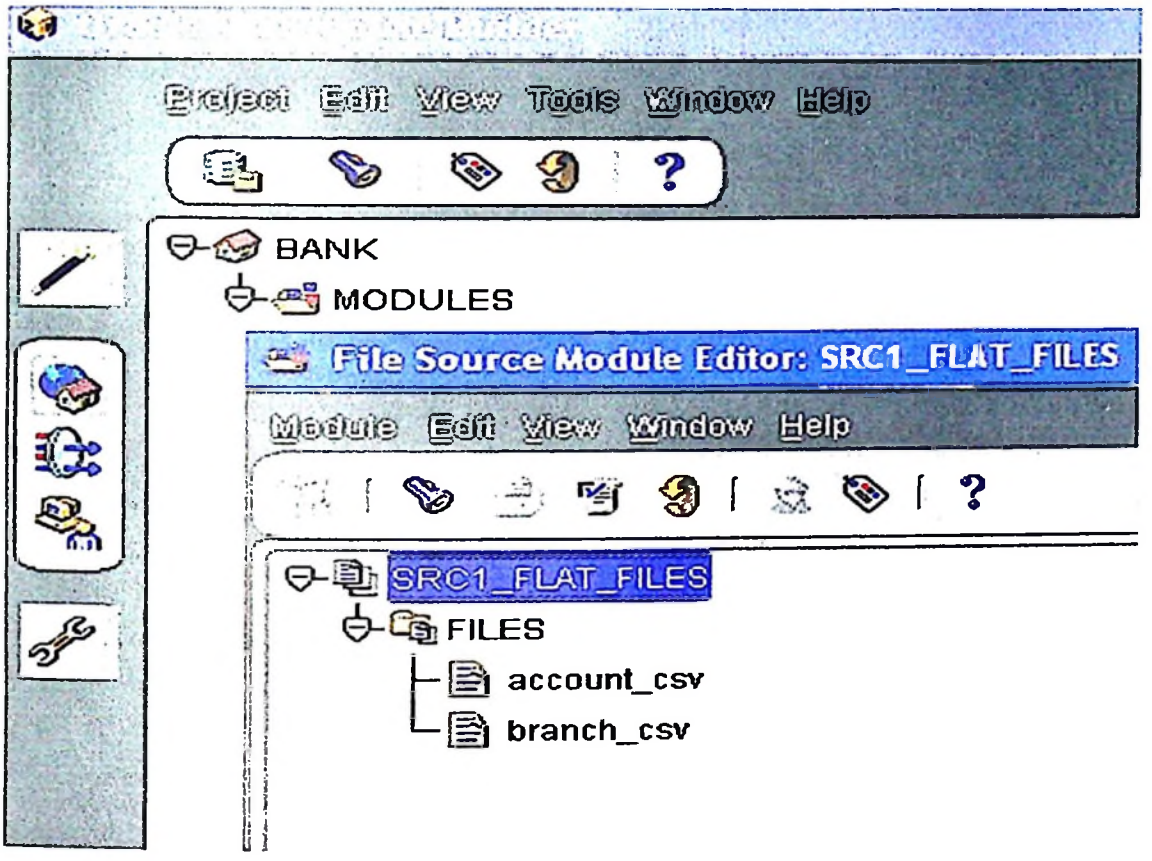
The development of ETL process of our project includes the following steps:

1.Source to Staging Area

i. Defining Source Metadata

In Warehouse Builder, you create source modules to store definitions from different source systems. The source modules represent the source database schemas and flat files structures in the format expected by Oracle Warehouse Builder. These definitions can be created using the Warehouse Builder wizards or by importing them from external sources.

For our project we have used two sources: Oracle Database (OLTP) and flat files and defined two source modules: ***SRC1-FLAT-FILES*** and ***SRC2_BANK_OLTP*** shown below:



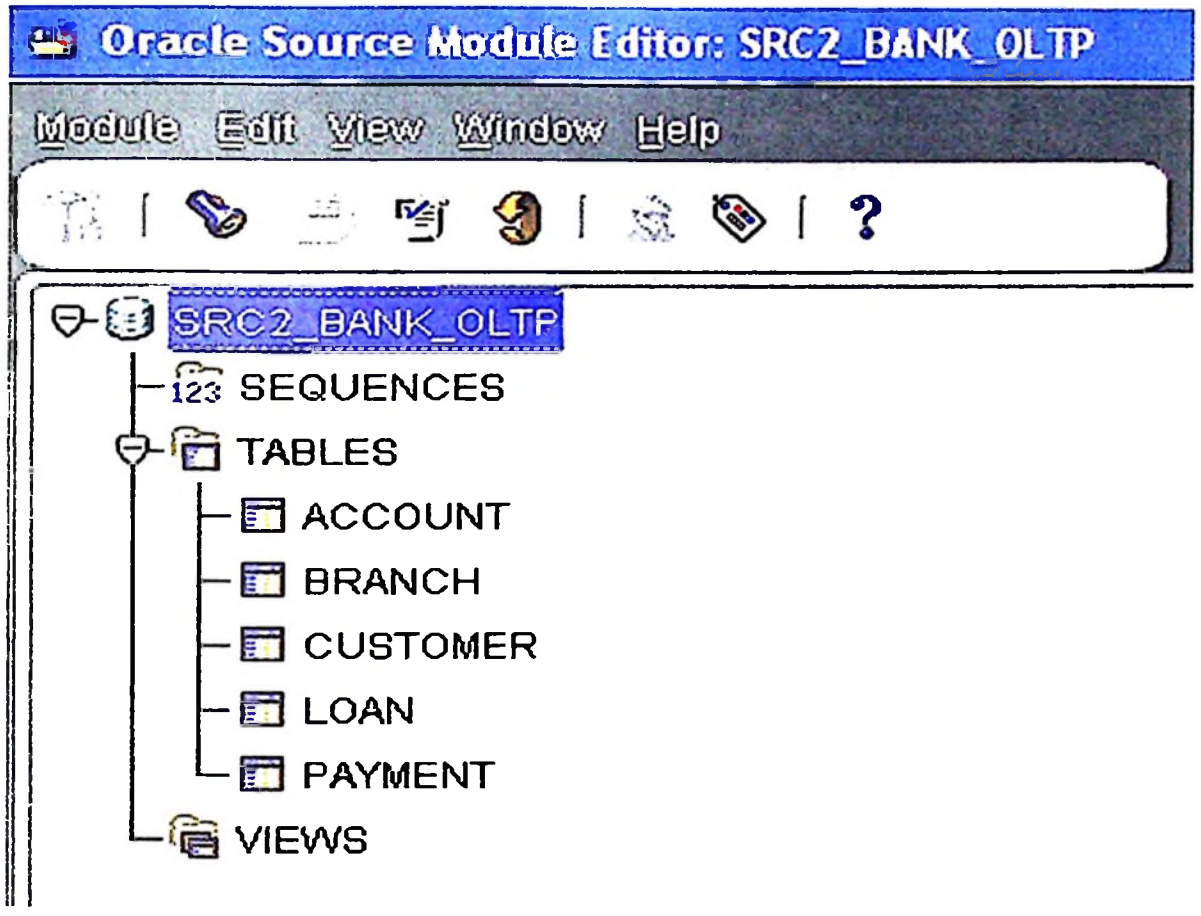


Fig.9.1

ii. Defining Staging Area Metadata

Define targets by creating a model of the target warehouse you intend to create upon deployment. Use Oracle database warehouse modules to store the metadata definitions. You can create and design new objects, or you can also import definitions from source data locations.

Warehouse Builder stores the definitions for the target schema in warehouse modules.

We have defined a staging area target schema called ***TRG1_STAGING_AREA.***

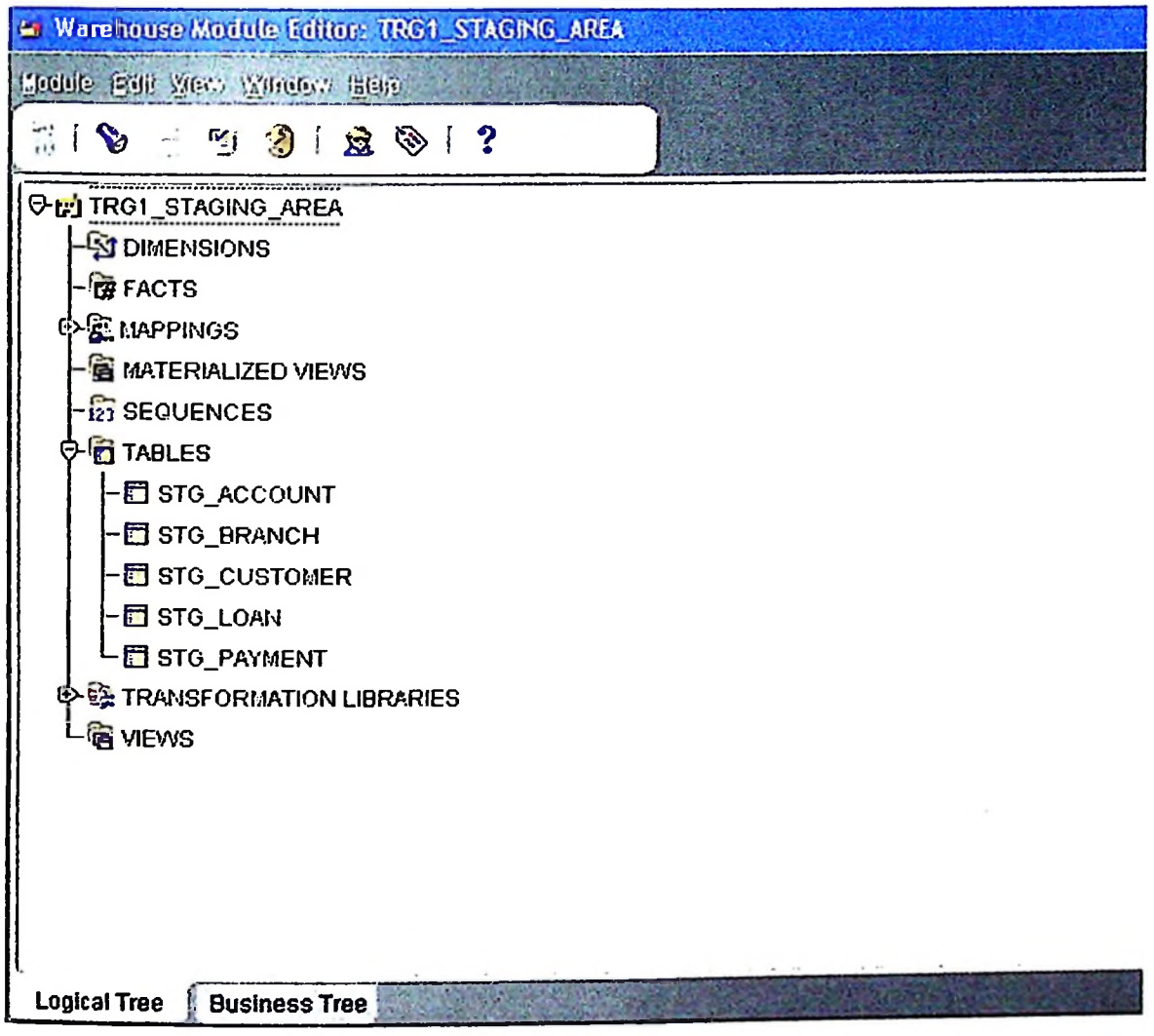


Fig.9.2

iii. **Map source to Staging Area.**

After you create and import data object definitions in Staging Area Module, you can define extraction, transformation, and loading (ETL) operations that move data from sources to Staging Area Module. In Warehouse Builder, you design these operations in a mapping.

Mappings describe a series of operations that extract data from sources, transform it, and load it into targets. They provide a visual representation of the flow of the data and the operations performed on the data.

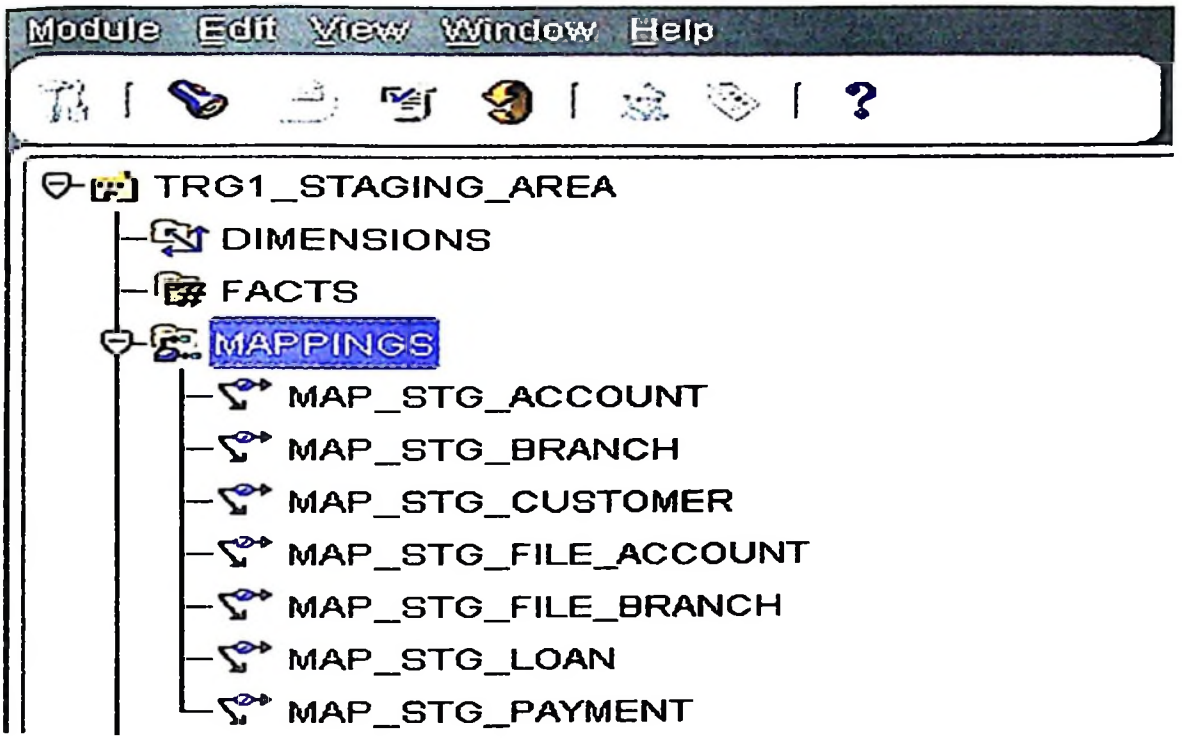


Fig.9.3

Defining the Mapping

When you define a mapping, you create a container that holds the operators defining the ETL logic.

To define a mapping, complete the following steps:

- Creating a Mapping
- Adding Operators
- Editing Operators

- Connecting Operators
- Setting Operator Properties
- Configuring Mappings References
- Reconciling Operators and Repository Objects

Mapping from Account (in the Source Module) to Stg_Account (in the Staging Area Module).

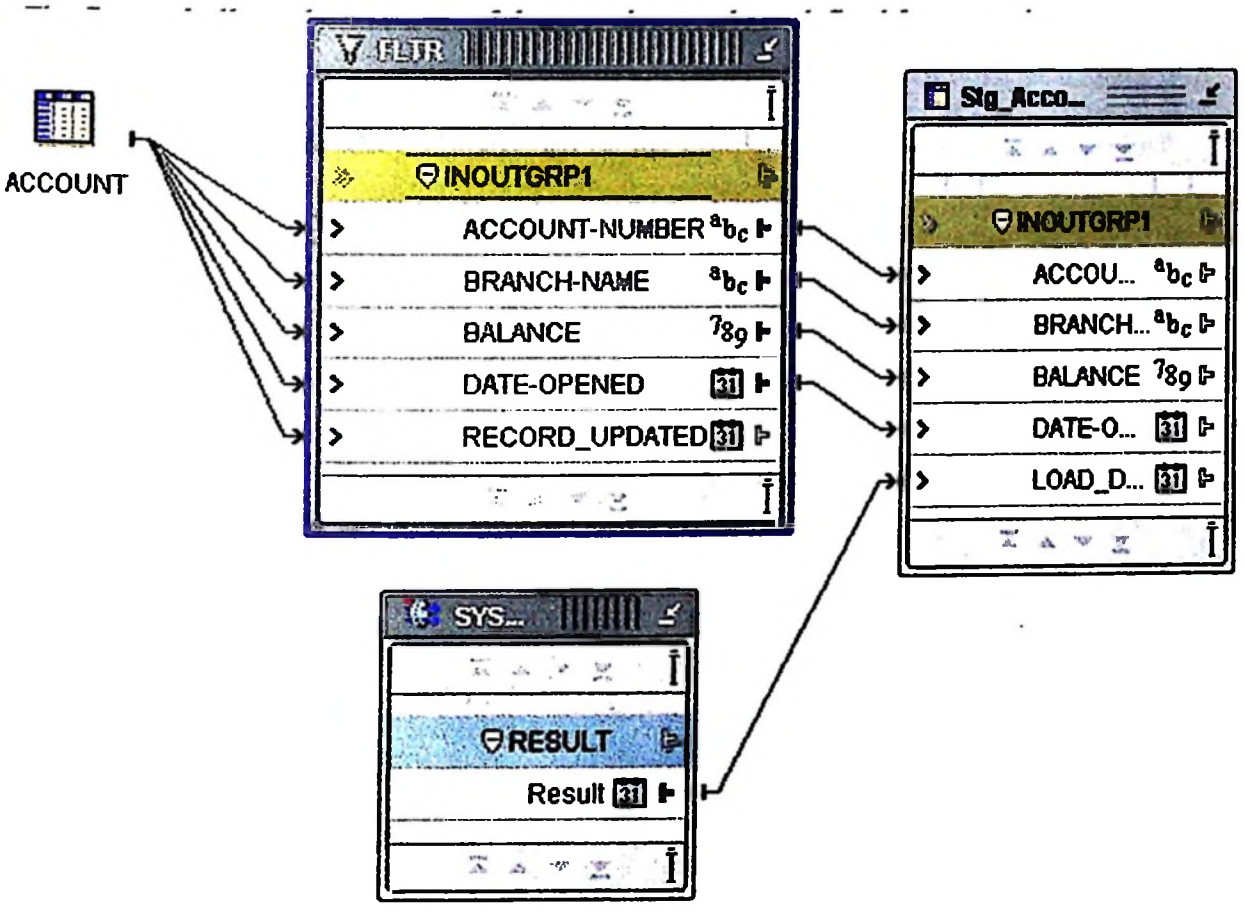


Fig.9.4

Mapping from Branch (in the Source Module) to Stg_Branch (in the Staging Area Module).

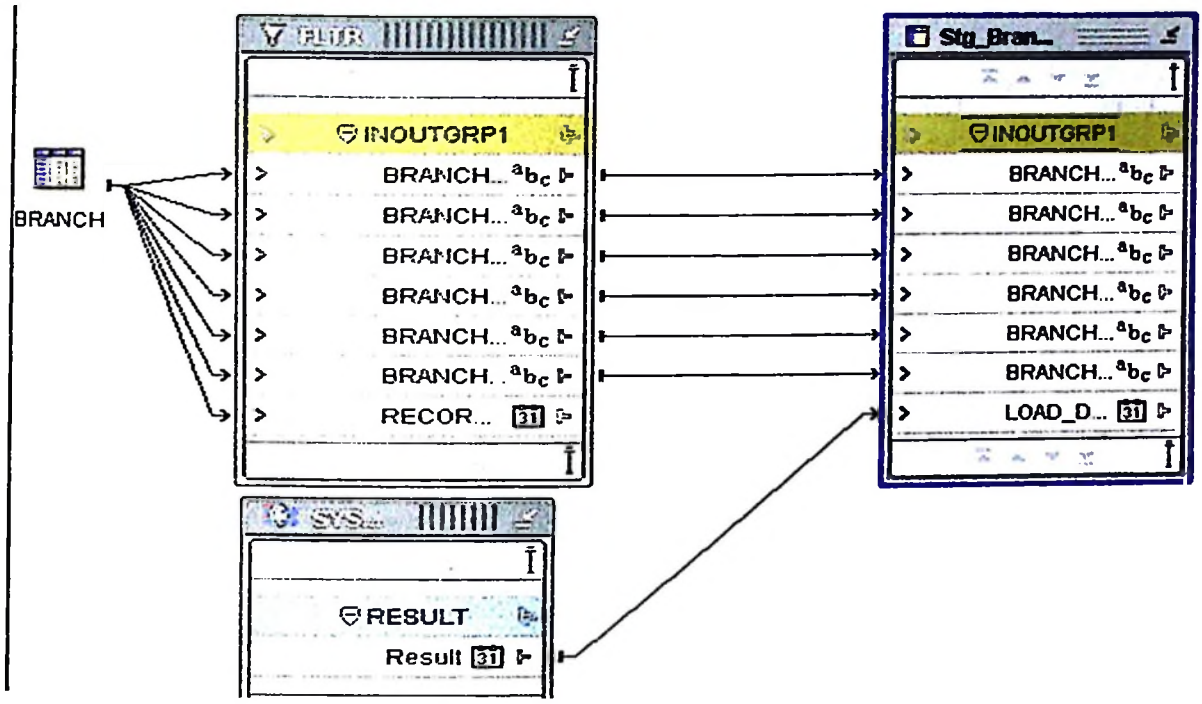


Fig.9.5

iv. Validating and Generating

After you define source and target modules and define the ETL logic you can validate and generate the code to check for errors prior to deployment.

Validation

You can validate all objects and modules to ensure that no definitions are invalid or incomplete. If you find invalid definitions, use the editors to correct the problems. If all definitions are valid, you are ready to configure the modules and their components for deployment.

Generation

You can generate and view scripts prior to deployment. The type of code generated depends on the configuration parameters defined for the type of target. When you generate the code, you can also save it if you want to deploy it from outside Warehouse Builder.

v. **Deploying and Executing**

Deploy the design to the target environment after the target model is complete. You must install a runtime repository and define a Runtime Repository Connection before deployment. Warehouse Builder provides two methods for deployment. You can deploy directly from the main console or you can use the Deployment Manager. You can then execute mappings and process flows from the Deployment Manager. You can also validate and generate objects prior to deployment.

2. Staging to Warehouse

i. Define enterprise model (warehouse) metadata

Define Enterprise Warehouse Metadata by creating a model of the target warehouse you intend to create upon deployment. Use Oracle database warehouse modules to store the metadata definitions. You can create and design new objects, or you can also import definitions from Staging Area data locations.

Warehouse Builder stores the definitions for the target schema in warehouse modules. We have defined a Enterprise target schema called ***TRG2_STAGING_AREA.***

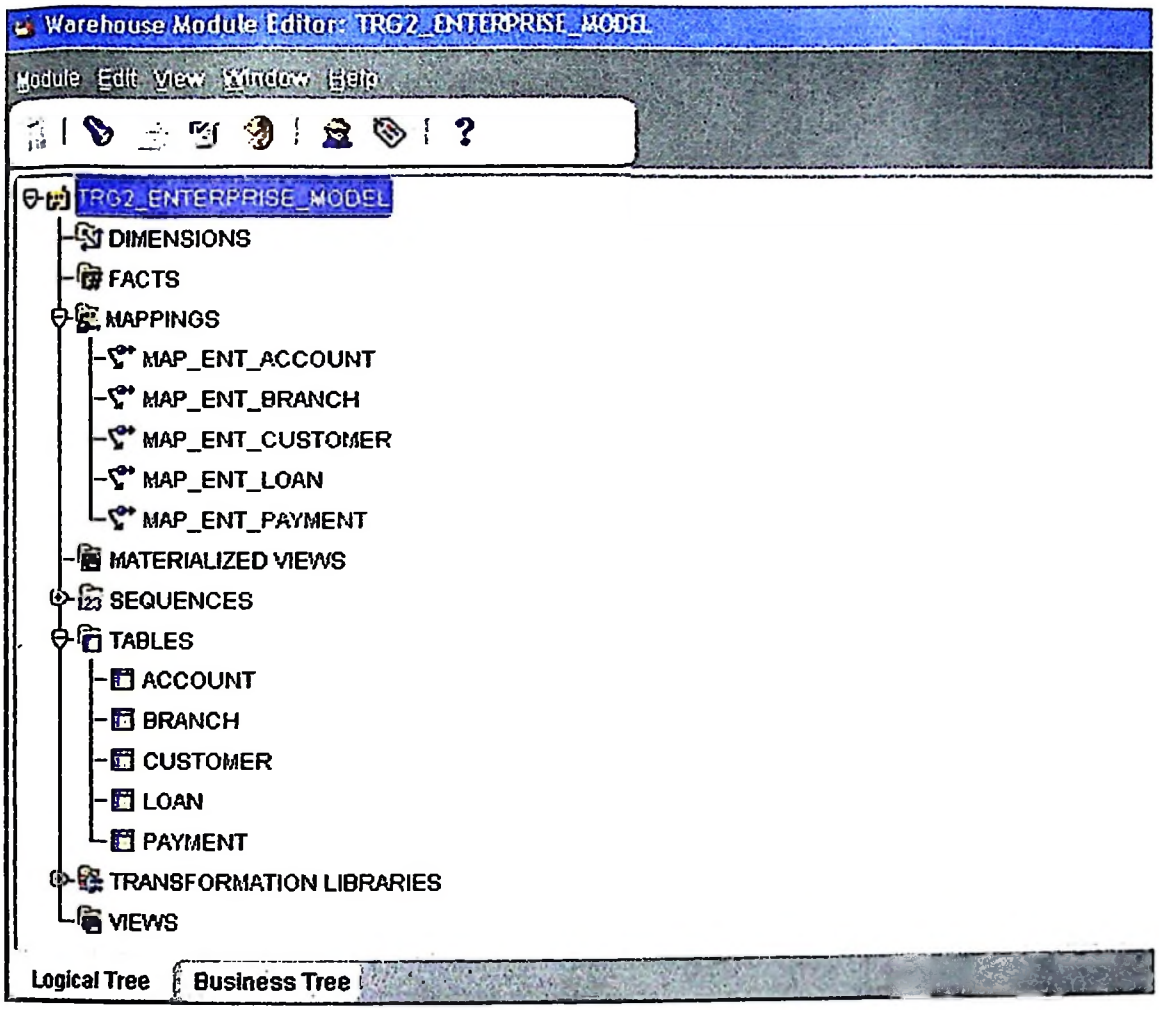

Warehouse Module Editor: TRG2_ENTERPRISE_MODEL
Module Edit View Window Help
TRG2_ENTERPRISE_MODEL
DIMENSIONS
FACTS
MAPPINGS
MAP_ENT_ACCOUNT
MAP_ENT_BRANCH
MAP_ENT_CUSTOMER
MAP_ENT_LOAN
MAP_ENT_PAYMENT
MATERIALIZED VIEWS
SEQUENCES
TABLES
ACCOUNT
BRANCH
CUSTOMER
LOAN
PAYMENT
TRANSFORMATION LIBRARIES
VIEWS
Logical Tree
Business Tree

Fig.9.6

ii. **Map staging area to enterprise model**

After you create and import data object definitions in Enterprise Warehouse, you can define extraction, transformation, and loading (ETL) operations that move data from Staging Area Module (***TRG1_STAGING_AREA***) to Enterprise Warehouse Module (***TRG2_ENTERPRISE_MODEL***).

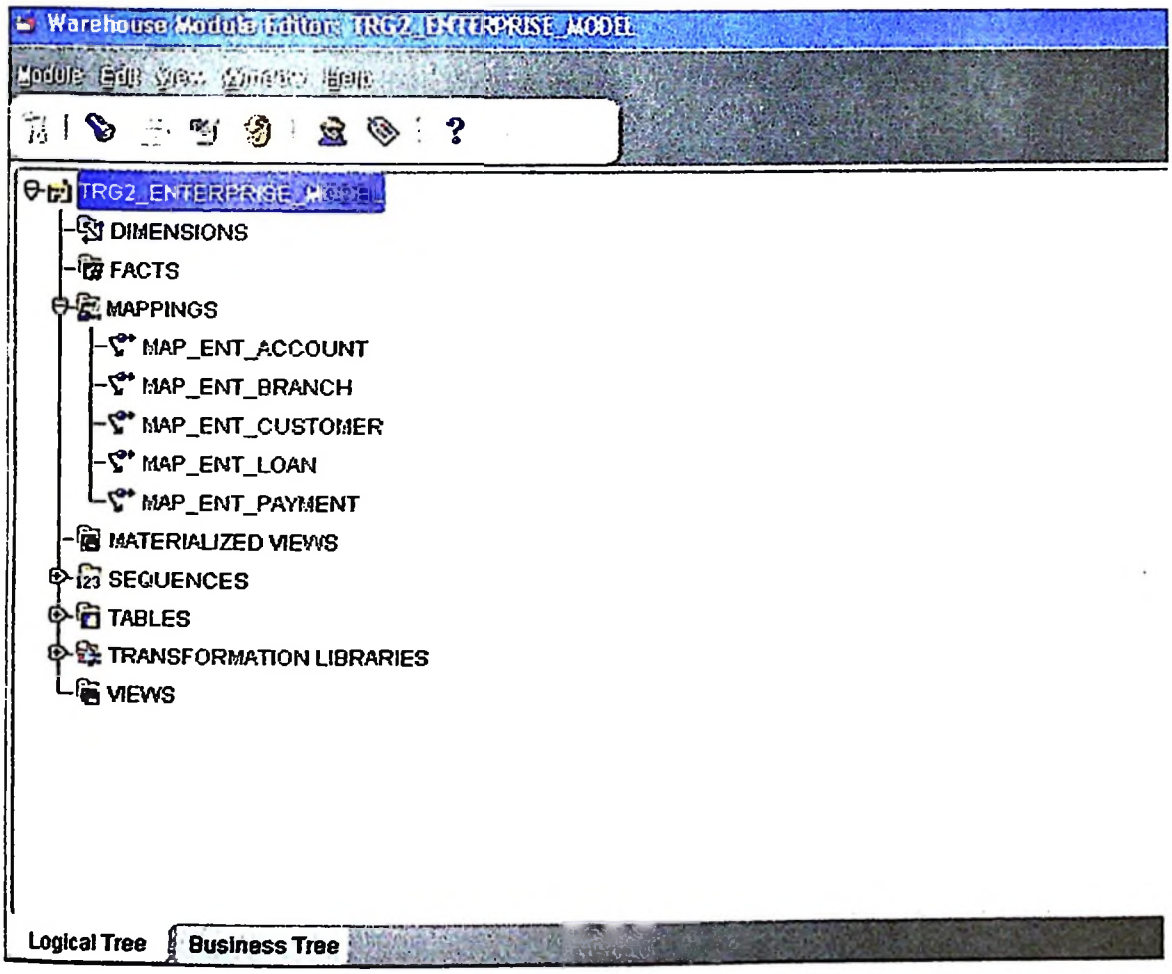


Fig.9.7

This process is similar to that of creating mappings from source to Staging Area but with added features as shown below.

Mapping from Stg_Account (in the Staging Area Module) to Account (in the Enterprise Warehouse Module).

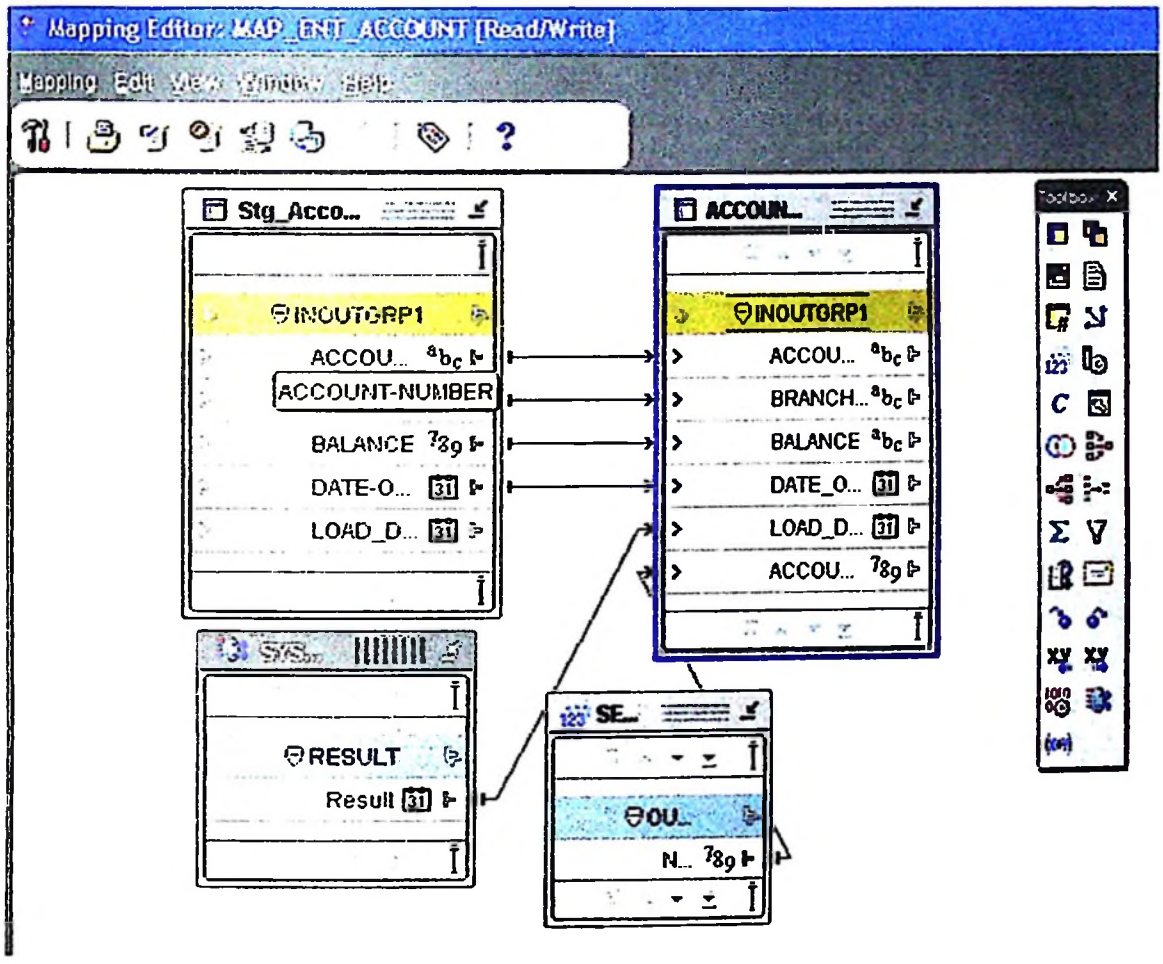


Fig.9.8

Mapping from Stg_Branch (in the Staging Area Module) to Branch (in the Enterprise Warehouse Module).

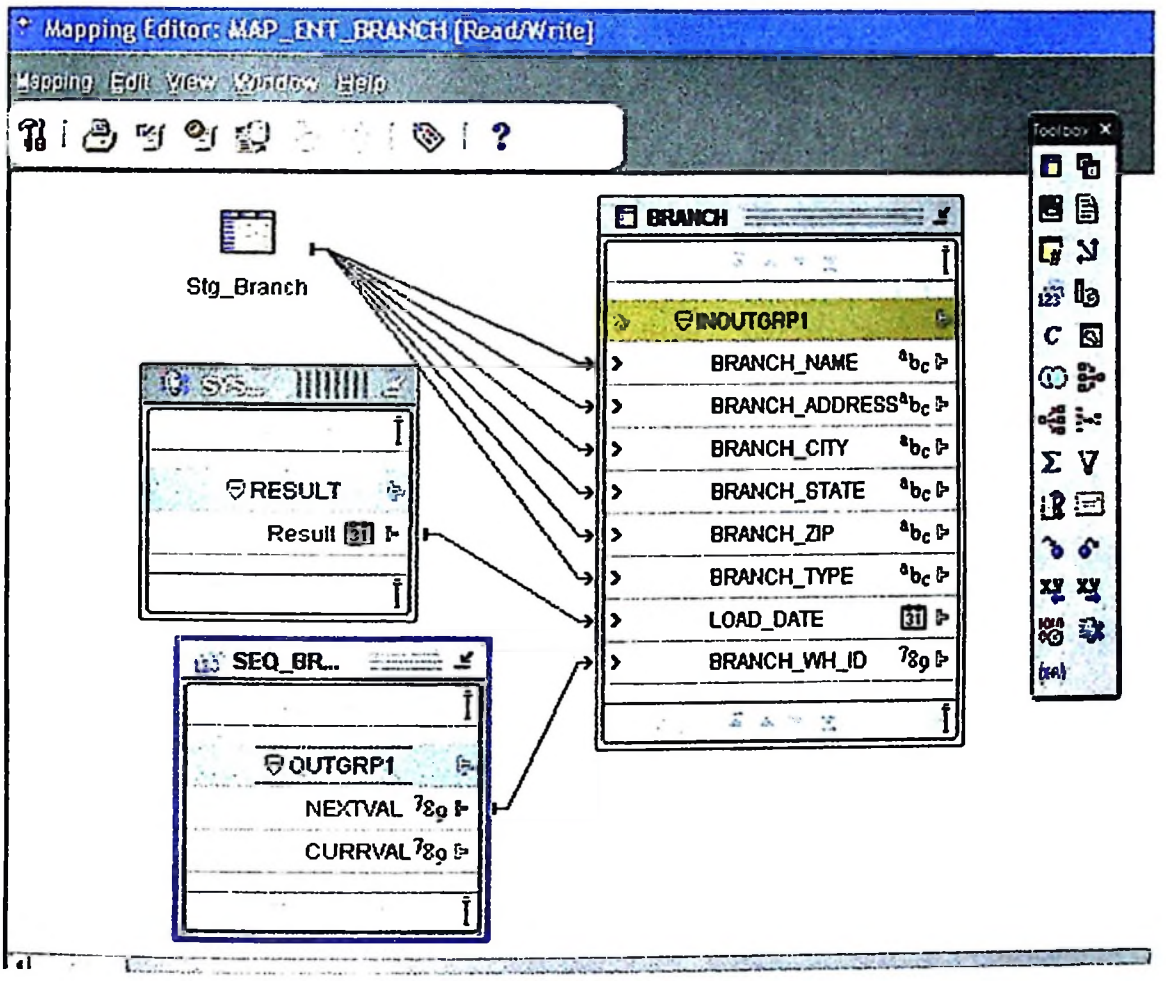


Fig.9.10

iv. Validating and Generating

After you define staging area module and target enterprise modules and define the ETL logic you can validate and generate the code to check for errors prior to deployment.

v. Deploying and Executing

Deploy the design to the target environment after the target model is complete. You must install a runtime repository and define a Runtime Repository Connection before deployment. Warehouse Builder provides two methods for deployment. You can deploy directly from the main console or you can use the Deployment Manager. You can then execute mappings and process flows from the Deployment Manager. You can also validate and generate objects prior to deployment.

9.2 OLAP IMPLEMENTATION

Online Analytical Processing (OLAP) means analyzing large quantities of data in real time. Unlike Online Transaction Processing (OLTP), where typical operations read and modify individual and small numbers of records, OLAP deals with data in bulk, and operations are generally read-only. The term 'online' implies that even though huge quantities of data are involved – typically many millions of records, occupying several gigabytes – the system must respond to queries fast enough to allow an interactive exploration of the data.

We have developed and implemented OLAP using Mondrian

MONDRIAN

Mondrian is an OLAP engine written in Java. It executes queries written in the MDX language, reading data from a relational database (RDBMS), and presents the results in a multidimensional format via a Java API.

9.2.1 Architecture

A Mondrian OLAP System consists of four layers, working from the eyes of the end-user to the bowels of the data center, these are as follows: the presentation layer, the calculation layer, the aggregation layer, and the storage layer.

The **presentation layer** determines what the end-user sees on his her monitor, and how he or she can interact to ask new questions. There are many ways to present multidimensional datasets, including pivot tables (an interactive version of the table shown above), pie, line and bar charts, and advanced visualization tools such as clickable maps and dynamic graphics. These might be written in Swing or JSP, charts rendered in JPEG or GIF format, or transmitted to a remote application via XML. What all of these forms of presentation have in common is the multidimensional 'grammar' of dimensions, measures and cells in which the presentation layer asks the question is asked, and OLAP server return the answer.

The second layer is the **calculation layer**. The calculation layer parses, validates and executes MDX queries. A query is evaluated in multiple phases. The axes are computed first, then the values of the cells within the axes. For efficiency, the calculation layer sends cell-requests to the aggregation layer in batches. A query transformer allows the application to manipulate existing queries, rather than building an MDX statement from scratch for each request. And metadata describes the dimensional model, and how it maps onto the relational model.

9.2.1 Architecture

A Mondrian OLAP System consists of four layers, working from the eyes of the end-user to the bowels of the data center, these are as follows: the presentation layer, the calculation layer, the aggregation layer, and the storage layer.

The **presentation layer** determines what the end-user sees on his her monitor, and how he or she can interact to ask new questions. There are many ways to present multidimensional datasets, including pivot tables (an interactive version of the table shown above), pie, line and bar charts, and advanced visualization tools such as clickable maps and dynamic graphics. These might be written in Swing or JSP, charts rendered in JPEG or GIF format, or transmitted to a remote application via XML. What all of these forms of presentation have in common is the multidimensional 'grammar' of dimensions, measures and cells in which the presentation layer asks the question is asked, and OLAP server return the answer.

The second layer is the **calculation layer**. The calculation layer parses, validates and executes MDX queries. A query is evaluated in multiple phases. The axes are computed first, then the values of the cells within the axes. For efficiency, the calculation layer sends cell-requests to the aggregation layer in batches. A query transformer allows the application to manipulate existing queries, rather than building an MDX statement from scratch for each request. And metadata describes the dimensional model, and how it maps onto the relational model.

The third layer is the **aggregation layer.** An aggregation is a set of measure values ('cells') in memory, qualified by a set of dimension column values. The calculation layer sends requests for sets of cells. If the requested cells are not in the cache, or derivable by rolling up an aggregation in the cache, the aggregation manager and sends a request to the storage layer.

The **storage layer** is an RDBMS. It is responsible for providing aggregated cell data, and members from dimension tables.

These components can all exist on the same machine, or can be distributed between machines. Layers 2 and 3, which comprise the Mondrian server, must be on the same machine. The storage layer could be on another machine, accessed via remote JDBC connection. In a multi-user system, the presentation layer would exist on each end-user's machine (except in the case of JSP pages generated on the server).

Mondrian's aggregation strategy is as follows:

- Fact data is stored in the RDBMS. Why develop a storage manager when the RDBMS already has one?
- Read aggregate data into the cache by submitting **group by** queries. Again, why develop an aggregator when the RDBMS has one?

- The RDBMS supports materialized views, and the database administrator chooses to create materialized views for particular aggregations, then Mondrian will use them implicitly. Ideally, Mondrian's aggregation manager should be aware that these materialized views exist and that those particular aggregations are cheap to compute. If should even offer tuning suggestions to the database administrator.

API

Mondrian provides an API for client applications to execute queries. Mondrian uses a language called MDX ('Multi-Dimensional eXpressions') to specify queries.

A *connection* is created via DriverManger, in a similar way to JDBC. A *query* is analogous to a JDBC *Statement*, and is created by parsing an MDX string. A *Result* is analogous to a JDBC *ResultSet*; since we are dealing with multi-dimensional data, it consists axes and cells, rather than rows and columns. Since OLAP is intended for data exploration, you can modify the parse tree contained in a query by operations drilldown and sort then execute the query.

The API also presents the database schema as a set of objects: Schema, Cube, Dimension, Hierarchy, Level, and Member.

MDX

MDX, an acronym for Multidimensional Expressions, is a syntax that supports the definition and manipulation of multidimensional objects and data. MDX is similar in many ways to the Structured Query Language (SQL) syntax, but is not an extension of the SQL language; in fact, some of the functionality that is supplied by MDX can be supplied, although not as efficiently or intuitively, by SQL.

As with an SQL query, each MDX query requires a data request (the SELECT clause), a starting point (the FROM clause), and a filter (the WHERE clause). These and other keywords provide the tools used to extract specific portions of data from a cube for analysis. MDX also supplies a robust set of functions for the manipulation of retrieved data, as well as the ability to extend MDX with user-defined functions.

MDX, like SQL, provides data definition language (DDL) syntax for managing data structures. There are MDX commands for creating (and deleting) cubes, dimensions, measures, and their subordinate objects.

MDX language. Mondrian's extensions to MDX are parameters and modified builtin functions.

Parameters

A parameter is a named variable embedded in an MDX query. Every parameter has a default value, but you can supply a different value when you run the query.

Parameters are declared and used by using a special function `Parameter`:

Parameter(<*name*>, <*type*>, <*defaultValue*>[, <*description*>])

The arguments of `Parameter` are as follows:

- *name* is a string constant. It must be unique within the query.
- *type* is either NUMERIC, STRING, or the name of a hierarchy.
- *defaultValue* is an expression. The expression's type must be consistent with the `type` parameter; if `type` was a hierarchy, the expression must be a member of that hierarchy.
- *description* is an optional string constant.

If you want to use a parameter more than once in a query, use the `ParamRef` function:

ParamRef(<*name*>)

The `name` argument must be the name of a parameter declared elsewhere in the query by calling the `Parameter` function.

Built-in functions

The `StrToSet()` and `StrToTuple()` functions take an optional parameter not present in the standard MDX versions of these functions, describing the hierarchy the result will belong to:

StrToSet(<String Expression>[, <Hierarchy>])

StrToTuple(<String Expression>[, <Hierarchy>])

The MDX Query

A basic Multidimensional Expressions (MDX) query is structured in a fashion similar to the following example:

SELECT [<axis_specification>

[, <axis_specification>...]]

FROM [<cube_specification>]

[WHERE [<slicer_specification>]]

MDX Syntax - SELECT Statement

In MDX, the SELECT statement is used to specify a dataset containing a subset of multidimensional data.

To specify a dataset, an MDX query must contain information about:

- The number of axes. You can specify up to 128 axes in an MDX query.
- The members from each dimension to include on each axis of the MDX query.
- The name of the cube that sets the context of the MDX query.
- The members from a slicer dimension on which data is sliced for members from the axis dimensions.

Axis and Slicer Dimensions

When formulating a Multidimensional Expressions (MDX) query, an application typically looks at the cubes and divides the set of dimensions into two subsets:

- Axis dimensions, for which data is retrieved for multiple members.
- Slicer dimensions, for which data is retrieved for a single member.

Because axis and slicer dimensions can be constructed from multiple dimensions of the cube to be queried, these terms are used to differentiate the dimensions employed by the cube to be queried from the dimensions created in the cube returned by an MDX query.

Specifying the Contents of an Axis Dimension

Axis dimensions determine the edges of a multidimensional result set. Multidimensional Expressions (MDX) uses the SELECT clause to specify axis dimensions by assigning a set to a particular axis.

The following information describes how this assignment is handled in MDX.

In the following syntax example, each <axis_specification> value defines one axis dimension. The number of axes in the dataset is equal to the number of <axis_specification> values in the Multidimensional Expressions (MDX) query.

An MDX query can support up to 128 specified axes, but very few MDX queries will use more than 5 axes.

The breakdown of the <axis_specification> syntax is:

<axis_specification>::= <set> ON <axis_name>

<axis_name> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAPTERS | AXIS(<index>)

Specifying the Contents of a Slicer Dimension

Slicer dimensions filter multidimensional data. You can use them to limit the data returned by including them in the WHERE clause of a Multidimensional Expressions (MDX) query.

Dimensions that are not explicitly assigned to an axis are assumed to be slicer dimensions and filter with their default members. The default member of a dimension can be explicitly specified in its **Default Member** property in Analysis Manager. This property is equivalent to the **DefaultMember** property in Decision Support Objects (DSO). If no default member is explicitly specified, the default member is the All member if an (All) level exists, or else an arbitrary member of the highest level. (The name of the All member is not necessarily All.)

Slicer dimensions can also be specified explicitly by using the WHERE clause of the MDX syntax. The breakdown of the WHERE clause syntax is:

[WHERE [<slicer_specification>]]

Establishing Cube Context

To establish cube context, indicate the cube on which you want the Multidimensional Expressions (MDX) query to run. The FROM clause in an MDX query determines the cube context. The following syntax indicates which cube supplies the context for the MDX query:

FROM «cube_specification»

The «cube_specification» is completed with the name of a single cube.

MDX Query Example

The following MDX query example is used to discuss the various parts of basic MDX SELECT statement syntax:

```
SELECT
  { [Measures].[balance], [Measures].[Account Number] } ON COLUMNS,
  { [Time].[1997], [Time].[1998] } ON ROWS
FROM Balances
WHERE ( [branch].[India].[Hyderabad] )
```

9.2.2 Designing Mondrian Schema

A schema defines a multi-dimensional database. It contains a logical model, consisting of cubes, hierarchies, and members, and a mapping of this model onto a physical model.

The logical model consists of the constructs used to write queries in MDX language: cubes, dimensions, hierarchies, levels, and members.

The physical model is the source of the data, which is presented through the logical model. It is typically a star schema, which is a set of tables in a relational database.

Mondrian schemas are represented in an XML file.

To create a schema, we edit a schema XML file in a text editor.

Logical model

The most important components of a schema are cubes, measures, and dimensions.

- A *cube* is a collection of dimensions and measures in a particular subject area.
- A *measure* is a quantity that you are interested in measuring, for example, unit sales of a product, or cost price of inventory items.
- A *dimension* is an attribute, or set of attributes, by which you can divide measures into sub-categories. For example, you might wish to break down product sales by their color, the gender of the customer, and the store in which the product was sold; color, gender, and store are all dimensions.

The logical model consists of the constructs used to write queries in MDX language: cubes, dimensions, hierarchies, levels, and members.

The physical model is the source of the data, which is presented through the logical model. It is typically a star schema, which is a set of tables in a relational database.

Mondrian schemas are represented in an XML file.

To create a schema, we edit a schema XML file in a text editor.

Logical model

The most important components of a schema are cubes, measures, and dimensions.

- A *cube* is a collection of dimensions and measures in a particular subject area.
- A *measure* is a quantity that you are interested in measuring, for example, unit sales of a product, or cost price of inventory items.
- A *dimension* is an attribute, or set of attributes, by which you can divide measures into sub-categories. For example, you might wish to break down product sales by their color, the gender of the customer, and the store in which the product was sold; color, gender, and store are all dimensions.

Cube

A cube is little more than a named collection of measures and dimensions. The one thing the measures and dimensions have in common is the fact table.As we shall see, the fact table holds the columns from which measures are calculated, and contains references to the tables which hold the dimensions.

```
<Cube name="nameof the cube">
  <Table name="Fact table name"/>
  ...
</Cube>
```

The fact table is defined using the Table element. If the fact table is not in the default schema, you can provide an explicit schema using the "schema" attribute, for example

```
<Table schema="schema name" name=" Fact table name "/>
```

Measures

The cube can define more than one measure.

```
<Measure name="measure attribute" column="column name"
    aggregator="sum" formatString="#,###"/>
```

Dimensions

The dimension consists of a single or more hierarchies. Example

```
<Dimension name="nameofdimension" foreignKey="theforeignkey attribute">
  <Hierarchy hasAll="true" primaryKey="the primary key attribute ">
    <Table name="name of the table"/>
    <Level              name="nameofthelevel"          column="columnname"
uniqueMembers="true"/>
  </Hierarchy>
</Dimension>
```

9.3 FRONT END

Regardless of the strength of the OLAP engine and the integrity of the data, if the users cannot visualize the reports, query information and analyze data, the data warehouse brings zero value to them. Hence front-end development is an important part of a data warehousing initiative.

So what are the things to look out for in selecting a front-end deployment methodology? The most important thing is that the reports should need to be delivered over the web, so the only thing that the user needs is the standard browser. These days it is no longer desirable nor feasible to have the IT department doing program installations on end users desktops just so that they can view reports. So, whatever strategy one pursues, make sure the ability to deliver over the web is a must.

Query

In response to pressure for timely information, many banking facilities are developing large web enabled data warehouses. The data warehouse at the State Bank of Hyderabad has been assembled from several disparate legacy and departmental systems using Oracle RDBMS. To enable bank staffs, bank partners and customers to query the data in a timely and independent manner we have constructed an interface that is powerful, intuitive, and easy to use.

Method.

The interface uses the Web and consists of a data dictionary, a collection of Java Server Pages (JSP) programs implemented using the 'Java' programming language, and Servelets-enabled HTML pages. Together, these components facilitate the navigation and retrieval of data from the warehouse. The HTML pages generate SQL automatically in response to point-and-click actions by the user, enabling submission of ad-hoc queries without prior knowledge of SQL. The SQL queries are sent to the JSP programs that query the data warehouse and return results in dynamically created HTML pages. The entire process is controlled by the contents of the data dictionary, which is used to format SQL results, set up HTML links for data "drill-down.

The example below shows how the customer information can be obtained by searching a customer using a customer's account number.

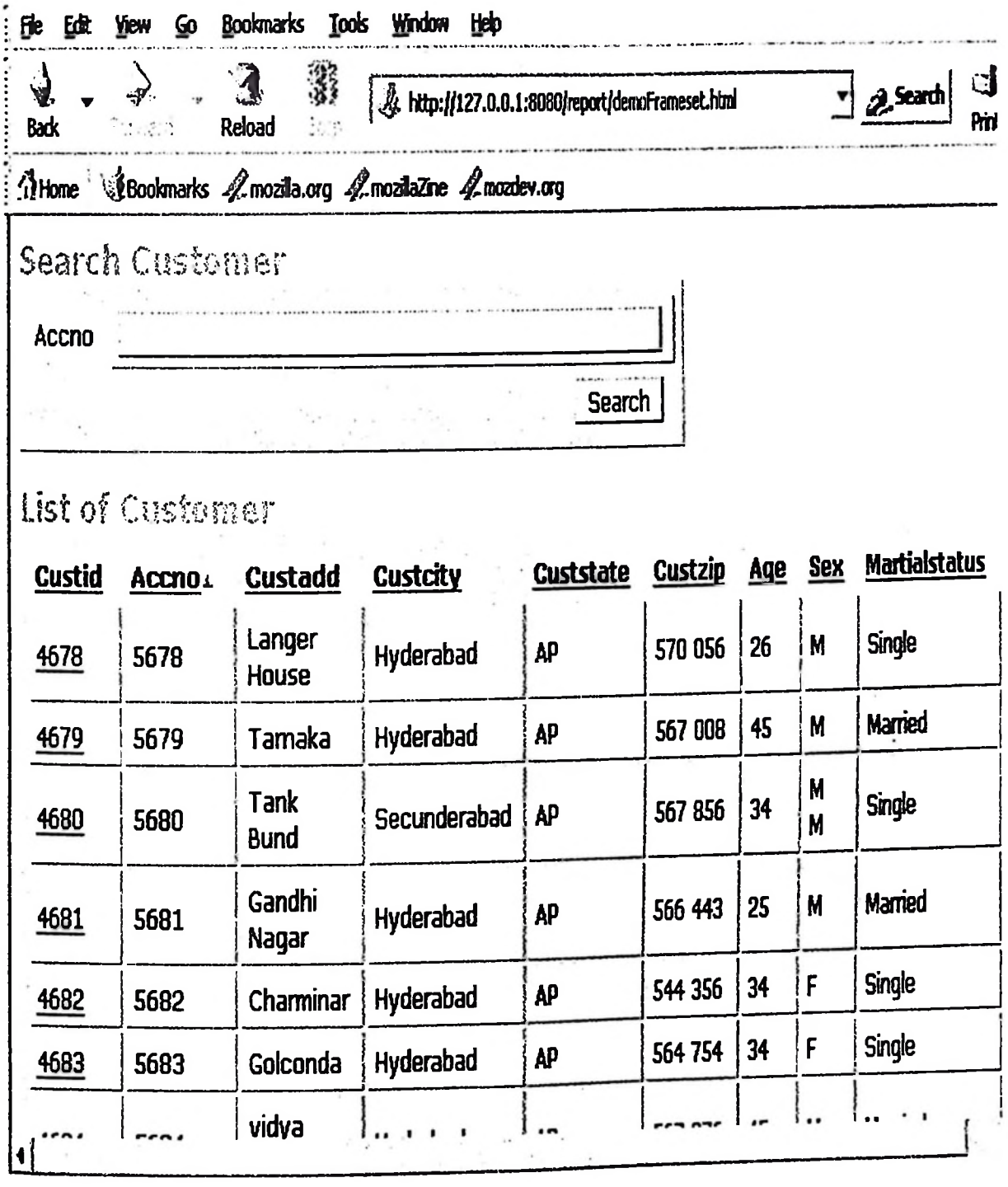


Fig 9.3.1 Customer Search Window

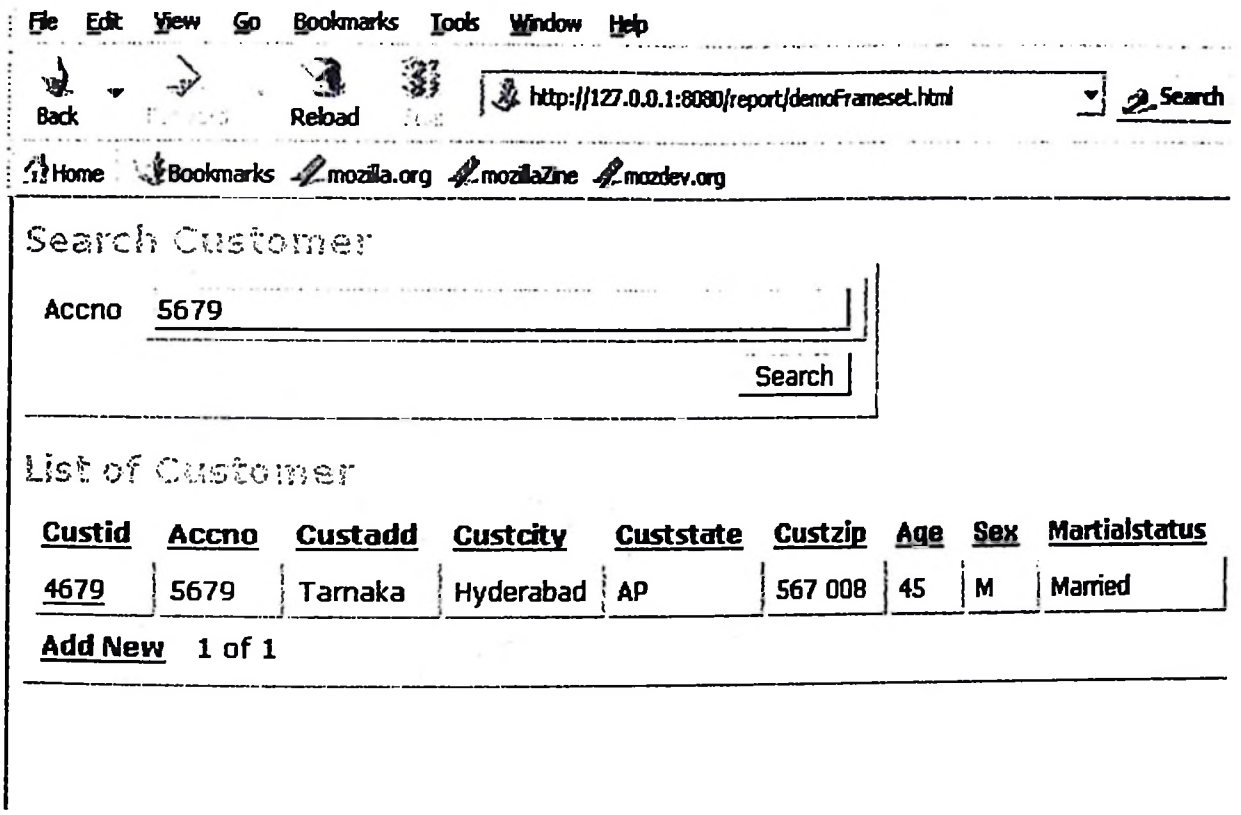
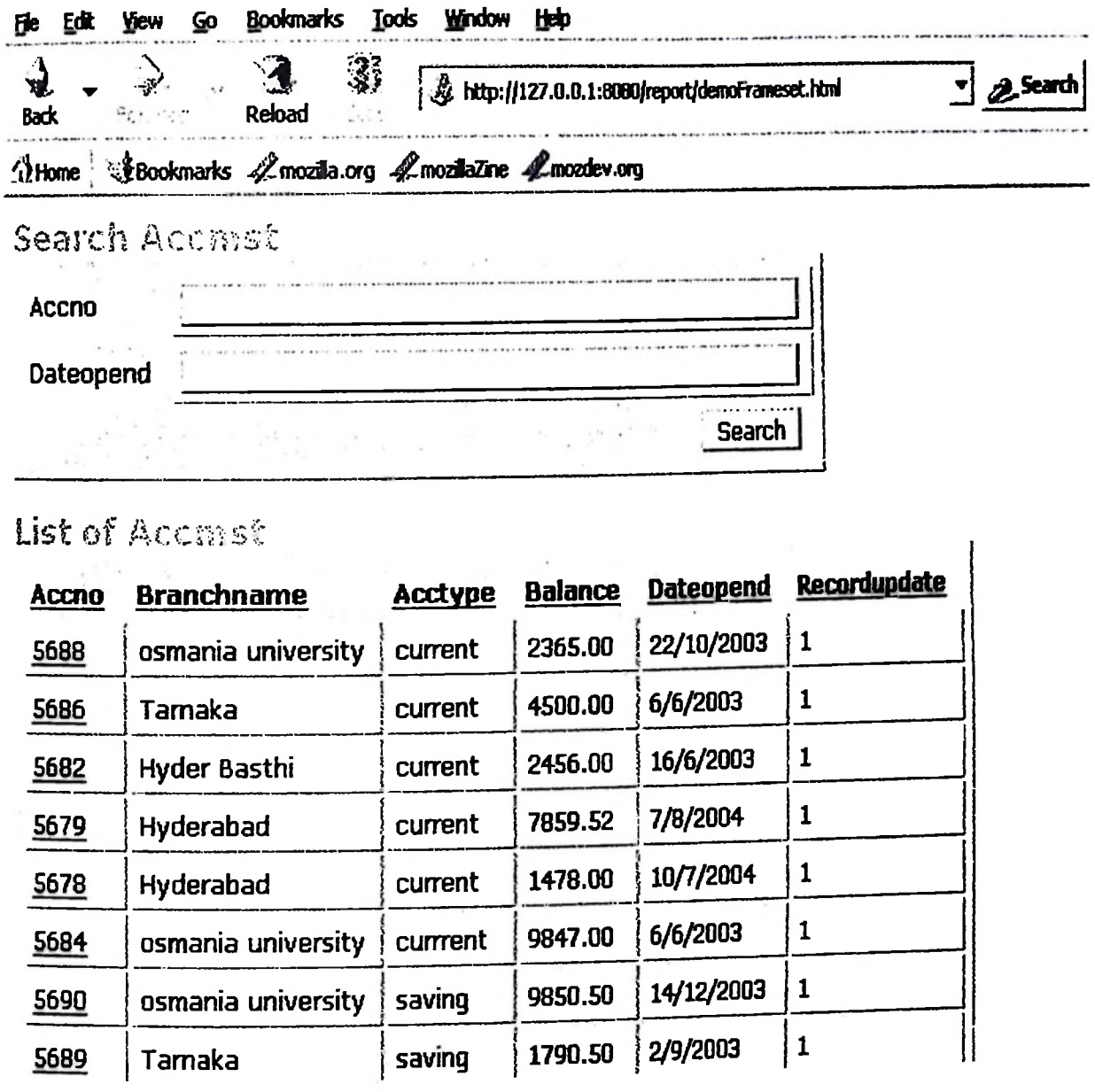


Fig 9.3.2 Customer Search Results

The example below shows how the account information can be obtained.



Accno	Branchname	Acctype	Balance	Dateopend	Recordupdate
5688	osmania university	current	2365.00	22/10/2003	1
5686	Tarnaka	current	4500.00	6/6/2003	1
5682	Hyder Basthi	current	2456.00	16/6/2003	1
5679	Hyderabad	current	7859.52	7/8/2004	1
5678	Hyderabad	current	1478.00	10/7/2004	1
5684	osmania university	current	9847.00	6/6/2003	1
5690	osmania university	saving	9850.50	14/12/2003	1
5689	Tarnaka	saving	1790.50	2/9/2003	1

Fig 9.3.3 Account Search Window

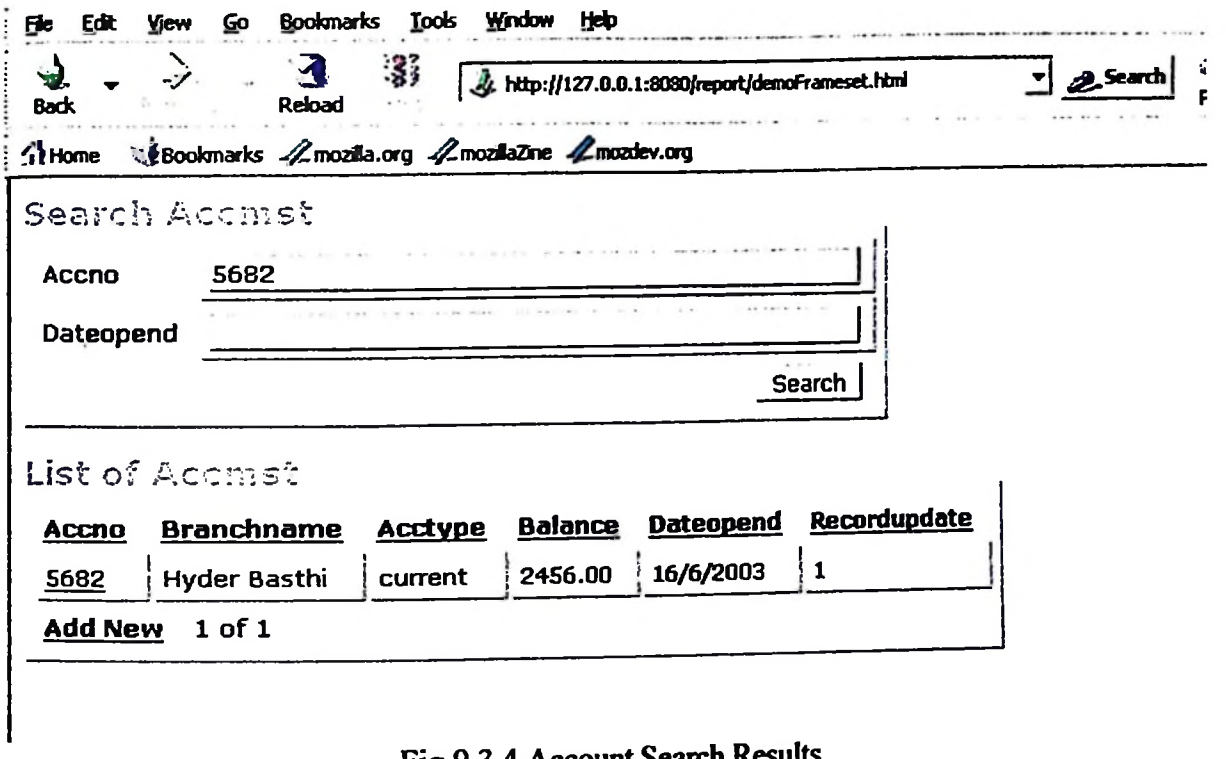


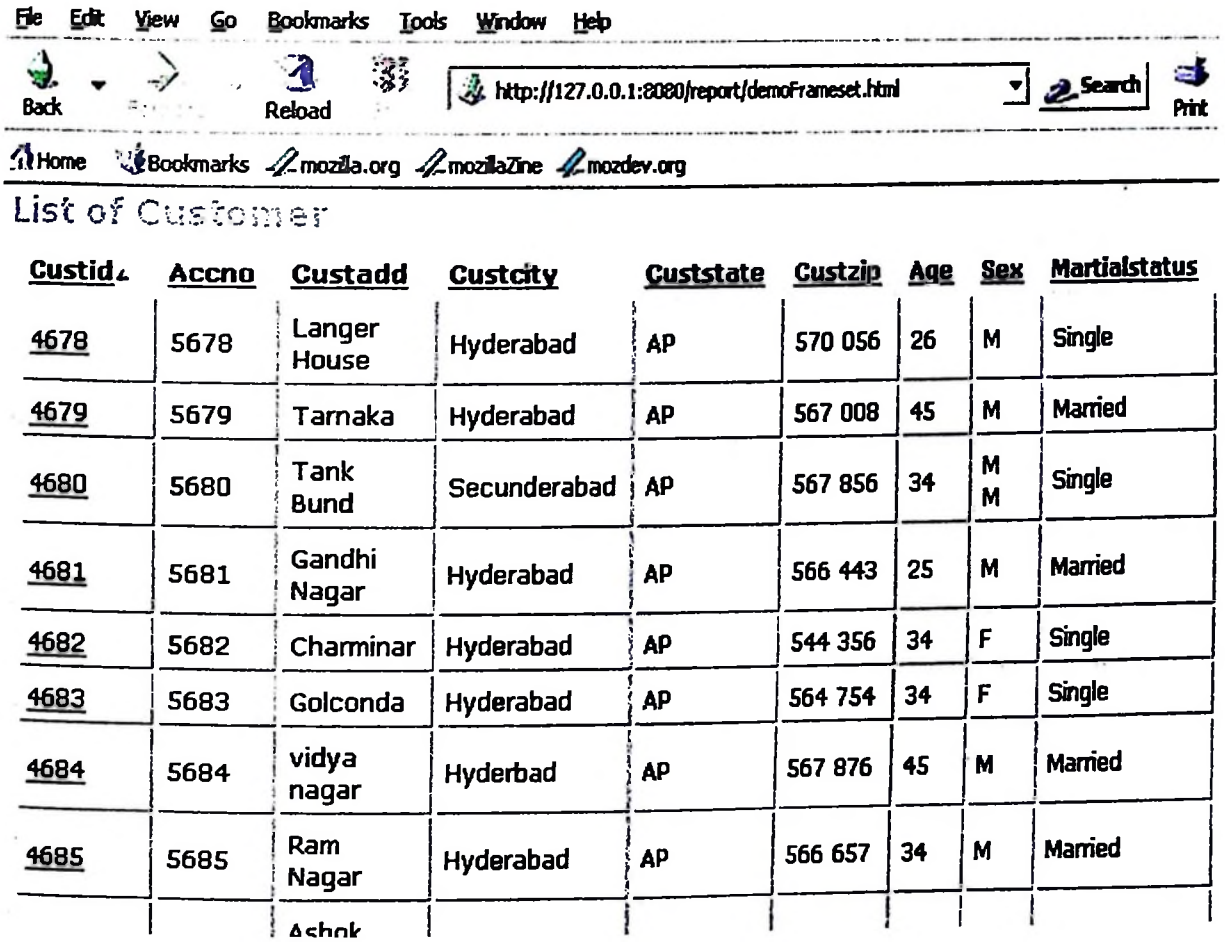
Fig 9.3.4 Account Search Results

Report

Web reporting tools, or Web-based reporting systems, allow companies and organizations to gain a better understanding of all of their business by putting critical information in the hands of all those who need it – employees, managers, partners, and customers. A true enterprise-reporting tool must be capable of accessing all of the information assets in the enterprise and it must support wide-scale deployment – potentially hundreds or even thousands of users – not just a handful of power users.

The same methods and technologies used in querying has been used to implement the reporting part of the data warehouse.

The example below shows a report of the list of customers along with their account details. The reporting tool allows a user to perform more operations such sorting using any attribute as a key by just clicking on the desired key.



Custid	Accno	Custadd	Custcity	Coststate	Custzip	Age	Sex	Martialstatus
4678	5678	Langer House	Hyderabad	AP	570 056	26	M	Single
4679	5679	Tarnaka	Hyderabad	AP	567 008	45	M	Married
4680	5680	Tank Bund	Secunderabad	AP	567 856	34	M M	Single
4681	5681	Gandhi Nagar	Hyderabad	AP	566 443	25	M	Married
4682	5682	Charminar	Hyderabad	AP	544 356	34	F	Single
4683	5683	Golconda	Hyderabad	AP	564 754	34	F	Single
4684	5684	vidya nagar	Hyderbad	AP	567 876	45	M	Married
4685	5685	Ram Nagar	Hyderabad	AP	566 657	34	M	Married
		Ashok						

Fig 9.3.5 List of Customers

Analysis

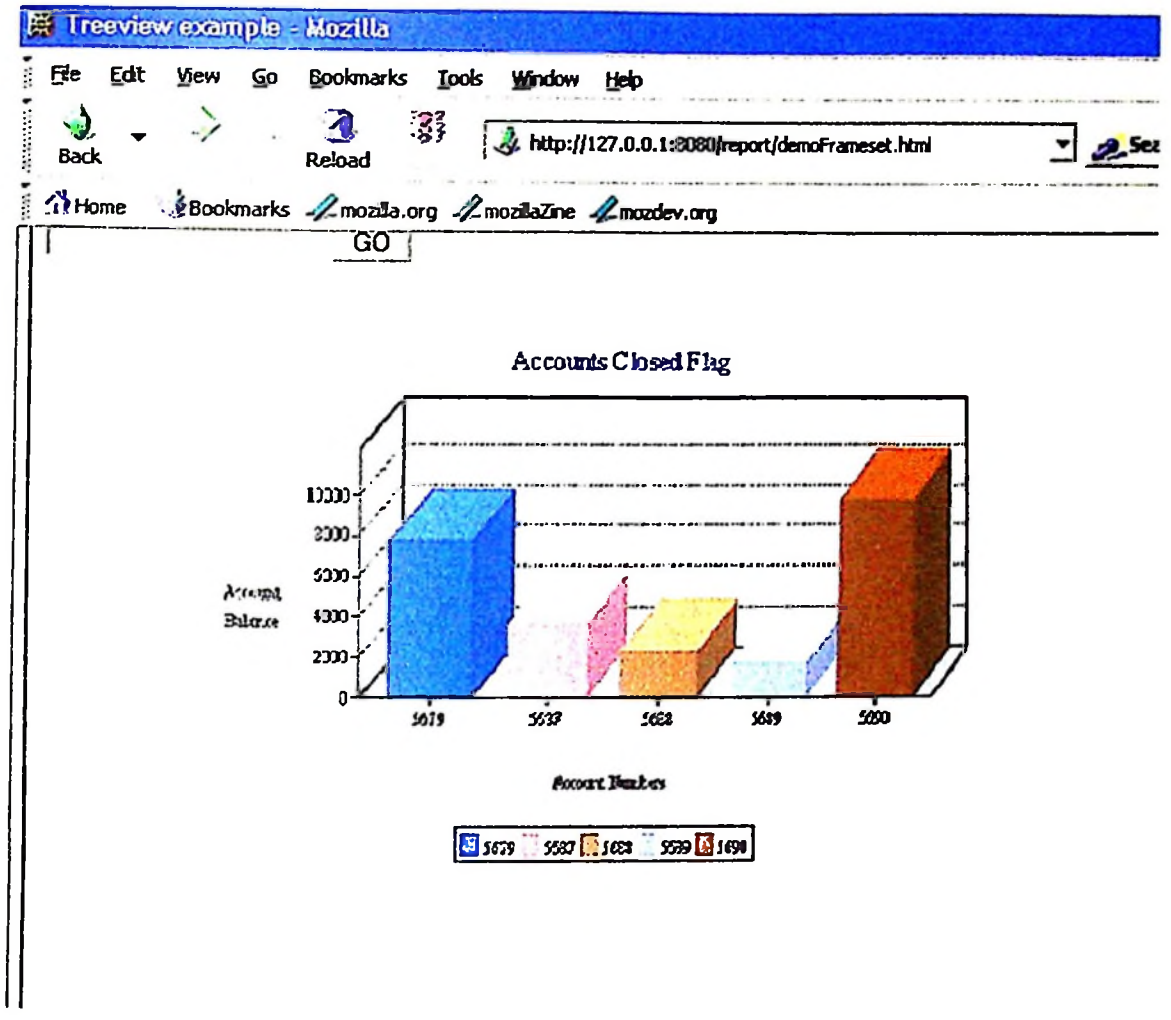
The analysis application has been implemented using JPivot.

JPivot is a JSP custom tag library that renders an OLAP table and let users perform typical OLAP navigations like slice and dice, drill down and roll up. It uses Mondrian as its OLAP Server.

JPivot is designed to work with several OLAP servers, especially XMLA and Mondrian. So JPivot does not use the Mondrian API directly. It defines its own OLAP model in terms of interfaces in the packages *olap.model* and *olap.navi*. To make JPivot work with Mondrian these interfaces have to be implemented using Mondrian classes.

JPivot uses WCF (Web Component Framework) packages that support building UI components via XML and XSLT.

The examples below show some of the screenshots for bank analysis implemented.



Accounts Closed

Acc#	Branch	Account Type	Balance
5679	Hyderabad	Current	7860
5687	Secunderabad	Saving	3646
5688	Osmania university	Current	2365
5689	Tarnaka	Saving	1790
5690	Osmania university	Saving	9850

Fig 9.3.1 Account Closed Flag

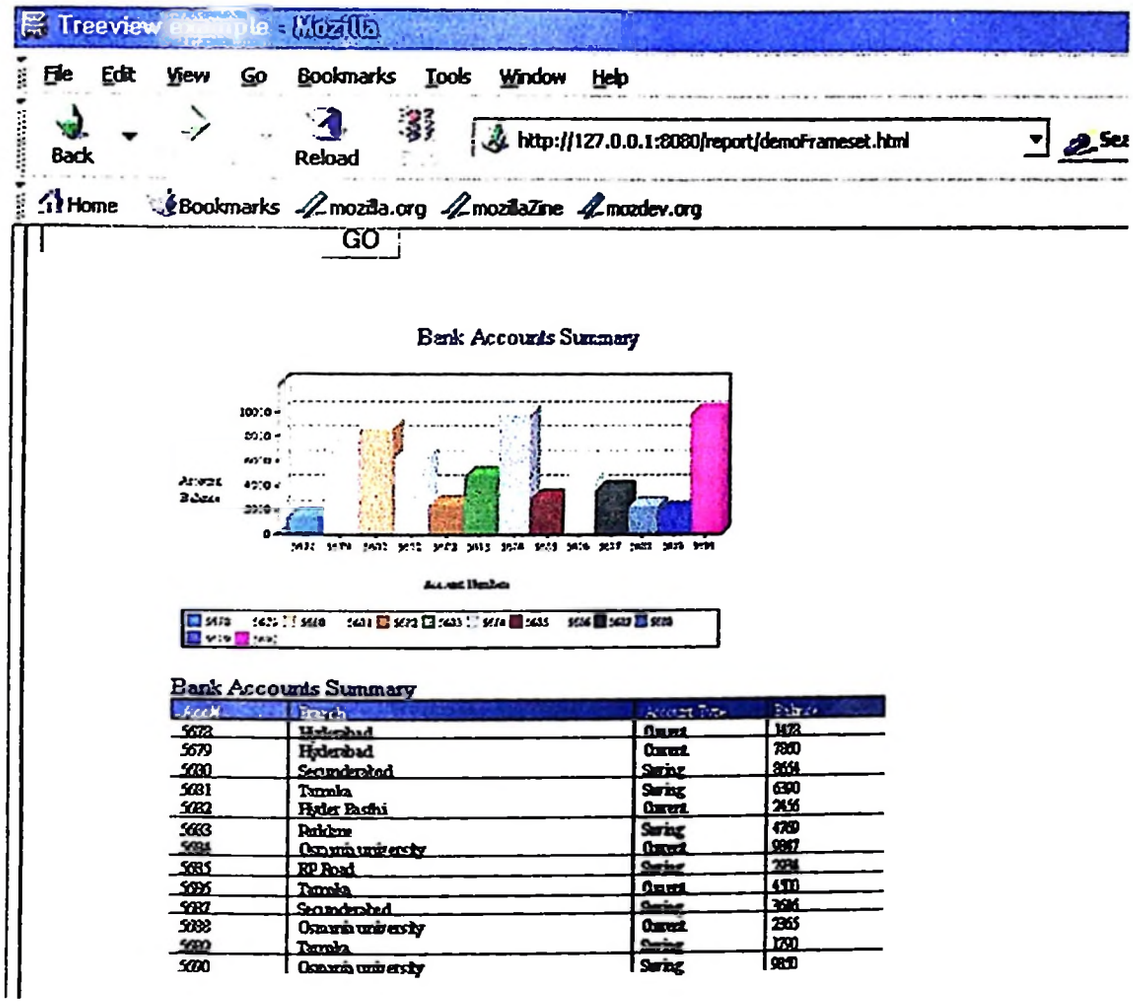


Fig 9.3.2 Bank Account Summary

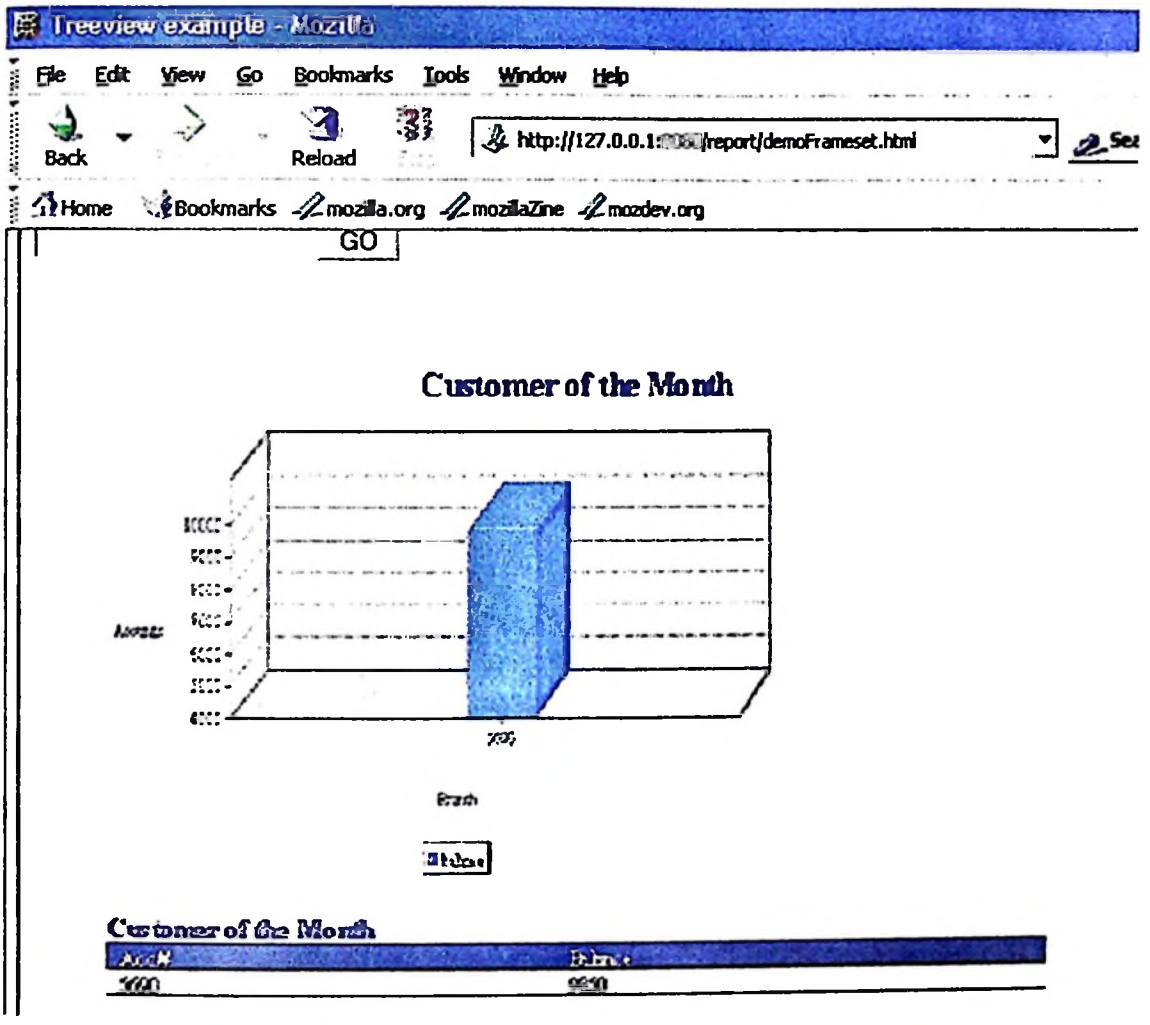


Fig 9.3.3 Customer of the Month

Finally, this thesis has shown the viability of adapting the Data warehouse for the Web (taking the **Warehouse to the Web**) and the integration of OLAP and Web technologies.

10.1 Future Work

For the immediate future work, the most pressing issue is bringing the **Web to the warehouse.** When you talk about Web-enabling the data warehouse, the first, and perhaps the only thought that comes to mind is the use of Web technology as an information delivery mechanism. Ironically, it rarely crosses your mind that Web content is a valuable and potent data source for your data warehouse. You may hesitate before extracting data from the Web for your Web-enables data warehouse.

Information content on the Web is so disparate and fragmented. We need to build a special search and extract system to sift through the mounds of information and pick up what is relevant for our data warehouse.

How can someone use Web content to enrich his/her data warehouse? Here are a few important uses: -

- Add more descriptive attributes to the business dimensions.
- Include nominal and ordinal data about a dimension so that more options are available for pivoting and cross-tabulations.
- Add linkage data to a dimension so that correlation analysis with other dimensions can be performed.
- Create a new dimension table
- Create a new fact table.

What we have discussed here is much beyond the usual use of the Web as an information delivery medium. Data selection and data extraction from the Web is a radically new paradigm.

GLOSSARY:

Aggregation: One way of speeding up query performance. Facts are summed up for selected dimensions from the original fact table. The resulting aggregate table will have fewer rows, thus making queries that can use them go faster.

Attribute: Attributes represent a single type of information in a dimension. For example, year is an attribute in the Time dimension.

Conformed Dimension: A dimension that has exactly the same meaning and content when being referred from different fact tables.

Data Mart: Data marts have the same definition as the data warehouse (see below), but data marts have a more limited audience and/or data content.

Data Warehouse: A warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process (as defined by Bill Inmon).

Data Warehousing: The process of designing, building, and maintaining a data warehouse system.

Dimension: The same category of information. For example, year, month, day, and week are all part of the Time Dimension.

Dimensional Model: A type of data modeling suited for data warehousing. In a dimensional model, there are two types of tables: dimensional tables and fact tables.

Dimensional table records information on each dimension, and fact table records all the "fact", or measures.

Dimensional Table: Dimension tables store records related to this particular dimension. No facts are stored in a dimensional table.

Drill Across: Data analysis across dimensions.

Drill Down: Data analysis to a child attribute.

Drill Through: Data analysis that goes from an OLAP cube into the relational database.

Drill Up: Data analysis to a parent attribute.

ETL: Stands for Extraction, Transformation, and Loading. The movement of data from one area to another.

Fact Table: A type of table in the dimensional model. A fact table typically includes two types of columns: fact columns and foreign keys to the dimensions.

Hierarchy: A hierarchy defines the navigating path for drilling up and drilling down. All attributes in a hierarchy belong to the same dimension.

Metadata: Data about data. For example, the number of tables in the database is a type of metadata.

Metric: A measured value. For example, total sales is a metric.

MOLAP: Multidimensional OLAP. MOLAP systems store data in the multidimensional cubes.

OLAP: On-Line Analytical Processing. OLAP should be designed to provide end users a quick way of slicing and dicing the data.

ROLAP: Relational OLAP. ROLAP systems store data in the relational database.

Snowflake Schema: A common form of dimensional model. In a snowflake schema, different hierarchies in a dimension can be extended into their own dimensional tables. Therefore, a dimension can have more than a single dimension table.

Star Schema: A common form of dimensional model. In a star schema, each dimension is represented by a single dimension table.

Thin client. A client machine that performs very little data or application processing – the processing is done in the server, and the client machine processes the output for the screen display.

Bibliography

[1].Paulraj Ponniah,

Data warehousing Fundamentals, Canada: John Wiley, 2001

[2].Erik Thomsen,

OLAP SOLUTIONS: Building Multidimensional Information Systems, USA: John Wiley, 2002

[3].Michael L.Gonzales,

IBM Data Warehousing: DB2, Intelligent Miner, DB2 OLAP with Data Warehousing Architecture, Analytics & Data Management, USA: Willey Publishing, 2003

[4].Ralph Kimball,

Data Warehouse ToolKit: Practical Techniques for Building dimensional Data Warehouses, USA: John Wiley,2002

[5].Arun K Pujari,

Data Mining Techniques, India: Universities Press Pvt Ltd,2002

[6].Ralph Kimball,

The Data Warehouse Lifecycle ToolKit: Expert Methods for Designing, Developing and Deploying Data Warehouse, USA: John Wiley, 2003

[7].Ralph Kimball and Margy Ross,

The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, USA: John Wiley and Sons, 2002

[8].Tom Soukup and Ian Davidson,

Visual Data Mining: Techniques and Tools for Data Visualization and Mining, USA: Wiley Publishing,2002.

[9].Marty Hall and Larry Brown,

Core Servlets and JavaServer Pages, India: Pearson Education Asia, 2004

[10].Campione,Walrath,Huml and Tutorial team,

The Java Tutorial Continued: The Rest of the JDK, India: Pearson Education Asia, 2001

[11].Silberschatz,Korth and Sudarshan,

Database System Concepts, USA: Mc Graw Hill,2002

[12].Kevin Loney ,Marlene Theriault and the Experts at TUSC,

Oracle 9i:DBA Handbook, USA: Mc Graw Hill,2002

[13].Kevin Loney ,George Koch and the Experts at TUSC,

Oracle 9i The Complete Reference: Master All the features and Functionality of

Oracle9i:DBA Handbook, USA: Mc Graw Hill,2002

[14].Dr.K.Nirmala Prasad J.Chandradass,

Banking and Financial System, India: Himalaya Publishing House, 2002

[15].Nita K.Brozowski,

Data Warehousing Fundamentals: Student Guide, USA: Oracle India Private

Limited, 2002

[16].John B.Dawson,

Data Warehouse Database Design: Student Guide, USA: Oracle India Private

Limited, 2001

[17].May Lonn Chan-Villareal,

Oracle iDS Implement Warehouse Builder: Student Guide Volume 1, USA:

Oracle India Private Limited, 2001

[18]. May Lonn Chan-Villareal,

Oracle iDS Implement Warehouse Builder: Student Guide Volume 2, USA: Oracle India Private Limited, 2001