Chapter 11: Advanced Database Concepts

The **ODBMS** which is an abbreviation for **object oriented database management system**, is the data model in which data is stored in form of objects, which are instances of classes. These classes and objects together makes an object oriented data model.

Components of Object Oriented Data Model:

The OODBMS is based on three major components, namely: Object structure, Object classes, and Object identity. These are explained as following below.

1. Object Structure:

The structure of an object refers to the properties that an object is made up of. These properties of an object are referred to as an attribute. Thus, an object is a real world entity with certain attributes that makes up the object structure. Also an object encapsulates the data code into a single unit which in turn provides data abstraction by hiding the implementation details from the user.

The object structure is further composed of three types of components: Messages, Methods, and Variables. These are explained as following below.

a. Messages

A message provides an interface or acts as a communication medium between an object and the outside world. A message can be of two types:

- **Read-only message:** If the invoked method does not change the value of a variable, then the invoking message is said to be a read-only message.

- **Update message:** If the invoked method changes the value of a variable, then the invoking message is said to be an update message.

b. Methods

When a message is passed then the body of code that is executed is known as a method. Every time when a method is executed, it returns a value as output. A method can be of two types:

- **Read-only method:** When the value of a variable is not affected by a method, then it is known as read-only method.

- **Update-method:** When the value of a variable changes by a method, then it is known as an update method.

c. Variables

It stores the data of an object. The data stored in the variables makes the object distinguishable from one another.

2. Object Classes:

An object which is a real world entity is an instance of a class. Hence first we need to define a class and then the objects are made which differ in the values they store but share the same class definition. The objects in turn corresponds to various messages and variables stored in it.

3. Object Identity

Object identity is a fundamental object orientation concept. With object identity, objects can contain or refer to other objects. Identity is a property of an object that distinguishes the object from all other objects in the application.

To identify an object with a unique key (also called identifier keys) is a method that is commonly used in database management systems. Object identity is a stronger notion of identity than typically found in programming languages or in data models not based on object orientation.

Several forms of identity:

- **value**: A data value is used for identity (e.g., the primary key of a tuple in a relational database).
- **name**: A user-supplied name is used for identity (e.g., file name in a file system).
- **built-in**: A notion of identity is built-into the data model or programming languages, and no user-supplied identifier is required (e.g., in OO systems).

Object identity is typically implemented via a *unique*, *system-generated* OID. The value of the OID is not visible to the external user, but is used internally by the system to identify each object uniquely and to create and manage inter-object references.

There are many situations where having the system generate identifiers automatically is a benefit, since it frees humans from performing that task. However, this ability should be used with care. System-generated identifiers are usually specific to the system, and have to be translated if data are moved to a different database system.
System-generated identifiers may be redundant if the entities being modeled already have unique identifiers external to the system, e.g., SIN#.

Advantages of Object Oriented Database:

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

1. 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.

2. 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 super-class that can be shared with sub-classes can greatly reduce redundancy within system 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.

3. 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.

4. 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.

5. 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.

6. Support for schema evolution

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

7. 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.

8. 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.

9. 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 Object Oriented Database:

1. 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 per-relational systems.

2. 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

3. 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.

4. 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 incrustation.

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.

5. 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.

6. 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.

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

Popular Object Oriented Databases:

Here is a list of some of the popular object databases and their features.

1. Cache

InterSystems's Caché is a high-performance object database. Caché database engine is a set of services including data storage, concurrency management, transactions, and process management. You can think of the Caché engine as a powerful database toolkit.

Caché is also a full-featured relational database. All the data within a Caché database is available as true relational tables and can be queried and modified using standard SQL via ODBC, JDBC, or object methods. Caché is one of the fastest, most reliable, and most scalable relational databases.

2. ConceptBase

ConceptBase.cc is a multi-user deductive database system with an object-oriented (data, class, metaclass, meta-metaclass, etc.) makes it a powerful tool for metamodeling and engineering of customized modeling languages. The system is accompanied by a highly configurable graphical user interface that builds upon the logic-based features of the ConceptBase.cc server.

3. Db4o

b4o is the world's leading open-source object database for Java and .NET. Leverage fast native object persistence, ACID transactions, query-by-example, S.O.D.A object query API, automatic class schema evolution, small size.

4. ObjectDB Object Database

ObjectDB is a powerful Object-Oriented Database Management System (ODBMS). It is compact, reliable, easy to use and extremely fast. ObjectDB provides all the standard database management services (storage and retrieval, transactions, lock management, query processing, etc.) but in a way that makes development easier and applications faster.

5. ObjectDatabase++

ObjectDatabase++ (ODBPP) is an embeddable object-oriented database designed for server applications that require minimal external maintenance. It is written in C++ as a real-time ISAM level database with the ability to auto recover from system crashes while maintaining database integrity.

6. Objectivity/DB

Objectivity/DB is a scalable, high performance, distributed Object Database (ODBMS). It is extremely good at handling complex data, where there are many types of connections between objects and many variants.

Objectivity/DB runs on 32 or 64-bit processors running Linux, Mac OS X, UNIX (Oracle Solaris) or Windows.

There are C++, C#, Java and Python APIs.

All platform and language combinations are interoperable. For example, objects stored by a program using C++ on Linux can be read by a C# program on Windows and a Java program on Mac OS X.

Objectivity/DB generally runs on POSIX filesystems, but there are plugins that can be modified for other storage infrastructure.

Objectivity/DB client programs can be configured to run on a standalone laptop, networked workgroups, large clusters or in grids or clouds with no changes to the application code.

7. ObjectStore

ObjectStore is an enterprise object-oriented database management system for C++ and Java.

ObjectStore delivers multi-fold performance improvement by eliminating the middleware requirement to map and convert application objects into flat relational rows by directly persisting objects within an application into an object store

ObjectStore eliminates need to flatten complex data for consumption in your application logic reducing the overhead of using a translation layer that converts

complex objects into flat objects, dramatically improving performance and often entirely eliminating the need to manage a relational database system

ObjectStore is OO storage that directly integrates with Java or C++ applications and treats memory and persistent storage as one – improving the performance of application logic while fully maintaining ACID compliance against the transactional and distributed load.

8. Versant Object Database

Versant Object-Oriented Database is an object database that supports native object persistence and used to build complex and high-performance data management systems.

9. WakandaDB

WakandaDB is an object database and provides a native REST API to access interconnected DataClasses defined in Server-Side JavaScript. WakandaDB is the server within Wakanda which includes a dedicated, but not mandatory, Ajax Framework, and a dedicated IDE.

Object-Relational Databases Model:

Object-relational database (ORD), or object-relational database management systems (ORDBMS) are databases that support both objects and relational database features. OR databases are relational database management systems with the support of an object-oriented database model. That means, the entities are represented as objects and classes and OOP features such as inheritance are supported in database schemas and in the query language.

PostgreSQL is the most popular pure ORDBMS. Some popular databases including Microsoft SQL Server, Oracle, and IBM DB2 also support objects and can be considered as ORDBMS.

Advantages of Object Relational model

1. Inheritance

The Object Relational data model allows its users to inherit objects, tables etc. so that they can extend their functionality. Inherited objects contains new attributes as well as the attributes that were inherited.

2. Complex Data Types

Complex data types can be formed using existing data types. This is useful in Object relational data model as complex data types allow better manipulation of the data.

3. Extensibility

The functionality of the system can be extended in Object relational data model. This can be achieved using complex data types as well as advanced concepts of object oriented model such as inheritance.

Disadvantages of Object Relational model

The object relational data model can get quite complicated and difficult to handle at times as it is a combination of the Object oriented data model and Relational data model and utilizes the functionalities of both of them.

Distributed Database:

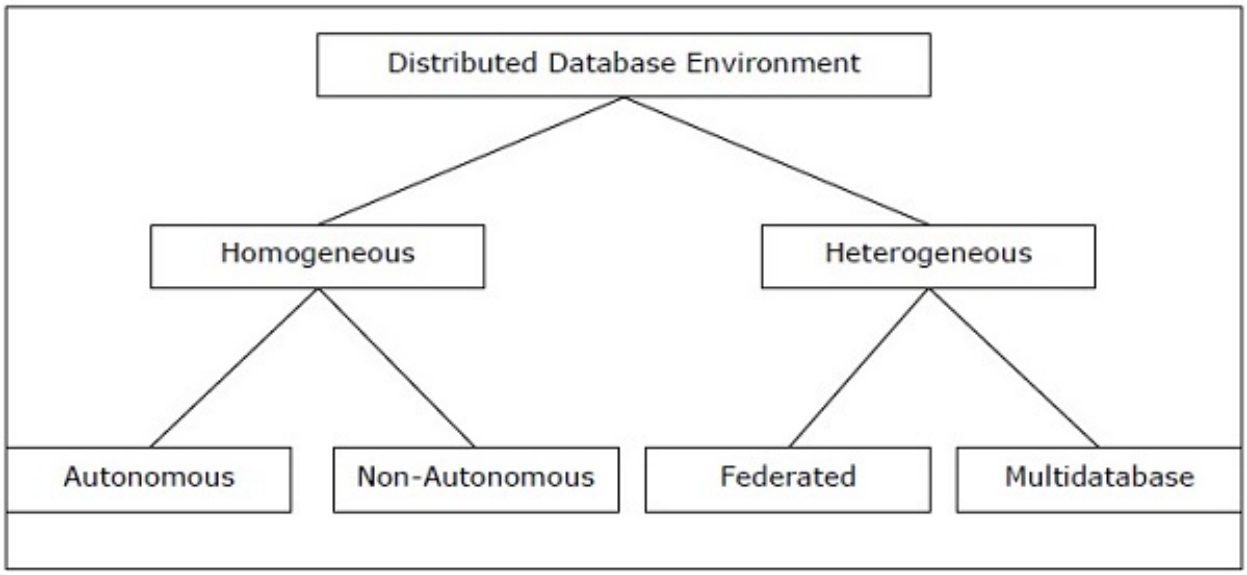
A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features of Distributed Database:

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



1. Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are –

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.
- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a single database.

2. Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are –

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Distributed Data Storage:

There are 2 ways in which data can be stored on different sites. These are:

1. Replication

In this approach, the entire relation is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.

This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.

However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency

This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. Fragmentation

In this approach, the relations are fragmented (i.e they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be must sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem. Fragmentation of relations can be done in two ways:

a. **Horizontal Fragmentation (Splitting By Rows):** The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.

b. **Vertical Fragmentation (Splitting By Columns):** The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure lossless join.

Advantages of Distributed Databases

1. Modular Development

If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning.

However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

2. More Reliable

In case of database failures, the total system of centralized databases comes to a halt.

However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

3. Better Response

If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response.

On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

4. Lower Communication Cost

In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

5. Improved Performance

As the data is located near the site of ‘greatest demand’, and given the inherent parallelism of distributed DBMSs, speed of database access may be better than that achievable from a remote centralized database.

Furthermore, since each site handles only a part of the entire database, there may not be the same contention for CPU and I/O services as characterized by a centralized DBMS.

6. Improved share ability and local autonomy

The geographical distribution of an organization can be reflected in the distribution of the data; users at one site can access data stored at other sites.

Data can be placed at the site close to the users who normally use that data.

In this way, users have local control of the data, and they can consequently establish and enforce local policies regarding the use of this data.

A global database administrator (DBA) is responsible for the entire system. Generally, part of this responsibility is assigned the local level, so that the local DBA can manage the local DBMS

Disadvantages of Distributed Database Management System

1. Complexity

A distributed DBMS that hides the distributed nature from the user and provides an acceptable level of performance, reliability, availability is inherently more complex then a centralized DBMS.

The fact that data can be replicated also adds an extra level of complexity to the distributed DBMS.

If the software does not handle data replication adequately, there will be degradation in availability, reliability and performance compared with the centralized system, and the advantages we cites above will become disadvantages.

2. Cost

Increased complexity means that we can expect the procurement and maintenance costs *for* a DDBMS to be higher than those for a centralized DBMS.

Furthermore, a distributed DBMS requires additional hardware to establish a network between sites. There are ongoing communication costs incurred with the use of this network.

There are also additional labor costs to manage and maintain the local DBMSs and the underlying network.

3. Security

In a centralized system, access to the data can be easily controlled.

However, in a distributed DBMS not only does access to replicated data have to be controlled in multiple locations but also the network itself has to be made secure.

In the past, networks were regarded as an insecure communication medium.

Although this is still partially true, significant developments have been made to make networks more secure.

4. Integrity control more difficult

Database integrity refers to the validity and consistency of stored data.

Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate.

Enforcing integrity constraints generally requires access to a large amount of data that defines the constraints.

In a distributed DBMS, the communication and processing costs that are required to enforce integrity constraints are high as compared to centralized system.

5. Lack of Standards

Although distributed DBMSs depend on effective communication, we are only now starting to see the appearance of standard communication and data access protocols.

This lack of standards has significantly limited the potential of distributed DBMSs.

There are also no tools or methodologies to help users convert a centralized DBMS into a distributed DBMS

6. Lack of experience

General-purpose distributed DBMSs have not been widely accepted, although many of the protocols and problems are well understood.

Consequently, we do not yet have the same level of experience in industry as we have with centralized DBMSs.

For a prospective adopter of this technology, this may be a significant deterrent.

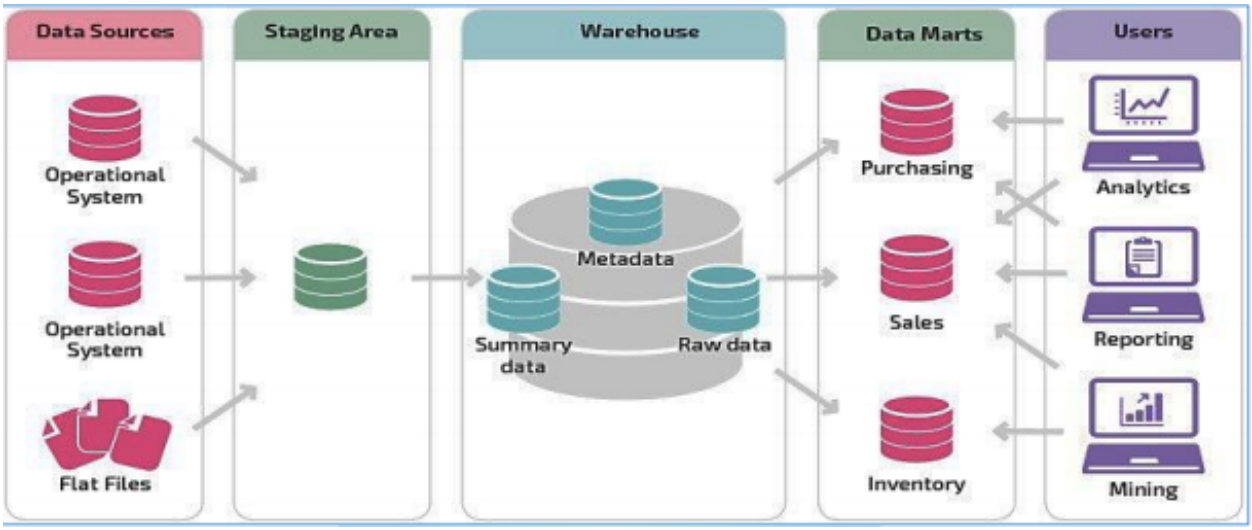
7. Database design more complex

Besides the normal difficulties of designing a centralized database, the design of a distributed database has to take account of fragmentation of data, allocation of fragmentation to specific sites, and data replication

Concepts of Data warehouse:

A **Data Warehousing** (DW) is process for collecting and managing data from varied sources to provide meaningful business insights. A Data warehouse is typically used to connect and analyze business data from heterogeneous sources. The data warehouse is the core of the BI system which is built for data analysis and reporting.

It is a blend of technologies and components which aids the strategic use of data. It is electronic storage of a large amount of information by a business which is designed for query and analysis instead of transaction processing. It is a process of transforming data into information and making it available to users in a timely manner to make a difference.



The decision support database (Data Warehouse) is maintained separately from the organization's operational database. However, the data warehouse is not a product but an environment. It is an architectural construct of an information system which provides users with current and historical decision support information which is difficult to access or present in the traditional operational data store.

You many know that a 3NF-designed database for an inventory system many have tables related to each other. For example, a report on current inventory information can include more than 12 joined conditions. This can quickly slow down the response time

of the query and report. A data warehouse provides a new design which can help to reduce the response time and helps to enhance the performance of queries for reports and analytics.

Data warehouse system is also known by the following name:

- Decision Support System (DSS)
- Executive Information System
- Management Information System
- Business Intelligence Solution
- Analytic Application
- Data Warehouse

How Data warehouse works?

A Data Warehouse works as a central repository where information arrives from one or more data sources. Data flows into a data warehouse from the transactional system and other relational databases.

Data may be:

1. Structured
2. Semi-structured
3. Unstructured data

The data is processed, transformed, and ingested so that users can access the processed data in the Data Warehouse through Business Intelligence tools, SQL clients, and spreadsheets. A data warehouse merges information coming from different sources into one comprehensive database.

By merging all of this information in one place, an organization can analyze its customers more holistically. This helps to ensure that it has considered all the information available. Data warehousing makes data mining possible. Data mining is looking for patterns in the data that may lead to higher sales and profits.

Types of Data Warehouse

Three main types of Data Warehouses (DWH) are:

1. Enterprise Data Warehouse (EDW):

Enterprise Data Warehouse (EDW) is a centralized warehouse. It provides decision support service across the enterprise. It offers a unified approach for organizing and representing data. It also provide the ability to classify data according to the subject and give access according to those divisions.

2. Operational Data Store:

Operational Data Store, which is also called ODS, are nothing but data store required when neither Data warehouse nor OLTP systems support organizations reporting needs. In ODS, Data warehouse is refreshed in real time. Hence, it is widely preferred for routine activities like storing records of the Employees.

3. Data Mart:

A data mart is a subset of the data warehouse. It specially designed for a particular line of business, such as sales, finance, sales or finance. In an independent data mart, data can collect directly from sources.

Components of Data warehouse

Four components of Data Warehouses are:

Load manager: Load manager is also called the front component. It performs with all the operations associated with the extraction and load of data into the warehouse. These operations include transformations to prepare the data for entering into the Data warehouse.

Warehouse Manager: Warehouse manager performs operations associated with the management of the data in the warehouse. It performs operations like analysis of data to ensure consistency, creation of indexes and views, generation of denormalization and aggregations, transformation and merging of source data and archiving and baking-up data.

Query Manager: Query manager is also known as backend component. It performs all the operation operations related to the management of user queries. The operations of this Data warehouse components are direct queries to the appropriate tables for scheduling the execution of queries.

End-user access tools:

This is categorized into five different groups like 1. Data Reporting 2. Query Tools 3. Application development tools 4. EIS tools, 5. OLAP tools and data mining tools.

Applications of Data Warehouse

Airline:

In the Airline system, it is used for operation purpose like crew assignment, analyses of route profitability, frequent flyer program promotions, etc.

Banking:

It is widely used in the banking sector to manage the resources available on desk effectively. Few banks also used for the market research, performance analysis of the product and operations.

Healthcare:

Healthcare sector also used Data warehouse to strategize and predict outcomes, generate patient's treatment reports, share data with tie-in insurance companies, medical aid services, etc.

Public sector:

In the public sector, data warehouse is used for intelligence gathering. It helps government agencies to maintain and analyze tax records, health policy records, for every individual.

Investment and Insurance sector:

In this sector, the warehouses are primarily used to analyze data patterns, customer trends, and to track market movements.

Retain chain:

In retail chains, Data warehouse is widely used for distribution and marketing. It also helps to track items, customer buying pattern, promotions and also used for determining pricing policy.

Telecommunication:

A data warehouse is used in this sector for product promotions, sales decisions and to make distribution decisions.

Hospitality Industry:

This Industry utilizes warehouse services to design as well as estimate their advertising and promotion campaigns where they want to target clients based on their feedback and travel patterns.

Steps to Implement Data Warehouse

The best way to address the business risk associated with a Data warehouse implementation is to employ a three-prong strategy as below

1. **Enterprise strategy**: Here we identify technical including current architecture and tools. We also identify facts, dimensions, and attributes. Data mapping and transformation is also passed.
2. **Phased delivery**: Data warehouse implementation should be phased based on subject areas. Related business entities like booking and billing should be first implemented and then integrated with each other.
3. **Iterative Prototyping**: Rather than a big bang approach to implementation, the Data warehouse should be developed and tested iteratively.

Advantages of Data Warehouse (DWH):

- Data warehouse allows business users to quickly access critical data from some sources all in one place.
- Data warehouse provides consistent information on various cross-functional activities. It is also supporting ad-hoc reporting and query.
- Data Warehouse helps to integrate many sources of data to reduce stress on the production system.
- Data warehouse helps to reduce total turnaround time for analysis and reporting.
- Restructuring and Integration make it easier for the user to use for reporting and analysis.
- Data warehouse allows users to access critical data from the number of sources in a single place. Therefore, it saves user's time of retrieving data from multiple sources.
- Data warehouse stores a large amount of historical data. This helps users to analyze different time periods and trends to make future predictions.

Disadvantages of Data Warehouse:

- Not an ideal option for unstructured data.
- Creation and Implementation of Data Warehouse is surely time confusing affair.
- Data Warehouse can be outdated relatively quickly
- Difficult to make changes in data types and ranges, data source schema, indexes, and queries.
- The data warehouse may seem easy, but actually, it is too complex for the average users.

- Despite best efforts at project management, data warehousing project scope will always increase.
- Sometime warehouse users will develop different business rules.
- Organizations need to spend lots of their resources for training and Implementation purpose.