Department of Computer Science

Aligarh Muslim University

Aligarh

COURSE CSM- 5541: ADVANCE DBMS AND DBA

Prof. Mohammad Ubaidullah Bokhari

Unit 1 – Part A.

Unit 1 Syllabus:

- Relational database
- Limitations of relational database
- Need of advanced DBMS
- Object Oriented Database
- Object Relational Databases
- Temporal Database concept
- Distributed Database Management: Reference Architecture, levels of distribution transparency, Distributed database design, Distributed Query Processing and Optimization
- Distributed Transaction Modeling.

Relational Database:

A *relational database* is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the *key*. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

The relational model means that the logical data structures—the data tables, views, and indexes—are separate from the physical storage structures. This separation means that database administrators can manage physical data storage without affecting access to that data as a logical structure. For example, renaming a database file does not rename the tables stored within it.

The distinction between logical and physical also applies to database operations, which are clearly defined actions that enable applications to manipulate the data and structures of the database. Logical operations allow an application to specify the content it needs, and physical operations determine how that data should be accessed and then carries out the task.

To ensure that data is always accurate and accessible, relational databases follow certain integrity rules. For example, an integrity rule can specify that duplicate rows are not allowed in a table in order to eliminate the potential for erroneous information entering the database.

In the early years of databases, every application stored data in its own unique structure. When developers wanted to build applications to use that data, they had to know a lot about the particular data structure to find the data they needed. These data structures were inefficient, hard to maintain, and hard to optimize for delivering good application performance. The relational database model was designed to solve the problem of multiple arbitrary data structures.

The relational model provided a standard way of representing and querying data that could be used by any application. From the beginning, developers recognized that the chief strength of the relational database model was in its use of tables, which were an intuitive, efficient, and flexible way to store and access structured information.

Over time, another strength of the relational model emerged as developers began to use structured query language (SQL) to write and query data in a database. For many years, SQL has been widely used as the language for database queries. Based on relational algebra, SQL provides an internally consistent mathematical language that makes it easier to improve the performance of all database queries. In comparison, other approaches must define individual queries.

Quick Questions:

Q1. WHAT IS A DATABASE?

A database is a set of data stored in a computer. This data is usually structured in a way that makes the data easily accessible.

Q2. WHAT IS A RELATIONAL DATABASE?

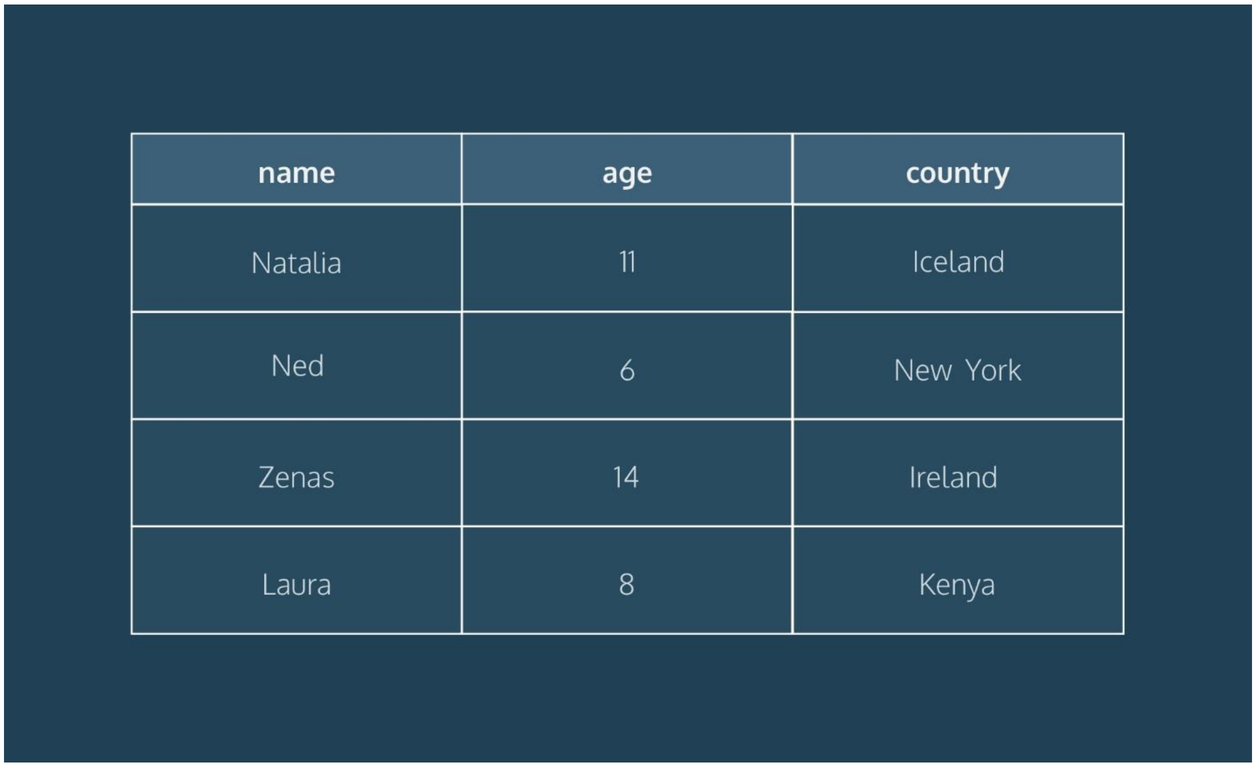
A relational database is a type of database. It uses a structure that allows us to identify and access data in relation to another piece of data in the database. Often, data in a relational database is organized into tables.

TABLES: ROWS AND COLUMNS

Tables can have hundreds, thousands, sometimes even millions of rows of data. These rows are often called records.

Tables can also have many columns of data. Columns are labeled with a descriptive name (say, age for example) and have a specific data type.

For example, a column called age may have a type of INTEGER (denoting the type of data it is meant to hold).



name	age	country
Natalia	11	Iceland
Ned	6	New York
Zenas	14	Ireland
Laura	8	Kenya

In the table above, there are three columns (name, age, and country).

The name and country columns store string data types, whereas agestores integer data types. The set of columns and data types make up the schema of this table.

The table also has four rows, or records, in it (one each for Natalia, Ned, Zenas, and Laura).

Q3. WHAT IS A RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)?

A relational database management system (RDBMS) is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database.

Q4. WHAT IS SQL?

SQL (**S**tructured **Q**uery **L**anguage) is a programming language used to communicate with data stored in a relational database management system. SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret.

Many RDBMSs use SQL (and variations of SQL) to access the data in tables. For example, SQLite is a relational database management system. SQLite contains a minimal set of SQL commands (which are the same across all RDBMSs). Other RDBMSs may use other variants.

(SQL is often pronounced in one of two ways. You can pronounce it by speaking each letter individually like “S-Q-L”, or pronounce it using the word “sequel”.)

POPULAR RELATIONAL DATABASE MANAGEMENT SYSTEMS

SQL syntax may differ slightly depending on which RDBMS you are using. Here is a brief description of popular RDBMSs:

MySQL

MySQL is the most popular open source SQL database. It is typically used for web application development, and often accessed using PHP.

The main advantages of MySQL are that it is easy to use, inexpensive, reliable (has been around since 1995), and has a large community of developers who can help answer questions.

Some of the disadvantages are that it has been known to suffer from poor performance when scaling, open source development has lagged since Oracle has taken control of MySQL, and it does not include some advanced features that developers may be used to.

PostgreSQL

PostgreSQL is an open source SQL database that is not controlled by any corporation. It is typically used for web application development.

PostgreSQL shares many of the same advantages of MySQL. It is easy to use, inexpensive, reliable and has a large community of developers. It also provides some additional features such as foreign key support without requiring complex configuration.

The main disadvantage of PostgreSQL is that it is slower in performance than other databases such as MySQL. It is also less popular than MySQL which makes it harder to come by hosts or service providers that offer managed PostgreSQL instances.

Oracle DB

Oracle Corporation owns Oracle Database, and the code is not open sourced.

Oracle DB is for large applications, particularly in the banking industry. Most of the world’s top banks run Oracle applications because Oracle offers a powerful combination of technology and comprehensive, pre-integrated business applications, including essential functionality built specifically for banks.

The main disadvantage of using Oracle is that it is not free to use like its open source competitors and can be quite expensive.

SQL Server

Microsoft owns SQL Server. Like Oracle DB, the code is close sourced.

Large enterprise applications mostly use SQL Server.

Microsoft offers a free entry-level version called *Express* but can become very expensive as you scale your application.

SQLite

SQLite is a popular open source SQL database. It can store an entire database in a single file. One of the most significant advantages this provides is that all of the data can be stored locally without having to connect your database to a server.

SQLite is a popular choice for databases in cellphones, PDAs, MP3 players, set-top boxes, and other electronic gadgets.

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

Advantages of using Relational model

- **Simplicity**: A relational data model is simpler than the hierarchical and network model.
- **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use**: The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
- **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence**: The structure of a database can be changed without having to change any application.
- **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Limitations of Relational Database:

Some of the key limitations of relational databases:

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.
- They tend to be slow and not scalable. If you have more servers you can’t always do more work with them.
- They have a fixed schema which is a plus unless this hurts productivity too much.
- Tables don’t always map to objects in applications very well.
- They are not secure enough to expose to the internet and need a layers to be added to protect them.
- They are not good at modelling certain kinds of data such as graphs and geo-spacial queries.
- They are not at storing very large records ie. 100s of MBs or GBs.

Relational databases are widely used in many industries to store financial records, keep track of inventory and to keep records on employees. In a relational database, information is stored in tables (often called relations) which help organize and structure data. Even though they are widely used, relational databases have some drawbacks. Detailed points over the bottlenecks of relational databases are discussed as below:

Cost

One disadvantage of relational databases is the expensive of setting up and maintaining the database system. In order to set up a relational database, you generally need to purchase special software. If you are not a programmer, you can use any number of products to set up a relational database. It does take time to enter in all the information and set up the program. If your company is large and you need a more robust database, you will need to hire a programmer to create a relational database using Structured Query Language (SQL) and a database administrator to maintain the database once it is built. Regardless of what data you use, you will have to either import it from other data like text files or Excel spreadsheets, or have the data entered at the keyboard. No matter the size of your company, if you store legally confidential or protected information in your database such as health information, social security numbers or credit card numbers, you will also have to secure your data against unauthorized access in order to meet regulatory standards.

Abundance of Information

Advances in the complexity of information cause another drawback to relational databases. Relational databases are made for organizing data by common characteristics. Complex images, numbers, designs and multimedia products defy easy categorization leading the way for a new type of database called object-relational database management systems. These systems are designed to handle the more complex applications and have the ability to be scalable.

Structured Limits

Some relational databases have limits on field lengths. When you design the database, you have to specify the amount of data you can fit into a field. Some names or search queries are shorter than the actual, and this can lead to data loss.

Isolated Databases

Complex relational database systems can lead to these databases becoming "islands of information" where the information cannot be shared easily from one large system to another. Often, with big firms or institutions, you find relational databases grew in separate divisions differently. For example, maybe the hospital billing department used one database while the hospital personnel department used a different database. Getting those databases to "talk" to each other can be a large, and expensive, undertaking, yet in a complex hospital system, all the databases need to be involved for good patient and employee care.

Need of Advance DBMS:

A **Database** is a collection of related data organized in a way that data can be easily accessed, managed and updated. Any piece of information can be a data, for example name of your school. Database is actually a place where related piece of information is stored and various operations can be performed on it.

A **DBMS** is a software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular DBMS, MySQL, Oracle, Sybase, Microsoft Access and IBM DB 2 etc.

DBMS and **advanced DBMS** are not two different things, but we learn advanced concepts of DBMS and those concepts are called as advanced DBMS.

List of things we get to learned in **DBMS**

1. DBMS - Introduction
2. Entity Relationship Model
3. Relational Data Model and Relational Algebra
4. SQL
5. File Organization and Database Index
6. Transactions
7. Concurrency Control and Recovery Management, etc.

List of things we get to learned in **Advanced DBMS**

1. Data Warehousing And Mining
2. Query Processing and Optimization
3. Distributed and Parallel Database
4. Relational Database Design & Normalization
5. Object Oriented and Object Relational Database
6. XML, etc.

Data Warehousing and Mining

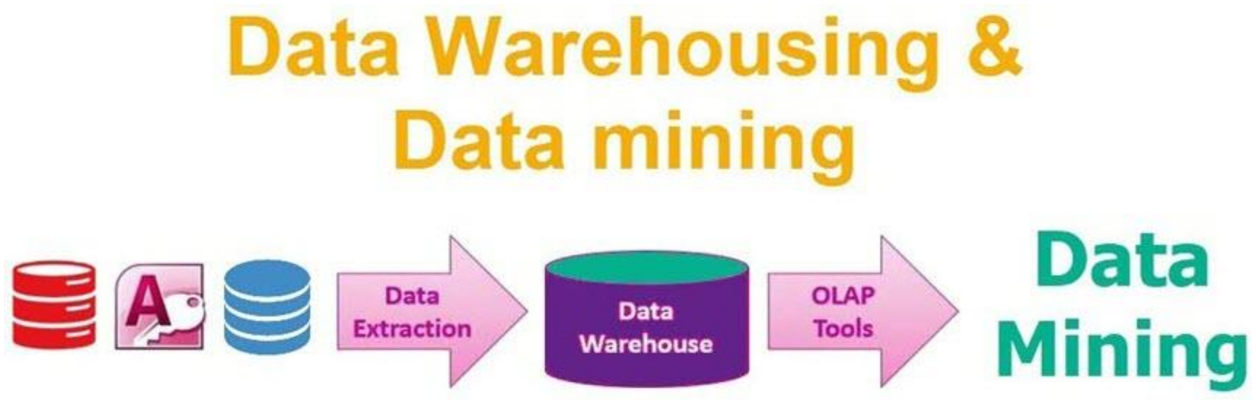
A data warehouse is a technique for collecting and managing data from varied sources to provide meaningful business insights. It is a blend of technologies and components which allows the strategic use of data.

Data Warehouse 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 for analysis.

Data mining is looking for hidden, valid, and potentially useful patterns in huge data sets. Data Mining is all about discovering unsuspected/ previously unknown relationships amongst the data.

It is a multi-disciplinary skill that uses machine learning, statistics, AI and database technology.

The insights extracted via Data mining can be used for marketing, fraud detection, and scientific discovery, etc.



Data Mining Vs Data Warehouse: Key Differences

Data Mining	**Data Warehouse**
Data mining is the process of analyzing unknown patterns of data.	A data warehouse is database system which is designed for analytical instead of transactional work.
Data mining is a method of comparing large amounts of data to finding right patterns.	Data warehousing is a method of centralizing data from different sources into one common repository.
Data mining is usually done by business users with the assistance of engineers.	Data warehousing is a process which needs to occur before any data mining can take place.
Data mining is the considered as a process of extracting data from large data sets.	On the other hand, Data warehousing is the process o pooling all relevant data together.
One of the most important benefits of data mining techniques is the detection and identification of errors in the system.	One of the pros of Data Warehouse is its ability to update consistently. That's why it is ideal for the business owner who wants the best and latest features

Data mining helps to create suggestive patterns of important factors. Like the buying habits of customers, products, sales. So that, companies can make the necessary adjustments in operation and production.	Data Warehouse adds an extra value to operational business systems like CRM systems when the warehouse is integrated.
The Data mining techniques are never 100% accurate and may cause serious consequences in certain conditions.	In the data warehouse, there is great chance that the data which was required for analysis by the organization may not be integrated into the warehous It can easily lead to loss of information.
The information gathered based on Data Mining by organizations can be misused against a group of people.	Data warehouses are created for a huge IT project. Therefore, it involves high maintenance system whicl can impact the revenue of medium to small-scale organizations.
After successful initial queries, users may ask more complicated queries which would increase the workload.	Data Warehouse is complicated to implement and maintain.
Organisations can benefit from this analytical tool by equipping pertinent and usable knowledge-based information.	Data warehouse stores a large amount of historical da which helps users to analyze different time periods ar trends for making future predictions.
Organisations need to spend lots of their resources for training and Implementation purpose. Moreover, data mining tools work in different manners due to different algorithms employed in their design.	In Data warehouse, data is pooled from multiple sources. The data needs to be cleaned and transforme This could be a challenge.
The data mining methods are cost-effective and efficient compares to other statistical data applications.	Data warehouse's responsibility is to simplify every type of business data. Most of the work that will be done on user's part is inputting the raw data.
Another critical benefit of data mining techniques is the identification of errors which can lead to losses. Generated data could be used to detect a drop-in sale.	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 mining helps to generate actionable strategies built on data insights.	Once you input any information into Data warehouse system, you will unlikely to lose track of this data again. You need to conduct a quick search, helps you to find the right statistic information.

Why use Data Warehouse?

Some most important reasons for using Data warehouse are:

- Integrates many sources of data and helps to decrease stress on a production system.
- Optimized Data for reading access and consecutive disk scans.
- Data Warehouse helps to protect Data from the source system upgrades.
- Allows users to perform master Data Management.
- Improve data quality in source systems.

Why use Data mining?

Some most important reasons for using Data mining are:

- Establish relevance and relationships amongst data. Use this information to generate profitable insights
- Business can make informed decisions quickly
- Helps to find out unusual shopping patterns in grocery stores.
- Optimize website business by providing customize offers to each visitor.
- Helps to measure customer's response rates in business marketing.
- Creating and maintaining new customer groups for marketing purposes.
- Predict customer defections, like which customers are more likely to switch to another supplier in the nearest future.
- Differentiate between profitable and unprofitable customers.
- Identify all kind of suspicious behavior, as part of a fraud detection process.

Query Processing and Optimization:

Query Processing is the scientific art of obtaining the desired information from a database system in a predictable and reliable fashion. Database systems must be able to respond to requests for information from the user i.e. process queries. In large database systems, which may be running on un-predictable and volatile environments, it is difficult to produce efficient database query plans based on information available solely at compile time. Getting the database results in a timely manner deals with the technique of Query Optimization. Efficient processing of queries is an important requirement in many interactive environments that involve massive amounts of data. Efficient query processing in domains such as the Web, multimedia search, and distributed systems has shown a great impact on performance. This paper will introduce the basic concepts of query processing and query optimization in the relational database. We also describe and difference query processing techniques in relational databases.

The fundamental part of any DBMS is query processing and optimization. The results of queries must be available in the timeframe needed by the submitting user. Query processing techniques based on multiple design dimensions can be classified as:
1. Query model: Processing techniques are classified according to the query model they assume. Some techniques assume a selection query model, where scores are attached directly to base tuples. Other techniques assume a join query model, where scores are computed over join results. A third category assumes an aggregate query model, where we are interested in ranking groups of tuples.
2. Data access methods: Processing techniques are classified according to the data access methods they assume to be available in the underlying data sources. For example, some techniques assume the availability of random access, while others are restricted to only sorted access.
3. Implementation level: Processing techniques are classified according to their level of integration with database systems. For example, some techniques are implemented in an application layer on top of the database system, while others are implemented as query operators.
4. Data and query uncertainty: Processing techniques are classified based on the uncertainty involved in their data and query models. Some techniques produce exact answers, while others allow for approximate answers, or deal with uncertain data.
5. Ranking function: Processing techniques are classified based on the restrictions they impose on the underlying ranking (scoring) function. Most proposed techniques assume monotone scoring functions.

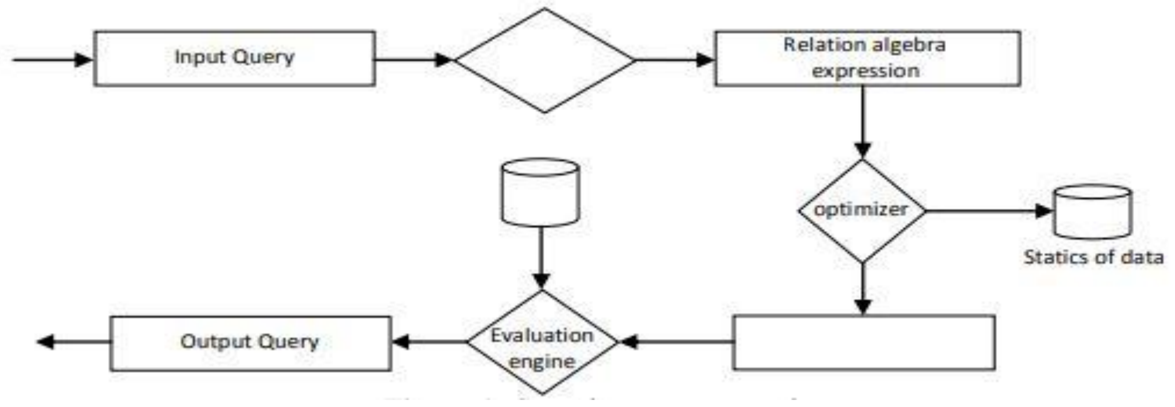
QUERY PROCESSING

Query processing refers to the range of activities involved in extracting data from a database. The activities include translation of queries in high-level database languages into expressions that can be used at the physical level of the file system, a variety of query-optimizing transformations, and actual evaluation of queries. A database query is the vehicle for instructing a DBMS to update or retrieve specific data to/from the physically stored medium. The actual updating and retrieval of data is performed through various “low- level” operations. Examples of such operations for a relational DBMS can be relational algebra operations such as project, join, select, Cartesian product, etc. While the DBMS is designed to process these low -level operations efficiently, it can be quite the burden to a user to submit requests to the DBMS in these formats. There are three phases that a query passes through during the DBMS' processing of that query:

1. Parsing and translation
2. Optimization
3. Evaluation

The first step in processing a query submitted to a DBMS is to convert the query into a form usable by the query processing engine. High- level query languages such as SQL represent a query as a string, or sequence, of characters. Certain sequences of characters represent various types of tokens such as keywords, operators, operands, literal strings, etc. Like all languages, there are rules (syntax and grammar) that govern how the tokens can be combined into understandable (i.e. valid) statements.

The primary job of the parser is to extract the tokens from the raw string of characters and translate them into the corresponding internal data elements (i.e. relational algebra operations and operands) and structures (i.e. query tree, query graph). The last job of the parser is to verify the validity and syntax of the original query string. In second stage, the query processor applies rules to the internal data structures of the query to transform these structures into equivalent, but more efficient representations. The rules can be based upon mathematical models of the relational algebra expression and tree (heuristics), upon cost estimates of different algorithms applied to operations or upon the semantics within the query and the relations it involves. Selecting the proper rules to apply, when to apply them and how they are applied is the function of the query optimization engine.



The final step in processing a query is the evaluation phase. The best evaluation plan candidate generated by the optimization engine is selected and then executed. Note that there can exist multiple methods of executing a query. Besides processing a query in a simple sequential manner, some of a query's individual operations can be processed in parallel either as independent processes or as interdependent pipelines of processes or threads. Regardless of the method chosen, the actual results should be same.

Distributed and Parallel Databases:

In centralized database:
1. Data is located in one place (one server)
2. All DBMS functionalities are done by that server
3. Enforcing ACID properties of transactions
4. Concurrency control, recovery mechanisms
5. Answering queries

In Distributed databases:
1. Data is stored in multiple places (each is running a DBMS)
2. New notion of distributed transactions
3. DBMS functionalities are now distributed over many machines

Why Distributed and Parallel Databases?

- Data is too large
- Applications are by nature distributed
- Bank with many branches
- Chain of retail stores with many locations
- Library with many branches
- Get benefit of distributed and parallel processing
- Faster response time for queries

Distributed processing usually imply parallel processing (not vice versa) can have parallel processing on a single machine.

Assumptions about architectures
Parallel Databases
• Machines are physically close to each other, e.g., same server room
• Machines connects with dedicated high-speed LANs and switches
• Communication cost is assumed to be small
• Can shared-memory, shared-disk, or shared-nothing architecture

Distributed Databases
• Machines can far from each other, e.g., in different continent
• Can be connected using public-purpose network, e.g., Internet
• Communication cost and problems cannot be ignored
• usually shared-nothing architecture

Relational Database Design & Normalization:

Database practitioners talk about normalization a lot, but it's often poorly understood. Some people see it as an academic detail that is impractical in the real world. Many practitioners are convinced that normalization is always too costly in terms of performance. In fact there are important practical benefits to normalization. Even though there may be legitimate reasons to eschew normalization in some circumstances, it is always best to make this decision from a standpoint of understanding the costs and benefits than dismissing it out of ignorance.

Normalization provides a set of rules and patterns that can be applied to any database to avoid common logical inconsistencies. Normalizing a database design will typically improve:

- **Consistency**, since errors that could be made with the database would be structurally impossible
- **Extensibility**, since changes to the database structure will only affect parts of the database they are logically dependent on
- **Efficiency**, since redundant information will not need to be stored

The database community has identified several distinct levels of normalization, each more stringent than the last. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). It's quite common in practice to speak of one database design as being more or less normalized than another, as defined by these levels.

In practical applications, you'll often see 1NF, 2NF, and 3NF along with the occasional 4NF. Fifth normal form is very rarely seen but is briefly discussed in this article. Fifth normal form means that the SQL selecting newbies won't produce misleading results by doing a join with artificial, spurious rows occurring due to a many-to-many dependency, or something equal to or more than a 3-way relationship, which are rare in comparison to the most of the many-to-one that will be detected by the earlier forms.

A database is a set of facts about the world. Database administrators would like their database to contain only true facts about the world, and no false facts. Normalization allows the RDBMS to prevent some classes of false fact from occurring in the database, and therefore helps consistency. Obviously much of the information in the database can't be verified by the computer; the only types of error that can be prevented are errors that are inconsistent and entail some logical contradiction. Generally speaking, with each additional level of normalization one or more classes of logical error are made impossible by virtue of the structure of the tables.

An *anomaly* occurs in a database if the database starts in a consistent state and an apparently valid action causes the database to become inconsistent. This can be further specified as an *update*, *insertion* or *deletion anomaly* depending on whether the action that causes the error is a row update, insertion or deletion respectively.

Object Oriented and Object Relational Database

An Object relational model is a combination of a Object oriented database model and a Relational database model. So, it supports objects, classes, inheritance etc. just like Object Oriented models and has support for data types, tabular structures etc. like Relational data model.

One of the major goals of Object relational data model is to close the gap between relational databases and the object oriented practises frequently used in many programming languages such as C++, C#, Java etc.

History of Object Relational Data Model

Both Relational data models and Object oriented data models are very useful. But it was felt that they both were lacking in some characteristics and so work was started to build a model that was a combination of them both. Hence, Object relational data model was created as a result of research that was carried out in the 1990's.

Advantages of Object Relational model

The advantages of the Object Relational model are:

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.

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.

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.

A relational databases relies on the relational model, on the other hand a object database relies on the OOP. In a object oriented database each element resembles a object from the object oriented paradigm.

The difference between relational database and object oriented database is that the relational data base stores data in the form of tables which contains rows and columns. Every column in the table has its specific name and every row of the table has its own primary key.

While in the object oriented database the data is stored in the form of objects. In the object oriented data the data is stored along with its actions that processes or reads the existing data.

An object database stores complex data and relationships between data directly, without mapping to relational rows and columns, and this makes them suitable for applications dealing with very complex data. Objects have a many to many relationship and are accessed by the use of pointers. Pointers are linked to objects to establish relationships. Another benefit of an OODBMS is that it can be programmed with small procedural differences without affecting the entire system.

But there also some important factors influencing the usage of object-oriented and relational databases:

- Popularity. There are a lot of products using relational databases unlike object-oriented ones, which aren't used much at the large commercial products.
- Request language and its standardization. There's a standard for requests language for all the relational databases. OODBs don't have one yet, also there's no unified opinion of how this language should work.
- Mathematical apparatus. For the relational DBs, Edgar Codd has laid the foundation of the mathematical apparatus of relational algebra. This apparatus explains how basic operations on relations in the database should be performed, proves their optimality (or it shows where to optimize). On the other hand, there is no such apparatus for OODB, even though works in this field have been conducted since the 80s. Thus, there are no strict terms in the OODB.
- The problem of storage of data and methods. In relational databases only bare data is stored. In OODB, on the contrary, objects should be stored, and the object is a collection of its properties (object parameters) and methods (object interface). There is also no consensus as to how OODB should store objects and how the developer should develop and design these objects. Here, the problem arises of storing the hierarchy of objects, storing abstract classes, and so on.