Chameli Devi Group of Institutions, Indore
Department of Computer Science and Engineering
B.Tech.-7th Semester (**Grading System**) Odd Sem 2024

CS-703(B) Open Elective, Data Mining and Warehousing

COURSE OBJECTIVES

1. Student should understand the value of Historical data and data mining in solving real-world problems.
2. Student should become affluent with the basic Supervised and unsupervised learning algorithms commonly used in data mining.
3. Student develops the skill in using data mining for solving real-world problems.

Unit-I

Data Warehousing: Introduction, Delivery Process, Data warehouse Architecture, Data Preprocessing: Data cleaning, Data Integration and transformation, Data reduction. Data warehouse Design: Data warehouse schema, Partitioning strategy Data warehouse Implementation, Data Marts, Meta Data, Example of a Multidimensional Data model. Introduction to Pattern Warehousing.

Data Warehousing

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. DW is combining data from multiple and usually varied sources in to one comprehensive and easily manipulated database. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources. DW is commonly used by companies to analyze trends over time.

As compare to the relational database, a data warehouse environment includes an extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users.

"A warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process"

Understanding a Data Warehouse

- A data warehouse is a database, which is kept separate from the organization's operational database.
- There is no frequent updating done in a data warehouse.
- It possesses consolidated historical data, which helps the organization to analyze its business.
- A data warehouse helps executives to organize, understand, and use their data to take strategic decisions.
- Data warehouse systems help in the integration of diversity of application systems.
- A data warehouse system helps in consolidated historical data analysis.

Data Warehouse Features

The key features of a data warehouse are discussed below −

Subject Oriented − A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations, rather it focuses on modelling and analysis of data for decision making.

Integrated − A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.

Time Variant − Data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view.

Non-volatile − Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database is not reflected in the data warehouse.

Note − A data warehouse does not require transaction processing, recovery, and concurrency controls, because it is physically stored and separate from the operational database.

Data Warehouse Applications

As discussed before, a data warehouse helps business executives to organize, analyze, and use their data for decision making. A data warehouse serves as a sole part of a plan-execute-assess "closed-loop" feedback system for the enterprise management. Data warehouses are widely used in the following fields −

- Financial services
- Banking services
- Consumer goods
- Retail sectors
- Controlled manufacturing

Needs for developing data Warehouse:

- Provides an integrated and total view of the enterprise.
- Make the organizations current and historical information easily available for decision making.
- Make decision support transactions possible without hampering operational system.
- Provide consistent organizations information
- Provide a flexible and interactive sources of strategic information.
- End user creation of reports: The creation of reports directly by end users is much easier to accomplish in a BI environment.
- Dynamic presentation through dashboards: Managers want access to an interactive display of up-to-date critical management data.
- Drill-down capability
- Metadata creation: This will make report creation much simpler for the end-user

Delivery Process

The delivery method is a variant of the joint application development approach adopted for the delivery of a data warehouse. We have staged the data warehouse delivery process to minimize risks. The approach that we will discuss here does not reduce the overall delivery time-scales but ensures the business benefits are delivered incrementally through the development process.

Note − The delivery process is broken into phases to reduce the project and delivery risk.

The following diagram explains the stages in the delivery process −

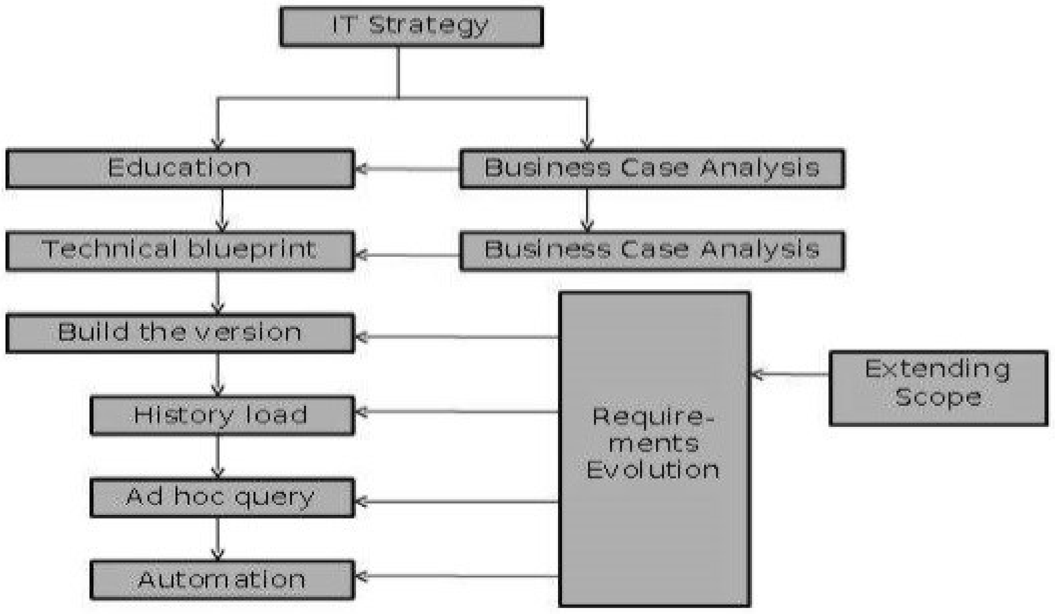


Figure 1: Stages in the delivery process

Data Warehouse Architectures

Data warehouses and their architectures vary depending upon the specifics of an organization's situation. Three common architectures are:

- Data Warehouse Architecture: Basic
- Data Warehouse Architecture with a Staging Area
- Data Warehouse Architecture: with a Staging Area and Data Marts

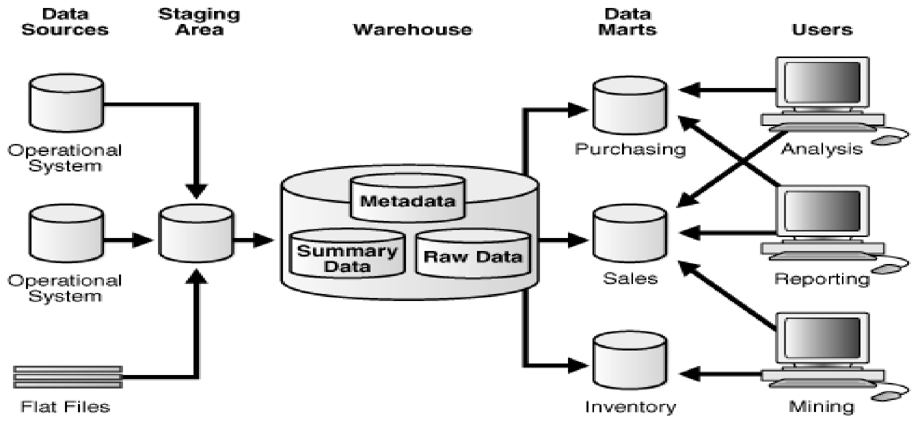


Figure 2: Data Warehouse Architecture

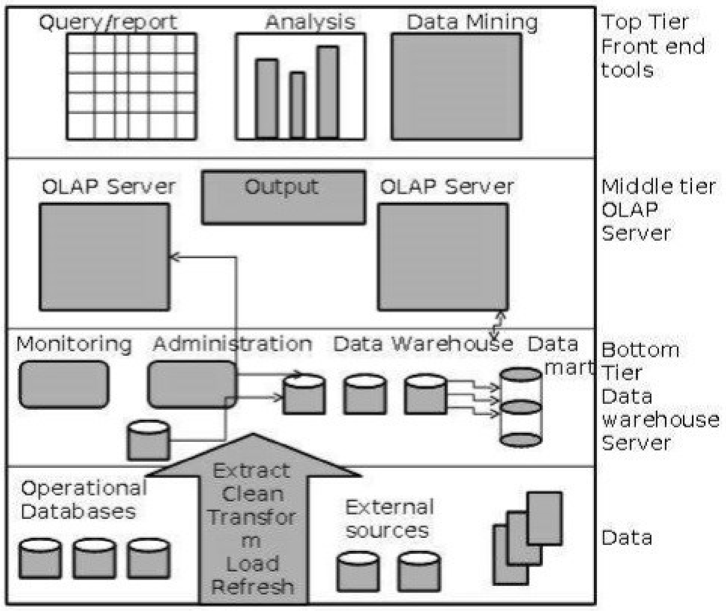


Figure 3: Three-tier architecture of data warehouse

Data warehouse systems and its Components:

Data warehousing is typically used by larger companies analyzing larger sets of data for enterprise purposes. The data warehouse architecture is based on a relational database system server that functions as the central warehouse for informational data. Operational data and processing is purely based on data warehouse processing. This central information system is used some key components designed to make the entire environment for operational systems. It's mainly created to support different analysis, queries that need extensive searching on a larger scale.

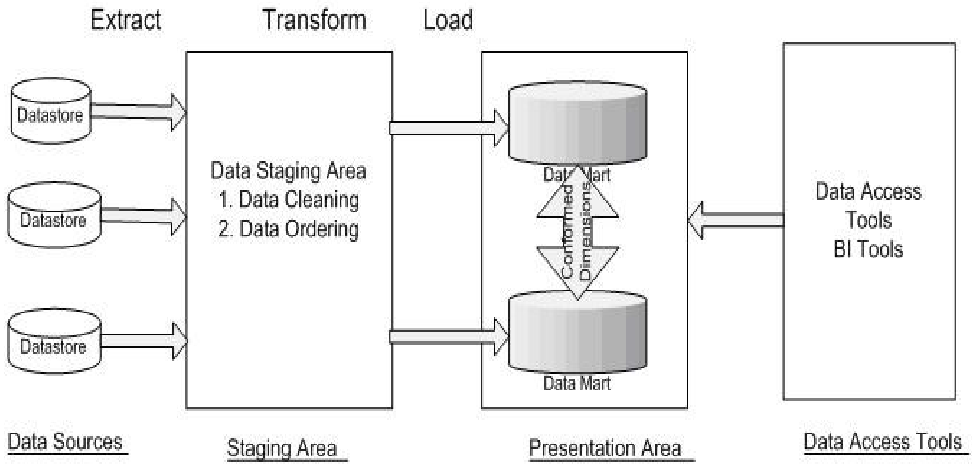


Figure 4: Data Warehouse Components

Operational Source System

Operational systems are tuned for known transactions and workloads, while workload is not known a priori in a data warehouse. Traditionally data base system used for transaction processing systems which stores transaction data of the organizations business. Its generally used one record at any time not stores history of the information's.

Data Staging Area

As soon as the data arrives into the Data staging area it is set of ETL process that extract data from source system. It is converted into an integrated structure and format.

Data is extracted from source system and stored, cleaned, transform functions that may be applied to load into data warehouse.

Removing unwanted data from operational databases.

Converting to common data names and definitions.

Establishing defaults for missing data accommodating source data definition changes.

Data Presentation Area

Data presentation area are the target physical machines on which the data warehouse data is organized and stored for direct querying by end users, report writers and other applications. It's the place where cleaned, transformed data is stored in a dimensionally structured warehouse and made available for analysis purpose.

Data Access Tools

End user data access tools are any clients of the data warehouse. An end user access tool can be a complex as a sophisticated data mining or modeling applications.

Why preprocessing?

1. Real world data are generally

- ➢ Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
- ➢ Noisy: containing errors or outliers
- ➢ Inconsistent: containing discrepancies in codes or names

2. Tasks **in data** preprocessing

- ➢ Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
- ➢ Data integration: using multiple databases, data cubes, or files.
- ➢ Data transformation: normalization and aggregation.
- ➢ Data reduction: reducing the volume but producing the same or similar analytical results.
- ➢ Data discretization: part of data reduction, replacing numerical attributes with nominal ones.

Steps Involved in Data Preprocessing:

1. Data Cleaning/Cleansing

Real-world data tend to be incomplete, noisy, and inconsistent. Data Cleaning/Cleansing routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. Data can be noisy, having incorrect attribute values. Owing to the following, the data collection instruments used may be fault. Maybe human or computer errors occurred at data entry. Errors in data transmission can also occur.

"Dirty" data can cause confusion for the mining procedure. Although most mining routines have some procedures, they deal incomplete or noisy data, which are not always robust. Therefore, a useful Data Preprocessing step is to run the data through some Data Cleaning/Cleansing routines.

2. Data Integration

Data Integration is involved in data analysis task which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. The issue to be considered in Data Integration is schema integration. It is tricky. How can real-world entities from multiple data sources be 'matched up'? This is referred as entity identification problem. For example, how can a data analyst be sure that customer_id in one database and cust_number in another refer to the same entity? The answer is metadata. Databases and data warehouses typically have metadata. Simply, metadata is data about data.
Metadata is used to help avoiding errors in schema integration. Another important issue is redundancy. An attribute may be redundant, if it is derived from another table. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

3. Data Transformation

Data are transformed into appropriate forms of mining. Data Transformation involves the following:

1. In Normalization, where the attribute data are scaled to fall within a small specified range, such as -1.0 to 1.0, or 0 to 1.0.

2. Smoothing works to remove the noise from the data. Such techniques include binning, clustering, and regression.

3. In Aggregation, summary or aggregation operations are applied to the data. For example, daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

4. In Generalization of the Data, low level or primitive/raw data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes are generalized to higher level concepts street into city or country. Similarly, the values for numeric attributes may be mapped to higher level concepts like, age into young, middle-aged, or senior.

4. Data Reduction

Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data Reduction techniques are helpful in analyzing the reduced representation of the data set without compromising the integrity of the original data and yet producing the qualitative knowledge. Strategies for data reduction include the following:

I. In Data Cube Aggregation, aggregation operations are applied to the data in the construction of a data cube.

II. In Dimension Reduction, irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

III. In Data Compression, encoding mechanisms are used to reduce data set size. The methods used for Data Compression are Wavelet Transform and Principle Component Analysis.

IV. In Numerosity Reduction, data is replaced or estimated by alternative and smaller data representations such as parametric models (which store only the model parameters instead of the actual data, e.g. Regression and Log-Linear Models) or non-parametric methods (e.g. Clustering, Sampling, and the use of histograms).

V. In Discretisation and Concept Hierarchy Generation, raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are powerful tools for data mining.

Design of Data Warehouse

Design Methods

Bottom-up design

This architecture makes the data warehouse more of a virtual reality than a physical reality. In the bottom-up approach, starts with extraction of data from operational database into the staging area where it is processed and consolidated for specific business processes. The bottom-up approaches reverse the positions of the data warehouse and the data marts. These data marts can then be integrated to create a comprehensive data warehouse.

Top-down design

The data flow in the top down OLAP environment begins with data extraction from the operational data sources. The top-down approach is designed using a normalized enterprise data model. The results are obtained quickly if it is implemented with iterations. It is time consuming process with an iterative method and the failure risk is very high.

Hybrid design

The hybrid approach aims to harness the speed and user orientation of the bottom up approach to the integration of the top-down approach. To consolidate these various data models, and facilitate the extract transform load process, data warehouses often make use of an operational data store, the information from which is parsed into the actual DW. The hybrid approach begins with an ER diagram of the data mart and a gradual extension of the data marts to extend the enterprise model in a consistent linear fashion. It will provide rapid development within an enterprise architecture framework.

Dimension and Measures

Data warehouse consists of dimensions and measures. It is a logical design technique used for data warehouses. Dimensional model allow data analysis from many of the commercial OLAP products available today in the market. For example, time dimension could show you the breakdown of sales by year, quarter, month, day and hour.

Measures are numeric representations of a set of facts that have occurred. The most common measures of data dispersion are range, the five number summery (based on quartiles), the inter-quartile range, and the standard deviation. Examples of measures include amount of sales, number of credit hours, store profit percentage, sum of operating expenses, number of past-due accounts and so forth.

Types

Conformed dimension

Junk dimension

Degenerate dimension

Role-playing dimension

Data Marts

A data mart is a specialized system that brings together the data needed for a department or related applications. A data mart is a simple form of a data warehouse that is focused on a single subject (or functional area), such as educational, sales, operations, collections, finance or marketing data.

The sources may contain internal operational systems, central data warehouse, or external data. It is a small warehouse which is designed for the department level.

Dependent, Independent or stand-alone and Hybrid Data Marts

Three basic types of data marts are dependent, independent or stand-alone, and hybrid. The categorization is based primarily on the data source that feeds the data mart.

Dependent data marts: Data comes from warehouse. It is actually created a separate physical data-store.

Independent data marts: A standalone systems built by drawing data directly from operational or external sources of data or both. Independent data mart are independent and focuses exclusively on one subject area. It has a separate physical data store.

Hybrid data marts: Can draw data from operational systems or data warehouses.

Dependent Data Marts

A dependent data mart allows you to unite your organization's data in one data warehouse. This gives you the usual advantages of centralization. Figure 5 shows a dependent data mart.

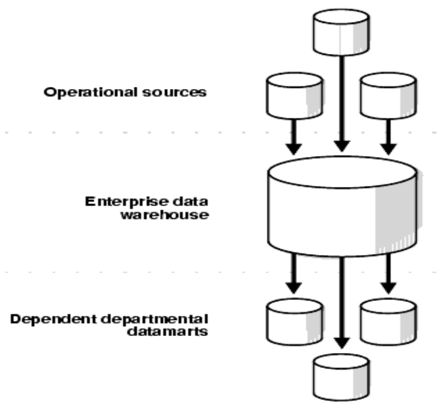


Figure 5: Dependent Data Mart

Independent or Stand-alone Data Marts

An independent data mart is created without the use of a central data warehouse. This could be desirable for smaller groups within an organization. It is not, however, the focus of this Guide. Figure 6 shows an independent data mart.

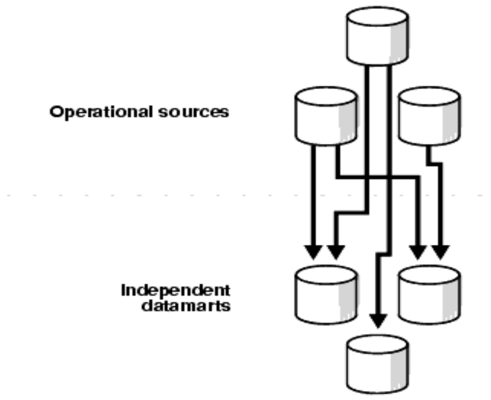


Figure 6: Independent Data Marts

Hybrid Data Marts

A hybrid data mart allows you to combine input from sources other than a data warehouse. This could be useful for many situations, especially when you need ad hoc integration, such as after a new group or product is added to the organization. Provides rapid development within an enterprise architecture framework. Figure 7 shows hybrid data mart.

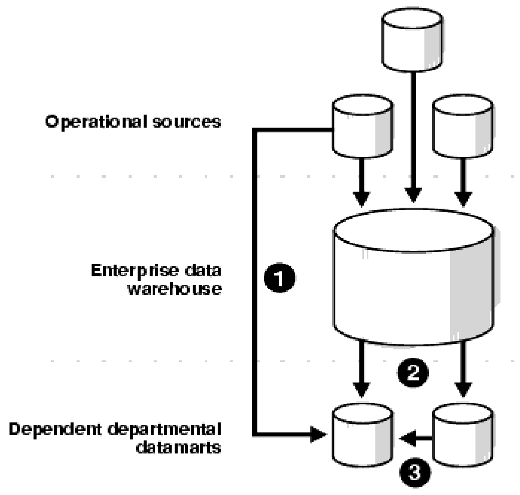


Figure 7: Hybrid Data Mart

What is Metadata?

Metadata is simply defined as data about data. The data that is used to represent other data is known as metadata. For example, the index of a book serves as a metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data. In terms of data warehouse, we can define metadata as follows.

- Metadata is the road-map to a data warehouse.
- Metadata in a data warehouse defines the warehouse objects.
- Metadata acts as a directory. This directory helps the decision support system to locate the contents of a data warehouse.

Metadata includes the following:

1. The location and descriptions of warehouse systems and components.
2. Names, definitions, structures, and content of data-warehouse and end-users views.
3. Identification of authoritative data sources.
4. Integration and transformation rules used to populate data.
5. Integration and transformation rules used to deliver information to end-user analytical tools.
6. Subscription information for information delivery to analysis subscribers.
7. Metrics used to analyze warehouses usage and performance.
8. Security authorizations, access control list, etc.

Categories of Metadata

Metadata can be broadly categorized into three categories −

Business Metadata − It has the data ownership information, business definition, and changing policies.

Technical Metadata − It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural information such as primary and foreign key attributes and indices.

Operational Metadata − It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.

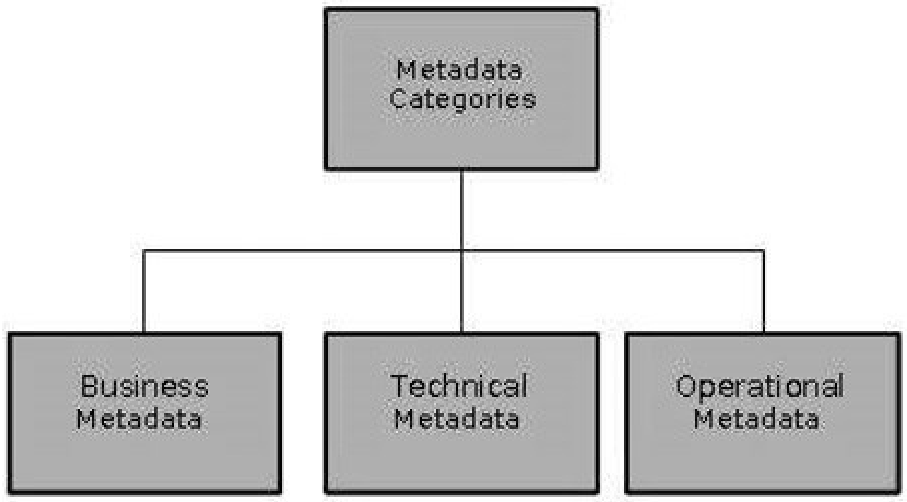


Figure 8: Categories of Metadata

Conceptual Modeling of Data Warehouses

It may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a set-grouping model. A conceptual data model identifies the highest-level relationships between the different entities. Features of conceptual data model include:

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

Conceptual Data Model

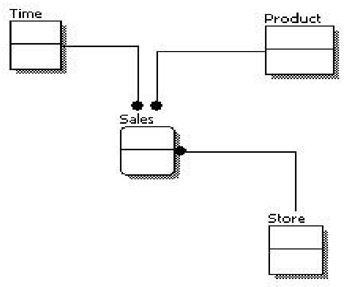


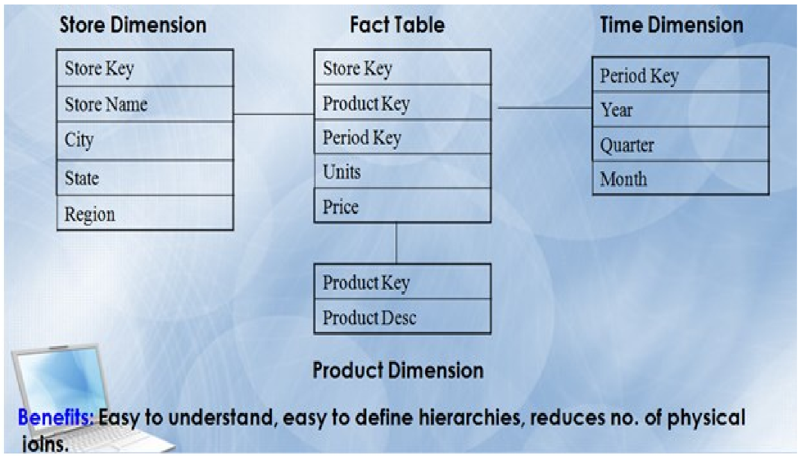
Figure 9: Conceptual data model

Data Warehousing Schemas
From the figure above, we can see that the only information shown via the conceptual data model is the entities that describe the data and the relationships between those entities. There may be more than one concept hierarchy for a given attribute or dimension, based on different users view points. No other information is shown through the conceptual data model.

- **Star Schema**
- **Snowflake Schema**
- **Fact Constellation**

Star Schema

- Consists of set of relations known as Dimension Table (DT) and Fact Table (FT)
- A single large central fact table and one table for each dimension.
- A fact table primary key is composition of set of foreign keys referencing dimension tables.
- Every dimension table is related to one or more fact tables.
- Every fact points to one tuple in each of the dimensions and has additional attributes
- Does not capture hierarchies directly.

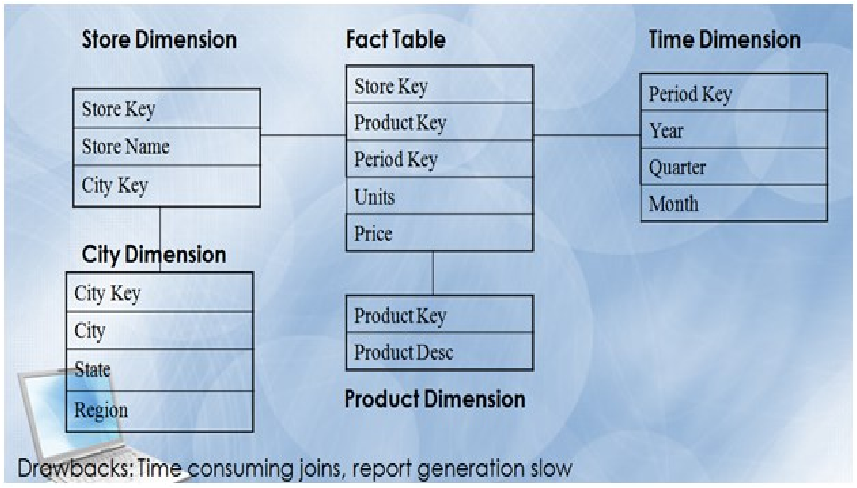


Benefits: Easy to understand, easy to define hierarchies, reduces no. of physical joins.

Snowflake Schema

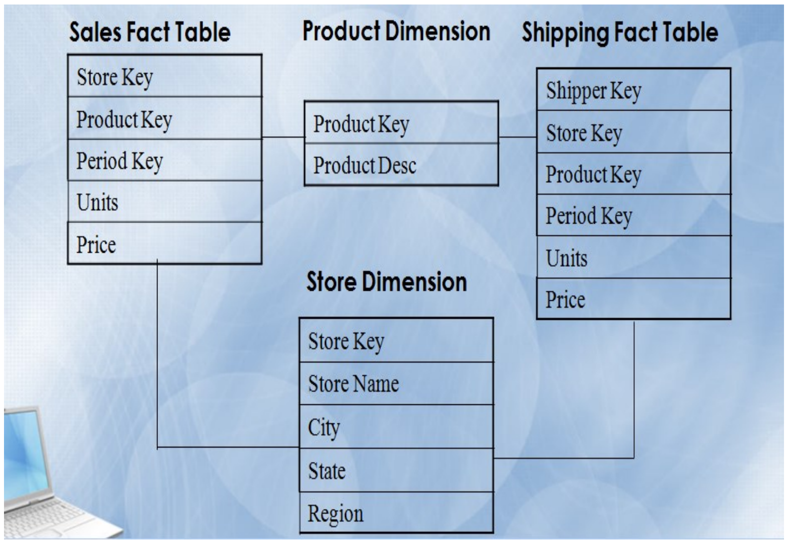
- Variant of star schema model.
- Used to remove the low cardinality.
- A single, large and central fact table and one or more tables for each dimension.

- Dimension tables are normalized split dimension table data into additional tables. But this may affect its performance as joins needs to be performed.
- Query performance would be degraded because of additional joins. (delay in processing)



Fact Constellation:

- As its name implies, it is shaped like a constellation of stars (i.e. star schemas).
- Allow to share multiple fact tables with dimension tables.
- This schema is viewed as collection of stars hence called galaxy schema or fact constellation.
- Solution is very flexible, however it may be hard to manage and support.
- Sophisticated application requires such schema.



Overview of Partitioning in Data Warehouses

Data warehouses often contain very large tables and require techniques both for managing these large tables and for providing good query performance across them. An important tool for achieving this, as well as enhancing data access and improving overall application performance is partitioning.

Partitioning offers support for very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions. This support is especially important for applications that access tables and indexes with millions of rows and many gigabytes of data. Partitioned tables and indexes facilitate administrative operations by enabling these operations to work on subsets of data. For example, you can add a new partition, organize an existing partition, or drop a partition with minimal to zero interruption to a read-only application.

When adding or creating a partition, you have the option of deferring the segment creation until the data is first inserted, which is particularly valuable when installing applications that have a large footprint.

Why is it Necessary to Partition?

Partitioning is important for the following reasons −

- For easy management,
- To assist backup/recovery,
- To enhance performance.

For Easy Management

The fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of fact table is very hard to manage as a single entity. Therefore it needs partitioning.

To Assist Backup/Recovery

If we do not partition the fact table, then we have to load the complete fact table with all the data. Partitioning allows us to load only as much data as is required on a regular basis. It reduces the time to load and also enhances the performance of the system.

Note − To cut down on the backup size, all partitions other than the current partition can be marked as read-only. We can then put these partitions into a state where they cannot be modified. Then they can be backed up. It means only the current partition is to be backed up.

To Enhance Performance

By partitioning the fact table into sets of data, the query procedures can be enhanced. Query performance is enhanced because now the query scans only those partitions that are relevant. It does not have to scan the whole data.

Horizontal Partitioning

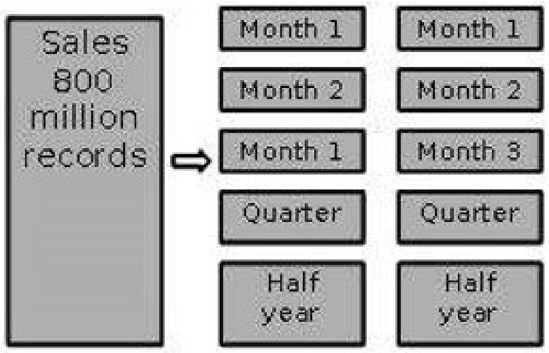
There are various ways in which a fact table can be partitioned. In horizontal partitioning, we have to keep in mind the requirements for manageability of the data warehouse.

Partitioning by Time into Equal Segments

In this partitioning strategy, the fact table is partitioned on the basis of time period. Here each time period represents a significant retention period within the business. For example, if the user queries for month to date data then it is appropriate to partition the data into monthly segments. We can reuse the partitioned tables by removing the data in them.

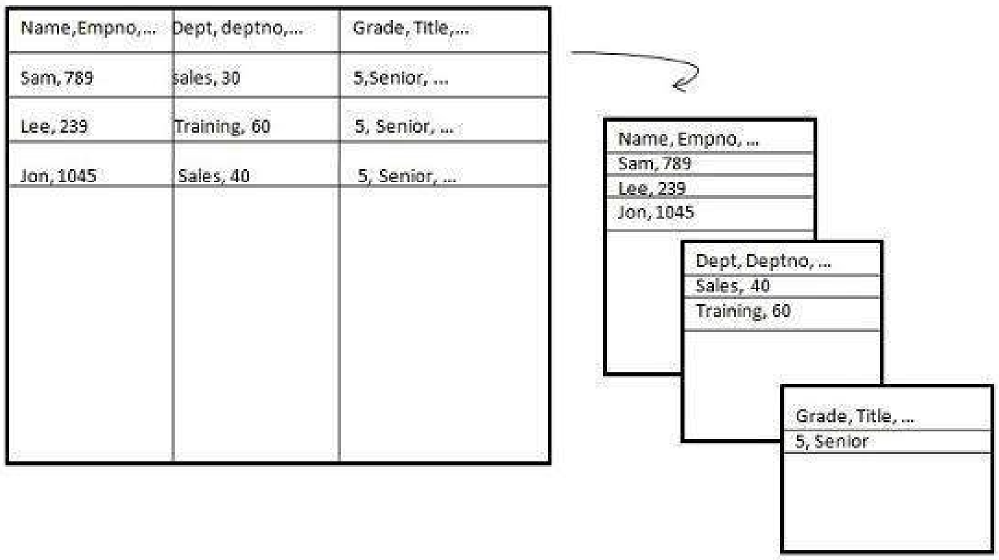
Partition by Time into Different-sized Segments

This kind of partition is done where the aged data is accessed infrequently. It is implemented as a set of small partitions for relatively current data, larger partition for inactive data.



Vertical Partition

Vertical partitioning, splits the data vertically. The following images depicts how vertical partitioning is done.



Vertical partitioning can be performed in the following two ways −

- Normalization
- Row Splitting

Multidimensional Data Model

Data warehouses are generally based on 'multi-dimensional" data model. The multidimensional data model provides a framework that is intuitive and efficient, that allow data to be viewed and analyzed at the desired level of details with a good performance. The multidimensional model start with the examination of factors affecting decision-making processes is generally organization specific facts, for example sales, shipments, hospital admissions, surgeries, and so on. One instances of a fact correspond with an event that occurred. For example, every single sale or shipment carried out is an event. Each fact is described by the values of a set of relevant measures that provide a quantitative description of events. For example, receipts of sales, amount of shipment, product cost are measures.

The multidimensional data model is an integral part of On-Line Analytical Processing, or OLAP. Because OLAP is on-line, it must provide answers quickly; analysts pose iterative queries during interactive sessions, not in batch jobs that run overnight. And because OLAP is also analytic, the queries are complex. Dimension tables support changing the attributes of the dimension without changing the underlying fact table. The multidimensional data model is designed to solve complex queries in real time. The multidimensional data model is important because it enforces simplicity.

The multidimensional data model is composed of logical cubes, measures, dimensions, hierarchies, levels, and attributes. The simplicity of the model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies. Figure shows the relationships among the logical objects.

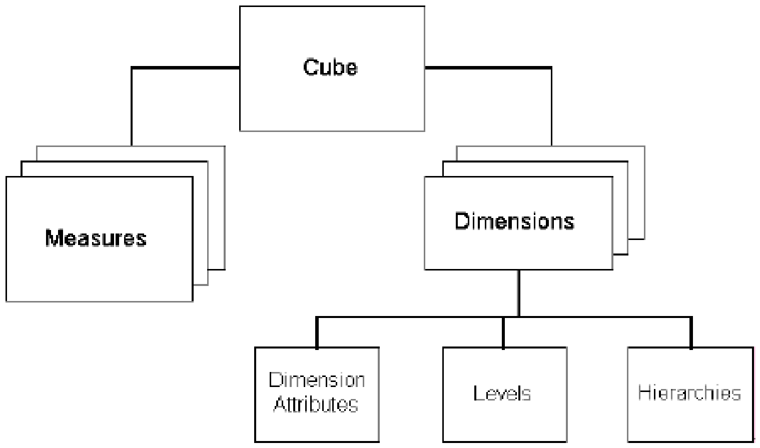


Figure 10: Logical Multidimensional Model

Aggregates

In data warehouse huge amount of data is stored that makes analyses of data very difficult. This is the basic reason why selection and aggregation is required to examine specific part of data. Aggregations are the way by which information can be divided so queries can be run on the aggregated part and not the whole set of data. These are pre-calculated summaries derived from the most granular fact table. It is a process for information is gathered and expressed in a summary form, for purposes such as statistical analysis. A common aggregation purpose is to get more information about particular groups based on specific variables such as age, profession, or income. The information about such groups can then be used for web site personalization. Tables are always changing along with the needs of the users so it is important to define the aggregations according to what summary tables might be of use.

Pattern Warehouse

The patterns a data mining system discovers are stored in a Pattern Warehouse. Just as a data warehouse stores data, the Pattern Warehouse stores patterns - it is an information repository that stores relationships between data items, but not the data. While data items are stored in data warehouse, we use the Pattern Warehouse to store the patterns and relationships among them.

The Pattern Warehouse is represented as a set of "pattern-tables" within a traditional relational database. This solves several potential issues regarding user access rights, security control, multiuser access, etc. But obviously, we need a language to access and query the contents of Pattern Warehouses. SQL may be considered an obvious first candidate for this, but when SQL was designed over 30 years ago, data mining was not a major issue. SQL was designed to access data stored in databases. We need pattern-oriented languages to access Pattern Warehouses storing various types of exact and inexact patterns. Often, it is very hard to access these patterns with SQL.

Hence a Pattern Warehouse cannot be conveniently queried in a direct way using a relational query language. Not only are some patterns not easily stored in a simple tabular format, but by just looking up influence factors in pattern-tables we may get incorrect results. We need a "pattern-kernel" that consistently manages and merges patterns. The pattern-kernel forms the heart of Pattern Query Language (PQL). The PQL, which does for decision support spaces what SQL, does for the data space. While SQL relies on the relational algebra, PQL uses the "pattern algebra". PQL was designed to access Pattern Warehouses just as SQL was designed to access databases. PQL was designed to be very similar to SQL. It allows knowledge-based queries just as SQL allows data based queries.

----------------------------***----------------------------