Object Oriented Database (**OODB**) provides all the facilities associated with object oriented paradigm. It enables us to create classes, organize objects, structure an inheritance hierarchy and call methods of other classes. Besides these, it also provides the facilities associated with standard database systems. However, object oriented database systems have not yet replaced the RDBMS in commercial business applications. Following are the two different approaches for designing an object-oriented database:

• Designed to store, retrieve and manage objects created by programs written in some object oriented languages (OOL) such as C++ or java.

Although a relational database can be used to store and manage objects, it does not understand objects as such. Therefore, a middle layer called object manager or object-oriented layer software is required to translate objects into tuples of a relation .

• Designed to provide object-oriented facilities to users of non object-oriented programming languages (OOPLs) such as C or Pascal.

The user will create classes, objects, inheritance and so on and the database system will store and manage these objects and classes. This second approach, thus, turns non-OOPLs into OOPLs. A translation layer is required to map the objects created by user into objects of the database system.

Advantages of OODBMS

Enriched modeling capabilities

The object-oriented data model allows the 'real world' to be modeled more closely. The object, which encapsulates both state and behavior, is a more natural and realistic representation of real-world objects. An object can store all the relationships it has with other objects, including many-to-many relationships, and objects can be formed into complex objects that the traditional data models cannot cope with easily.

Extensibility

OODBMSs allow new data types to be built from existing types. The ability to factor out common properties of several classes and form them into a superclass that can be shared with subclasses can greatly reduce redundancy within system and, as we stated· at the start of this chapter, is regarded as one of the main advantages of object orientation. Further, the reusability of classes promotes faster development and easier maintenance of the database and its applications.

Capable of handling a large variety of data types

Unlike traditional databases (such as hierarchical, network or relational), the object oriented database are capable of storing different types of data, for example, pictures, voice video, including text, numbers and so on.

Removal of impedance mismatch

A single language interface between the Data Manipulation Language (DML) and the programming language overcomes the impedance mismatch. This eliminates many of the efficiencies that occur in mapping a declarative language such as SQL to an imperative 'language such as 'C'. Most OODBMSs provide a DML that is computationally complete compared with SQL, the 'standard language of RDBMSs.

More expressive query language

Navigational access from the object is the most common form of data access in an OODBMS. This is in contrast to the associative access of SQL (that is, declarative statements with selection based on one or more predicates). Navigational access is more suitable for handling parts explosion, recursive queries, and so on.

Support for schema evolution

The tight coupling between data and applications in an OODBMS makes schema evolution more feasible.

Support for long-duration, transactions

Current relational DBMSs enforce serializability on concurrent transactions to maintain database consistency. OODBMSs use a different protocol to handle the types of long-duration transaction that are common in many advanced database application.

Applicability to advanced database applications

There are many areas where traditional DBMSs have not been particularly successful, such as, Computer-Aided Design (CAD), Computer-Aided Software Engineering (CASE), Office Information System(OIS), and Multimedia Systems. The enriched modeling capabilities of OODBMSs have made them suitable for these applications.

Improved performance

There have been a number of benchmarks that have suggested OODBMSs provide significant performance improvements over relational DBMSs. The results showed an average 30-fold performance improvement for the OODBMS over the RDBMS.

Disadvantages of OODBMSs

There are following disadvantages of OODBMSs:

Lack of universal data model: There is no universally agreed data model for an OODBMS, and most models lack a theoretical foundation. This .disadvantage is seen as a significant drawback, and is comparable to pre-relational systems.

Lack of experience: In comparison to RDBMSs the use of OODBMS is still relatively limited. This means that we do not yet have the level of experience that we have with traditional systems. OODBMSs are still very much geared towards the programmer, rather than the naïve end-user. Also there is a resistance to the acceptance of the technology. While the OODBMS is limited to a small niche market, this problem will continue to exist

Lack of standards: There is a general lack of standards of OODBMSs. We have already mentioned that there is not universally agreed data model. Similarly, there is no standard object-oriented query language.

Competition: Perhaps one of the most significant issues that face OODBMS vendors is the competition posed by the RDBMS and the emerging ORDBMS products. These products have an established user base with significant experience available. SQL is an approved standard and the relational data model has a solid theoretical formation and relational products have many supporting tools to help .both end-users and developers.

Query optimization compromises encapsulations: Query optimization requires. An understanding of the underlying implementation to access the database efficiently. However, this compromises the concept of incrassation.

Locking at object level may impact performance Many OODBMSs use locking as the basis for concurrency control protocol. However, if locking is applied at the object level, locking of an inheritance hierarchy may be problematic, as well as impacting performance.

Complexity: The increased functionality provided by the OODBMS (such as the illusion of a single-level storage model, pointer sizzling, long-duration transactions, version management, and schema evolution--makes the system more complex than that of traditional DBMSs. In complexity leads to products that are more expensive and more difficult to use.

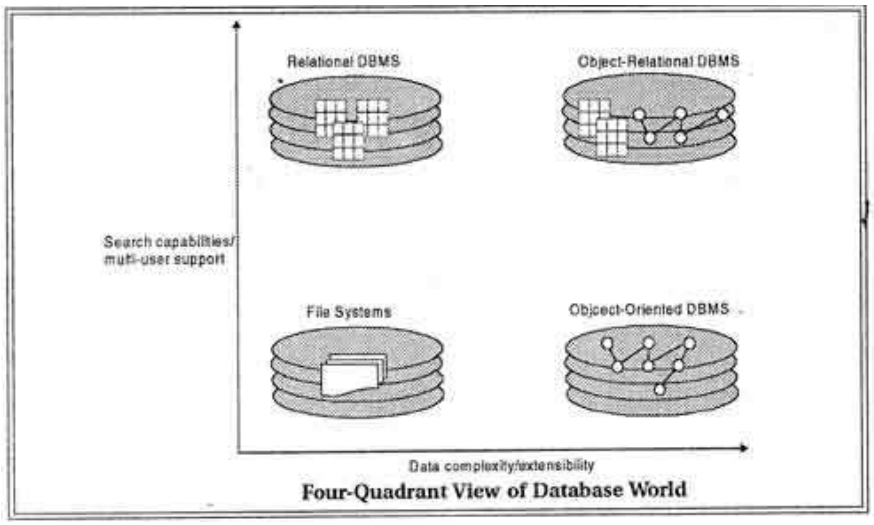
Lack of support for views: Currently, most OODBMSs do not provide a view mechanism, which, as we have seen previously, provides many advantages such as data independence, security, reduced complexity, and customization.

Lack of support for security: Currently, OODBMSs do not provide adequate security mechanisms. The user cannot grant access rights on individual objects or classes.

If OODBMSs are to expand fully into the business field, these deficiencies must be rectified.

OBJECT RELATIONAL DBMS

Relational DBMSs are currently the dominant database technology. The OODBMS has also become the favored system for financial and telecommunications applications. Although the OODBMS market is still same. The OODBMS continues to find new application areas, such as the World Wide Web. Some industry analysts expect the market for the OODBMSs to grow at over 50% per year, a rate faster than the total database market. However, their sales are unlikely to overtake those of relational systems because of the wealth of businesses that find RDBMSs acceptable, and because businesses have invested to much money and resources in their development that change is prohibitive. This is the approach that has been taken by many extended relational DBMSs, although each has implemented different combinations of features. Thus there is no single extended relational model rather, there are a variety of these models, whose characteristics depends upon the way and the degree to which extensions were made. However, all the models do share the same basic relational tables and query language, all incorporate some concept of 'object, and some have the ability to store methods (or procedures or triggers), as well as data in the database.



Four-Quadrant View of Database World

In a four-quadrant view of the database world, as illustrated in the figure, the lower-left quadrant are those applications that process simple data and have no requirements for querying the data.

These types of application, for example standard text processing packages such as Word,

WordPerfect, and Frame maker, can use the underlying operating system to obtain the essential DBMS functionality of persistence. In the lower-right quadrant are those applications that process complex data but again have no significant requirements for querying the data. For these types of application, for example computer-aided design packages, an OODBMS may be an appropriate choice of DBMS.

In the top-left quadrant are those applications that process simple data and also have requirements for complex querying. Many traditional business applications fall into this quadrant and an RDBMS may be the most appropriate DBMS.

Finally, in the top-right quadrant are those applications that process completed data and have complex querying requirements. This represents many of the advanced database applications and for these applications an ORDBMS may be the appropriate choice of DBMS.

Advantages and Disadvantages of ORDBMSS

ORDBMSs can provide appropriate solutions for many types of advanced database applications. However, there are also disadvantages.

Advantages of ORDBMSs

There are following advantages of ORDBMSs:

Reuse and Sharing: The main advantages of extending the Relational data model come from reuse and sharing. Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.

Increased Productivity: ORDBMS provides increased productivity both for the developer and for the, end user

Use of experience in developing RDBMS: Another obvious advantage is that .the extended relational approach preserves the significant body of knowledge and experience that has gone into developing relational applications. This is a significant advantage, as many organizations would find it prohibitively expensive to change. If the new functionality is designed appropriately, this approach should allow organizations to take advantage of the new extensions in an evolutionary way without losing the benefits of current database features and functions.

Disadvantages of ORDBMSs

The ORDBMS approach has the obvious disadvantages of complexity and associated increased costs. Further, there are the proponents of the relational approach that believe the· essential simplicity' and purity of the .relational model are lost with these types of extension.

ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexities. This potentially misses the point of object orientation, highlighting the large semantic gap between these two technologies. Object applications are simply not as data-centric as relational-based ones.

PARALLEL DATABASE

• A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries. Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel database improves processing and input/output speeds by using multiple CPUs and disks in

parallel. Centralized and client-server database systems are not powerful enough to handle such applications. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.

- A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries.
- Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel.
- Centralized and client–server database systems are not powerful enough to handle such applications.
- In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.
- Parallel databases can be roughly divided into two groups, the first group of architecture is the multiprocessor architecture, the alternatives of which are the followings :

- ✓ **Shared memory architecture,** where multiple processors share the main memory space.
- ✓ **Shared disk architecture**, where each node has its own main memory, but all nodes share mass storage, usually a storage area network. In practice, each node usually also has multiple processors.
- ✓ **Shared nothing architecture**, where each node has its own mass storage as well as main memory.

Distributed Database Architecture

A **distributed database system** allows applications to access data from local and remote databases. In a **homogenous distributed database system**, each database is an Oracle Database. In a **heterogeneous distributed database system**, at least one of the databases is not an Oracle Database. Distributed databases use client**/server** architecture to process information requests.

It contains the following database systems:

- Homogenous Distributed Database Systems
- Heterogeneous Distributed Database Systems
- Client/Server Database Architecture

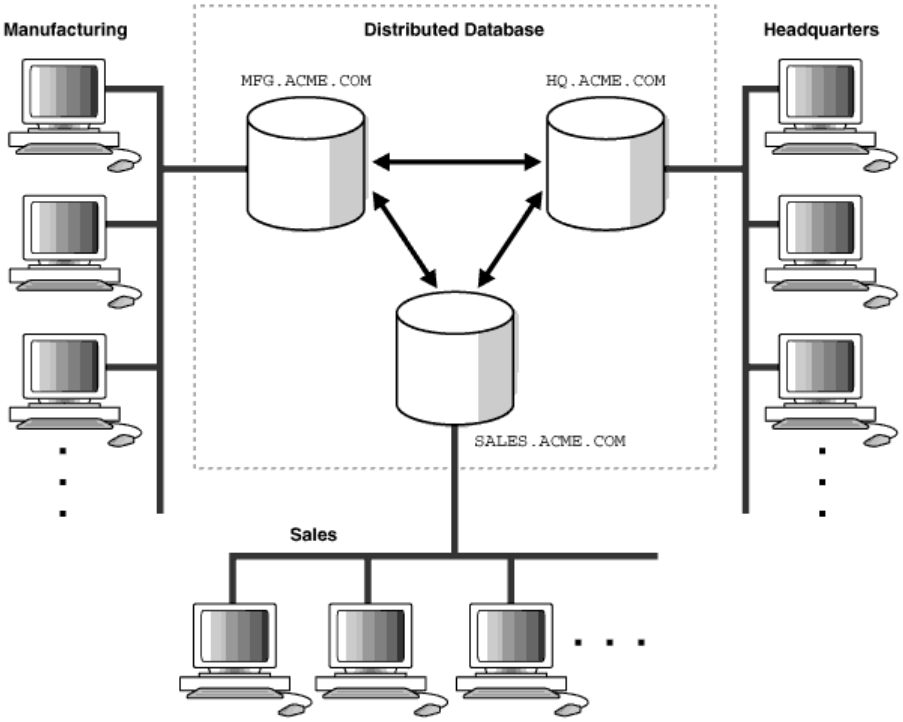
Homogenous Distributed Database Systems

A homogenous distributed database system is a network of two or more Oracle Databases that reside on one or more machines. Below Figure illustrates a distributed system that connects three databases: hq, mfg, and sales. An application can simultaneously access or modify the data in several databases in a single distributed environment. For example, a single query from a Manufacturing client on local database mfg can retrieve joined data from the products table on the local database and the dept table on the remote hq database.

For a client application, the location and platform of the databases are transparent. You can also create **synonyms** for remote objects in the distributed system so that users can access them with the same syntax as local objects. For example, if you are connected to database mfg but want to access data on database hq, creating a synonym on mfg for the remote dept table enables you to issue this query:

SELECT * FROM dept;

Homogeneous Distributed Database



An Oracle Database distributed database system can incorporate Oracle Databases of different versions. All supported releases of Oracle Database can participate in a distributed database system. Nevertheless, the applications that work with the distributed database must understand the functionality that is available at each node in the system. A distributed database application cannot expect an Oracle7 database to understand the SQL extensions that are only available with Oracle Database.

Heterogeneous Distributed Database Systems

In a heterogeneous distributed database system, at least one of the databases is a non-Oracle Database system. To the application, the heterogeneous distributed database system appears as a single, local, Oracle Database. The local Oracle Database server hides the distribution and heterogeneity of the data.

The Oracle Database server accesses the non-Oracle Database system using Oracle Heterogeneous Services in conjunction with an **agent**. If you access the non-Oracle Database data store using an Oracle Transparent Gateway, then the agent is a system-specific application. For example, if you include a Sybase database in an Oracle Database distributed system, then you need to obtain a Sybase-specific transparent gateway so that the Oracle Database in the system can communicate with it.

Alternatively, you can use **generic connectivity** to access non-Oracle Database data stores so long as the non-Oracle Database system supports the ODBC or OLE DB protocols.

Heterogeneous Services

Heterogeneous Services (HS) is an integrated component within the Oracle Database server and the enabling technology for the current suite of Oracle Transparent Gateway products. HS provides the common architecture and administration mechanisms for Oracle Database gateway products and other heterogeneous access facilities. Also, it provides upwardly compatible functionality for users of most of the earlier Oracle Transparent Gateway releases.

Transparent Gateway Agents

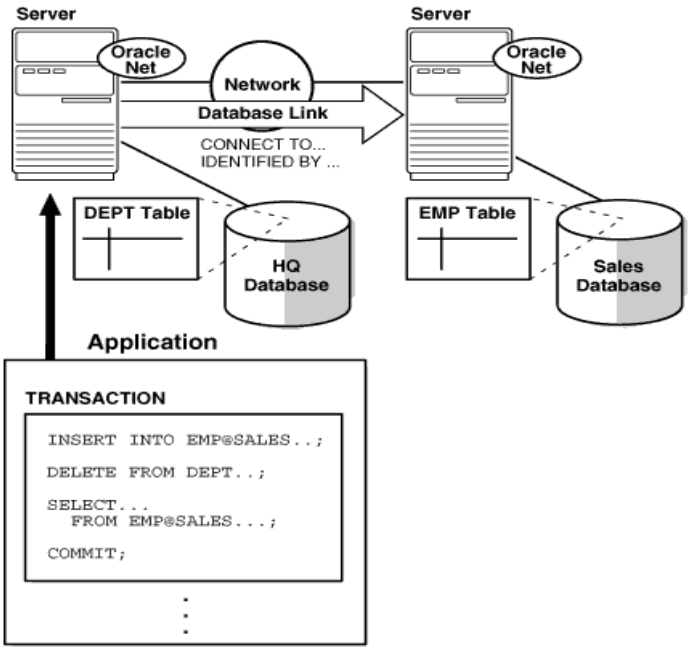
For each non-Oracle Database system that you access, Heterogeneous Services can use a transparent gateway agent to interface with the specified non-Oracle Database system. The agent is specific to the non-Oracle Database system, so each type of system requires a different agent.

The transparent gateway agent facilitates communication between Oracle Database and non-Oracle Database systems and uses the Heterogeneous Services component in the Oracle Database server. The agent executes SQL and transactional requests at the non-Oracle Database system on behalf of the Oracle Database server.

Client/Server Database Architecture

A database server is the Oracle software managing a database, and a client is an application that requests information from a server. Each computer in a network is a node that can host one or more databases. Each node in a distributed database system can act as a client, a server, or both, depending on the situation.

An Oracle Database Distributed Database System



A client can connect **directly** or **indirectly** to a database server. A direct connection occurs when a client connects to a server and accesses information from a database contained on that server. For example, if you connect to the hq database and access the dept table on this database as in below Figure, you can issue the following:

SELECT * FROM dept;

This query is direct because you are not accessing an object on a remote database.

In contrast, an indirect connection occurs when a client connects to a server and then accesses information contained in a database on a different server. For example, if you connect to the hq database but access the emp table on the remote sales database as in above Figure_you can issue the following:

SELECT * FROM emp@sales;

PARALLEL VS. DISTRIBUTED DATABASE :

Parallel Database System seeks to improve performance through parallelization of various operations, such as data loading, index building and query evaluating. Although data may be stored in a distributed fashion in such a system, the distribution is governed solely by performance considerations.

In **Distributed Database System**, data is physically stored across several sites, and each site is

typically managed by a DBMS capable of running independent of the other sites. In contrast to parallel databases, the distribution of data is governed by factors such as local ownership and increased availability.

PDB & DDB Comparison:

1. System Components
- Distributed DBMS consists of many **Geo-distributed**, **low-bandwidth link connected**, **autonomic** sites.
- Parallel DBMS consists of **tightly coupled**, **high-bandwidth link connected**, **non-autonomic** nodes.

2. Component Role
- Sites in Distributed DBMS can work independently to handle **local** transactions or work together to handle **global** transactions.
- Nodes in Parallel DBMS can only work together to handle **global** transactions.

3. Design Purposes
= Distributed DBMS is for:
- **Sharing Data**
- **Local Autonomy**
- **High Availability**

= Parallel DBMS is for:
- **High Performance**
- **High Availability**

But both PDB&DDB need to consider the following problems:

1. Data Distribution (Placement & Replicatioin);

2. Query Parallelization(Distributed Evaluation). And also, many parallel system consists of network of workstation, the difference between Parallel DB & Distributed DB is becoming smaller.

DATA WAREHOUSING

A data warehouse is a collection of data marts representing historical data from different operations in the company. This data is stored in a structure optimized for querying and data analysis as a data warehouse. Table design, dimensions and organization should be consistent throughout a data warehouse so that reports or queries across the data warehouse are consistent.

A data warehouse can also be viewed as a database for historical data from different functions within a company. The term Data Warehouse was coined by Bill Inmon in 1990, which he defined in the following way: "A warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process". He defined the terms in the sentence as follows: Subject Oriented: Data that gives information about a particular subject instead of about a company's ongoing operations.

Integrated: Data that is gathered into the data warehouse from a variety of sources and merged into a coherent whole.

Time-variant: All data in the data warehouse is identified with a particular time period.

Non-volatile: Data is stable in a data warehouse. More data is added but data is never removed. This enables management to gain a consistent picture of the business. It is a single, complete and consistent store of data obtained from a variety of different sources made available to end users in what they can understand and use in a business context. It can be

• Used for decision Support
• Used to manage and control business
• Used by managers and end-users to understand the business and make judgments

Benefits of data warehousing

• Data warehouses are designed to perform well with aggregate queries running on large amounts of data.
• The structure of data warehouses is easier for end users to navigate, understand and query against unlike the relational databases primarily designed to handle lots of transactions.
• Data warehouses enable queries that cut across different segments of a company's operation. E.g. production data could be compared against inventory data even if they were originally stored in different databases with different structures.
• Queries that would be complex in very normalized databases could be easier to build and maintain in data warehouses, decreasing the workload on transaction systems.
• Data warehousing is an efficient way to manage and report on data that is from a variety of sources, non uniform and scattered throughout a company.
• Data warehousing is an efficient way to manage demand for lots of information from lots of users.
• Data warehousing provides the capability to analyze large amounts of historical data for nuggets of wisdom that can provide an organization with competitive advantage.

Data Warehouse Characteristics

• A data warehouse can be viewed as an information system with the following attributes:

– It is a database designed for analytical tasks

– It’s content is periodically updated

– It contains current and historical data to provide a historical perspective of information

Data warehouse admin and management

The management of data warehouse includes,

• Security and priority management

• Monitoring updates from multiple sources

• Data quality checks

• Managing and updating meta data

• Auditing and reporting data warehouse usage and status

• Purging data

• Replicating, sub setting and distributing data

• Backup and recovery

• Data warehouse storage management which includes capacity planning, hierarchical storage management and purging of aged data etc..

DESIGN OF DATA WAREHOUSE

The following nine-step method is followed in the design of a data warehouse:

1. Choosing the subject matter

2. Deciding what a fact table represents

3. Identifying and conforming the dimensions

4. Choosing the facts

5. Storing pre calculations in the fact table

6. Rounding out the dimension table

7. Choosing the duration of the db

8. The need to track slowly changing dimensions

9. Deciding the query priorities and query models

Technical considerations

A number of technical issues are to be considered when designing a data warehouse environment. These issues include:

• The hardware platform that would house the data warehouse

• The dbms that supports the warehouse data

• The communication infrastructure that connects data marts, operational systems and end users

• The hardware and software to support meta data repository

• The systems management framework that enables admin of the entire environment

Implementation considerations

The following logical steps needed to implement a data warehouse:

• Collect and analyze business requirements

• Create a data model and a physical design

• Define data sources

• Choose the db tech and platform

• Extract the data from operational db, transform it, clean it up and load it into the warehouse

• Choose db access and reporting tools

• Choose db connectivity software

• Choose data analysis and presentation s/w

• Update the data warehouse

Access tools

Data warehouse implementation relies on selecting suitable data access tools. The best way to choose this is based on the type of data can be selected using this tool and the kind of access it

permits for a particular user. The following lists the various type of data that can be accessed:

• Simple tabular form data

• Ranking data

• Multivariable data

• Time series data

• Graphing, charting and pivoting data

• Complex textual search data

• Statistical analysis data

• Data for testing of hypothesis, trends and patterns

• Predefined repeatable queries

• Ad hoc user specified queries

• Reporting and analysis data

• Complex queries with multiple joins, multi level sub queries and sophisticated search criteria

Data extraction, clean up, transformation and migration

A proper attention must be paid to data extraction which represents a success factor for a data warehouse architecture. When implementing data warehouse several the following selection criteria that affect the ability to transform, consolidate, integrate and repair the data should be considered:

• Timeliness of data delivery to the warehouse

• The tool must have the ability to identify the particular data and that can be read by conversion tool

• The tool must support flat files, indexed files since corporate data is still in this type

• The tool must have the capability to merge data from multiple data stores

• The tool should have specification interface to indicate the data to be extracted

• The tool should have the ability to read data from data dictionary

• The code generated by the tool should be completely maintainable

• The tool should permit the user to extract the required data

• The tool must have the facility to perform data type and character set translation

• The tool must have the capability to create summarization, aggregation and derivation of records

• The data warehouse database system must be able to perform loading data directly from

these tools

Data placement strategies

– As a data warehouse grows, there are at least two options for data placement. One is to put some of the data in the data warehouse into another storage media.

– The second option is to distribute the data in the data warehouse across multiple servers.

User levels

The users of data warehouse data can be classified on the basis of their skill level in accessing the warehouse. There are three classes of users:Casual users: are most comfortable in retrieving info from warehouse in pre defined formats and running pre existing queries and reports. These users do not need tools that allow for building standard and ad hoc reports

Power Users: can use pre defined as well as user defined queries to create simple and ad hoc reports. These users can engage in drill down operations. These users may have the experience of using reporting and query tools.

Expert users: These users tend to create their own complex queries and perform standard analysis on the info they retrieve. These users have the knowledge about the use of query and report tools

Benefits of data warehousing

Data warehouse usage includes,

– Locating the right info

– Presentation of info

– Testing of hypothesis

– Discovery of info

– Sharing the analysis

The benefits can be classified into two:

• Tangible benefits (quantified / measureable):It includes,

– Improvement in product inventory

– Decrement in production cost

– Improvement in selection of target markets

– Enhancement in asset and liability management

• Intangible benefits (not easy to quantified): It includes,

– Improvement in productivity by keeping all data in single location and eliminating

– Reduced redundant processing

– Enhanced customer relation rekeying of data

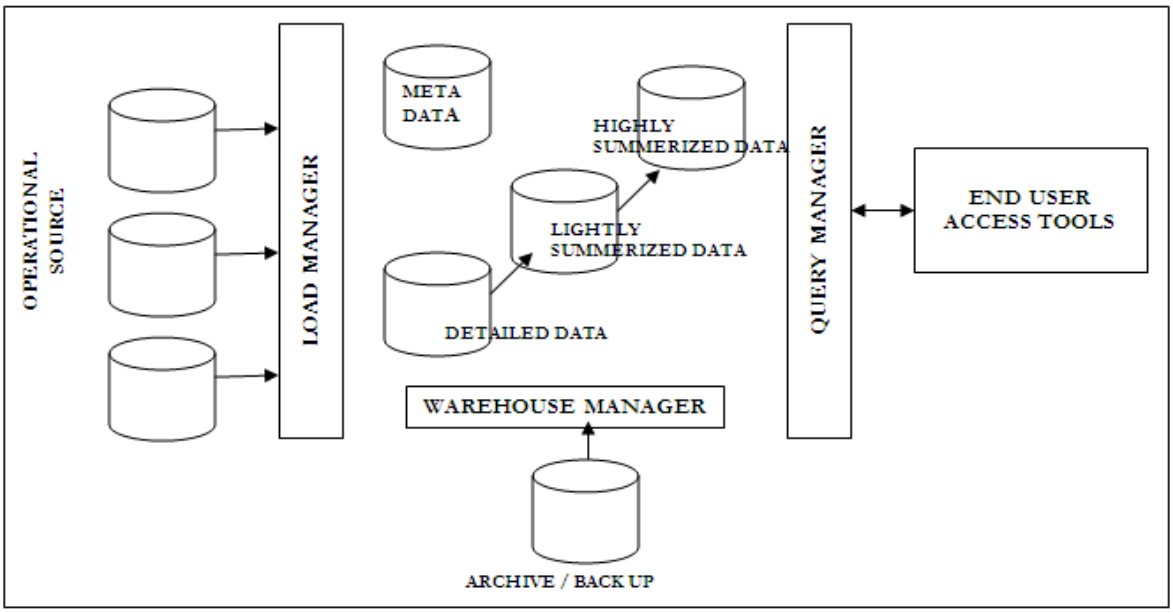
ARCHITECTURE OF DATA WAREHOUSING

The data in a data warehouse comes from operational systems of the organization as well as from other external sources. These are collectively referred to as ***source systems***. The data ***extracted*** from source systems is stored in a area called ***data staging area***, where the data is cleaned, ***transformed***, combined, duplicated to prepare the data for us in the data warehouse. The data staging area is generally a collection of machines where simple activities like sorting and sequential processing takes place. The data staging area does not provide any query or presentation services. As soon as a system provides query or presentation services, it is categorized as a ***presentation server***. A presentation server is the target machine on which the data is ***loaded*** from the data staging area organized and stored for direct querying by end users, report writers and other applications. The three different kinds of systems that are required for a data warehouse are:

1. Source Systems
2. Data Staging Area
3. Presentation servers

The data travels from source systems to presentation servers via the data staging area. The entire process is popularly known as ETL (extract, transform, and load) or ETT (extract, transform, and transfer). Oracle's ETL tool is called Oracle Warehouse Builder (OWB) and MS SQL Server's ETL tool is called Data Transformation Services (DTS).

A typical architecture of a data warehouse is shown below:



Each component and the tasks performed by them are explained below:

1. OPERATIONAL DATA

The sources of data for the data warehouse is supplied from:

(i) The data from the mainframe systems in the traditional network and hierarchical format.
(ii) Data can also come from the relational DBMS like Oracle, Informix.
(iii) In addition to these internal data, operational data also includes external data obtained from commercial databases and databases associated with supplier and customers.

2. LOAD MANAGER

The load manager performs all the operations associated with extraction and loading data into the data warehouse. These operations include simple transformations of the data to prepare the data for entry into the warehouse. The size and complexity of this component will vary between data warehouses and may be constructed using a combination of vendor data loading tools and custom built programs.

3. WAREHOUSE MANAGER

The warehouse manager performs all the operations associated with the management of data in the warehouse. This component is built using vendor data management tools and custom built programs. The operations performed by warehouse manager include:

(i) Analysis of data to ensure consistency
(ii) Transformation and merging the source data from temporary storage into data warehouse tables
(iii) Create indexes and views on the base table.
(iv) Denormalization
(v) Generation of aggregation
(vi) Backing up and archiving of data

In certain situations, the warehouse manager also generates query profiles to determine which indexes ands aggregations are appropriate.

4. QUERY MANAGER

The query manager performs all operations associated with management of user queries. This component is usually constructed using vendor end-user access tools, data warehousing monitoring tools, database facilities and custom built programs. The complexity of a query manager is determined by facilities provided by the end-user access tools and database.

5. DETAILED DATA

This area of the warehouse stores all the detailed data in the database schema. In most cases detailed data is not stored online but aggregated to the next level of details. However the detailed data is added regularly to the warehouse to supplement the aggregated data.

6. LIGHTLY AND HIGHLY SUMMERIZED DATA

The area of the data warehouse stores all the predefined lightly and highly summarized (aggregated) data generated by the warehouse manager. This area of the warehouse is transient as it will be subject to change on an ongoing basis in order to respond to the changing query profiles. The purpose of the summarized information is to speed up the query performance. The summarized data is updated continuously as new data is loaded into the warehouse.

7. ARCHIVE AND BACK UP DATA

This area of the warehouse stores detailed and summarized data for the purpose of archiving and back up. The data is transferred to storage archives such as magnetic tapes or optical disks.

8. META DATA

The data warehouse also stores all the Meta data (data about data) definitions used by all processes in the warehouse. It is used for variety of purposed including:

(i) The extraction and loading process – Meta data is used to map data sources to a common view of information within the warehouse.
(ii) The warehouse management process – Meta data is used to automate the production of summary tables.
(iii) As part of Query Management process Meta data is used to direct a query to the most appropriate data source.

The structure of Meta data will differ in each process, because the purpose is different. More about Meta data will be discussed in the later Lecture Notes.

9. END-USER ACCESS TOOLS

The principal purpose of data warehouse is to provide information to the business managers for strategic decision-making. These users interact with the warehouse using end user access tools. The examples of some of the end user access tools can be:

(i) Reporting and Query Tools
(ii) Application Development Tools
(iii) Executive Information Systems Tools
(iv) Online Analytical Processing Tools
(v) Data Mining Tools

THE E T L (EXTRACT TRANSFORMATION LOAD) PROCESS

In this section we will discussed about the 4 major process of the data warehouse. They are **extract** (data from the operational systems and bring it to the data warehouse), **transform** (the data into internal format and structure of the data warehouse), **cleanse** (to make sure it is of sufficient quality to be used for decision making) and **load** (cleanse data is put into the data warehouse).
The four processes from extraction through loading often referred collectively as **Data Staging**.

EXTRACT

Some of the data elements in the operational database can be reasonably be expected to be useful in the decision making, but others are of less value for that purpose. For this reason, it is necessary to extract the relevant data from the operational database before bringing into the data warehouse. Many commercial tools are available to help with the extraction process. **Data Junction** is one of the commercial products. The user of one of these tools typically has an easy-to-use windowed interface by which to specify the following:

(i) Which files and tables are to be accessed in the source database?
(ii) Which fields are to be extracted from them? This is often done internally by SQL Select statement.
(iii) What are those to be called in the resulting database?
(iv) What is the target machine and database format of the output?
(v) On what schedule should the extraction process be repeated?

TRANSFORM

The operational databases developed can be based on any set of priorities, which keeps changing with the requirements. Therefore those who develop data warehouse based on these databases are typically faced with inconsistency among their data sources. Transformation process deals with rectifying any inconsistency (if any).

One of the most common transformation issues is 'Attribute Naming Inconsistency'. It is common for the given data element to be referred to by different data names in different databases. Employee Name may be EMP_NAME in one database, ENAME in the other. Thus one set of Data Names are picked and used consistently in the data warehouse. Once all the data elements have right names, they must be converted to common formats. The conversion may encompass the following:

(i) Characters must be converted ASCII to EBCDIC or vise versa.
(ii) Mixed Text may be converted to all uppercase for consistency.
(iii) Numerical data must be converted in to a common format.
(iv) Data Format has to be standardized.
(v) Measurement may have to convert. (Rs/ $)
(vi) Coded data (Male/ Female, M/F) must be converted into a common format.

All these transformation activities are automated and many commercial products are available to perform the tasks. **DataMAPPER** from Applied Database Technologies is one such comprehensive tool.

CLEANSING

Information quality is the key consideration in determining the value of the information. The developer of the data warehouse is not usually in a position to change the quality of its underlying historic data, though a data warehousing project can put spotlight on the data quality issues and lead to improvements for the future. It is, therefore, usually necessary to go through the data entered into the data warehouse and make it as error free as possible. This process is known as **Data Cleansing**.

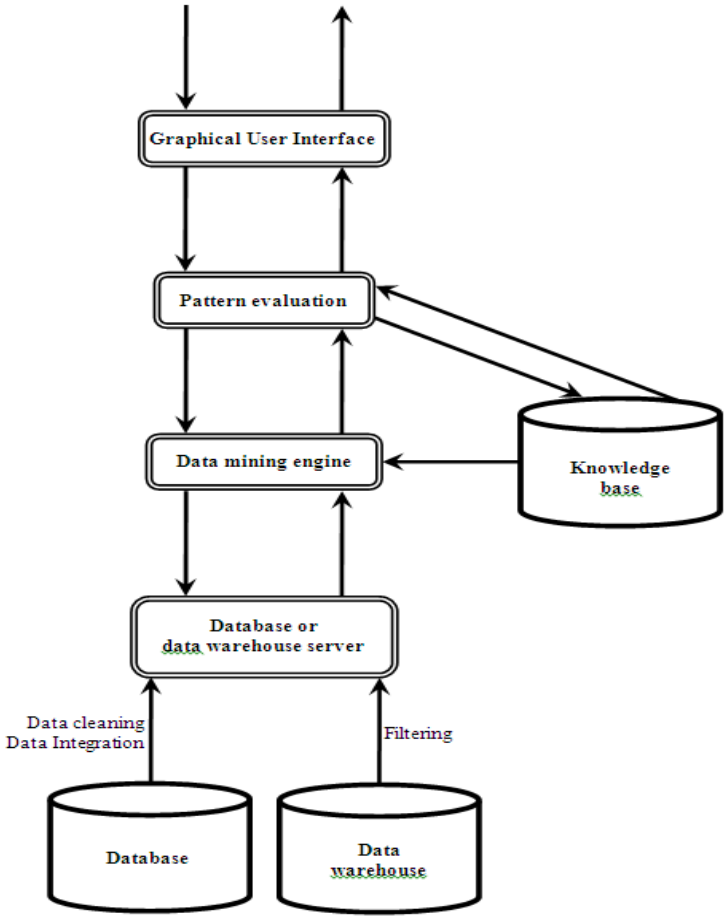
Data Cleansing must deal with many types of possible errors. These include missing data and incorrect data at one source; inconsistent data and conflicting data when two or more source are involved. There are several algorithms followed to clean the data, which will be discussed in the coming lecture notes.

LOADING

Loading often implies physical movement of the data from the computer(s) storing the source database(s) to that which will store the data warehouse database, assuming it is different. This takes place immediately after the extraction phase. The most common channel for data movement is a high-speed communication link. Ex: Oracle Warehouse Builder is the API from Oracle, which provides the features to perform the ETL task on Oracle Data Warehouse.

DATA MINING

Architecture of a Data Mining System



The architecture of a typical data mining system may have the following major components .

- **Database, data warehouse, or other information repository:** This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories. Data cleaning and data integration techniques may be performed on the data.

- **Database or data warehouse server:** The database or data warehouse server is responsible for fetching the relevant data, based on the user's data mining request.
- **Knowledge base:** This is the domain knowledge that is used to guide the search, or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize

attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern's interestingness based on its unexpectedness, may also be included. Other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

- **Data mining engine:** This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association, classification, cluster analysis, and evolution and deviation analysis.
- **Pattern evaluation module:** This component typically employs interestingness measures and interacts with the data mining modules so as to focus the search towards interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.
- **Graphical user interface:** This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

Functions of Data Mining

Data mining identifies facts or suggests conclusions based on sifting through the data to discover either patterns or anomalies. Data mining has five main functions:

- **Classification:** infers the defining characteristics of a certain group (such as customers who have been lost to competitors).

- **Clustering:** identifies groups of items that share a particular characteristic. (Clustering differs from classification in that no predefining characteristic is given in classification.)

- **Association:** identifies relationships between events that occur at one time (such as the contents of a shopping basket).

- **Sequencing:** similar to association, except that the relationship exists over a period of time (such as repeat visits to a supermarket or use of a financial planning product).

- **Forecasting:** estimates future values based on patterns within large sets of data (such as demand forecasting).

Data Mining Applications

The areas where data mining has been applied recently include:

- Science
  - astronomy,
  - bioinformatics,
  - drug discovery, ...
- Business
  - advertising,
  - Customer modeling and CRM (Customer Relationship management)
  - e-Commerce,
  - fraud detection
  - health care, ...
  - investments,
  - manufacturing,
  - sports/entertainment,
  - telecom (telephone and communications),
  - targeted marketing,
- Web:
  - search engines, bots, ...
- Government
  - anti-terrorism efforts
  - law enforcement,
  - profiling tax cheaters

One of the most important and widespread business applications of data mining is Customer Modeling, also called Predictive Analytics. This includes tasks such as

- predicting attrition or churn, i.e. find which customers are likely to terminate service
- targeted marketing:
  - customer acquisition - find which prospects are likely to become customers
  - cross-sell - for given customer and product, find which other product(s) they are likely to buy
- credit-risk - identify the risk that this customer will not pay back the loan or credit card
- fraud detection - is this transaction fraudulent?

The largest users of Customer Analytics are industries such as banking, telecom, retailers, where businesses with large numbers of customers are making extensive use of these technologies.