B.Sc (Computer Science)
Database Management Systems
UNIT-I

1. Define (a). Data (b). Field (c). Record (d). File.

Data: Data have little meaning unless they have recognized in some logical manner. The smallest piece of data that can be recognized by the computer is a single character.

Field: A character or group of characters that has a specific meaning. A field is used to define and store data.

Record: A logically connected set of one or more fields that describes an object, thing or place.

File: A collection of related records.

2. Define Database and DBMS.

Database: A database is a shared integrated computer structure that stores a collection of:

- End-user data that is raw facts of interest of end user.
- Metadata or data about data through which the end-user data are integrated and managed.

DBMS: A Database Management System (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database.

3. What is the Role of DBMS?

The DBMS serves as the intermediary between the user and database. The database structure itself is stored as a collection of files and the only way to access the data in those files is through the DBMS. It presents the end user or application program with single, integrated view of data in the database. The DBMS receives all application requests and translates them into the complex operations to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs and users. The application program might be written by a programmer using a programming language such as Visual Basic.NET, Java or C++, or it might be created through a DBMS utility program.

4. What are the advantages of DBMS?

A DBMS provides advantages such as:

(a). Improved Data Sharing: The DBMS helps create an environment in which end users have better access to more data and better manages data. Such access makes it possible for end users to respond quickly to change in their environment.

(b). Improved Data Security: The more users access the data, the greater the risks of data security breaches. Corporations invest considerable amounts of time effort and money to ensure that corporate data are used properly. A DBMS provides a framework for better enforcement of data privacy and security policies.

(c). Better Data Integration: Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.

(d). Minimized Data Inconsistency: Data inconsistency exists when different versions of the same data appear in different places. The probability of data inconsistency is greatly reduced in a properly designed database.

(e). Improved Data Access: The DBMS makes it possible to produce quick answers to queries. A query is a specific request issued to the DBMS for data manipulation like read or updates the data.

(f). Improved Decision Making: Better-managed data and improved data access make it possible to generate better quality information on which better decisions are made.

(g). Increased End-User Productivity: The availability of data combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between the success and failure in the global economy.

5. What are the problems with File System Data Management?

The file system method of organizing and managing data was a definite improvement over a manual system and the file system served a useful purpose in data management for over two decades-a very long time in the computer era.

Many problems and limitations became evident in File System:

1. The first and most glaring problem with the file system approach is that even the simplest data retrieval task requires extensive programming. With the older file system, programmers had to specify what must be done and how it was to be done. Modern databases use a non-procedural data manipulation language that allow the user to specify what must be done without specifying how it must be done.

2. Making changes in an existing structure can be difficult in a file system environment. In fact, any change to a file structure, no matter how minor, forces modifications in all of the programs that use the data in that file. Modifications are likely to produce errors and additional time is spent using a debugging process to find these errors.

3. System Administration becomes more difficult if the number of files in the system expands. Even a simple file system with a few files requires the creation and maintenance of several file management programs.

4. Another fault of the file system database is that security features are difficult to program and are therefore omitted in a file system environment. Such features include effective password protection, the ability to lock out parts of files or parts of the system itself and other measures designed to safeguard data confidently. Even when an attempt is made to improve system and data security, the security devices tend to be limited in scope.

6. What is structural and data independence and why is it important?

Changes in the characteristics of data, such as changing a field from integer to decimal require changes in all the programs that access the file. Because all data access programs are subject to change when any of the file's data storage characteristics change, the file system is said to exhibit data dependence. Conversely data independence exits when it is possible to make changes in the data storage characteristics without affecting the application program's ability to access the data.

The practical significance of data dependence is the difference between the logical data format (how the human being views the data) and the physical data format (how the computer must work with the data). Any program that accesses a file system's file must tell the computer not only what to do, but also how to do it. Each program must contain lines that specify the opening of a specific file type, its record specification and its field definitions. Data dependence makes the file system extremely cumbersome from the point of view of a programmer and database manager.

7. Why database design is important?

Database design refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data. A database that meets all user requirements does not just happen; its structure must be designed carefully. Even a good DBMS will perform poorly with a badly designed database.

Proper database design requires the designer to identify precisely the database expected use. Designing a transactional database recognizes accurate and consistent data and operational speed. The design

of a data warehouse database recognizes the use of historical and aggregated data. Designing a database to be in a centralized, single-user environment requires a different approach from that used in the design of a distributed, multi-user environment.

A well-designed database facilities data management and generates accurate and valuable information. A poorly designed database is likely to become a breeding ground for difficult-to-trace errors that may lead to bad decision making and bad decision making can lead to the failure of an organization.

8. What is Data Redundancy and which characteristics of the file system can lead to it?

Data Redundancy exists when the same data are stored unnecessarily at different places.

Uncontrolled data redundancy sets the stage for:
Data inconsistency: It exists when different and conflicting versions of the same data appear in different places.

Data anomalies: Anomaly means "an abnormality". A field value change should be made in only a single place. Data redundancy fosters an abnormal condition by forcing field value changes in many different locations. Anomalies are 3 types, namely, **Update** anomalies, **Insertion** anomalies and **Deletion** anomalies.

9. What are the main components of Database Systems?
(OR)
Explain Database System Environment.

The term database system refers to an organization of components that define and regulate the collection, storage, management and use of the data within a database environment.

The database system is composed of the five major parts:
1. hardware
2. software
3. people
4. procedures
5. data

Hardware: Hardware refers to all of the system's physical devices; for example, computers (microcomputers, workstations, servers and supercomputers), storage devices, printers, network devices(hubs, switches, routers, fiber optics) and other devices(automated teller machines, ID readers).

Software: Three types of software is needed
(a).Operating System Software: It manages all hardware components and makes it possible for other software to run on the computers. Examples of operating system software include Microsoft Windows, Linux, Mac OS and UNIX.
(b). DBMS Software: It manages the database within the database system. Some examples of DBMS software include Microsoft SQL Server, Oracle, MySQL and IBM DB2.
(c). Application programs and utility software: They are used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place. Application programs are most commonly used to access data found within the database to generate reports, tabulations and other information to facilitate decision making. Utilities are the software tools used to help manage the database system's computer components. For example, all major DBMS vendors now provide GUI to help create database structures, control database access and monitor database operations.

People: This component includes all users of the database system. On the basis of primary job functions, five types of users can be identified in a database system.
(a). System Administrators: They oversee the database system's general operations.
(b). Database Administrators: also known as DBA, manage the DBMS and ensure that the database is functioning properly. The DBA role is sufficiently important in Database Administration and Security.

(c). Database Designers: They design the database structure. They are, in effect, the database architects. If the database design is poor, even the best application programmers and the most dedicated DBA cannot produce a useful database environment.

(d). System Analysts and Programmers: They design and implement the application programs. They design and create the data entry screens, reports and procedures through which end users access and manipulate the database's data.

(e). End Users: They are the people who use the application programs to run the organization's daily operations. For example, clerks, supervisors, managers, etc.

Procedures: Procedures are the instructions and rules that govern the design and use of the database system. Procedures play an important role in a company because they enforce the standards by which business is conducted within the organization and with customers. Procedures are also used to ensure that there is an organized way to monitor and audit both the data and the information that is generated through the use of the data.

Data: The word data covers the collection of facts stored in the database. Because data are the raw material from which information is generated, the determination of what data are to be entered into the database and how that data are to organized is a vital part of the database designer's job.

10. What are the functions of DBMS?

A DBMS performs several functions that guarantee the integrity and consistency of the data in the database. Most of these functions are transparent to the users and most can be achieved only through the use of a DBMS.

Data Dictionary Management: The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary. In turn, all programs that access the data in the database work through the DBMS. The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving the programmer from having to code such complex relationships n each program. Additionally, any changes made in a database structure are automatically recorded in the data dictionary. In other words, the DBMS provides data abstraction and it removes structural and data dependency from the system.

Data Storage Management: The DBMS creates and manages the complex structures required for data storage, thus relieving programmer from the difficult task of defining and programming the physical data characteristics. A modern DBMS provides storage not only for the data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture formats, etc. Data storage management is also important for database performance tuning. Performance tuning relates to the activities that make the database perform more efficiently in terms of storage and access speed. Although the user sees the database as a single data storage unit, the DBMS actually stores the database in multiple physical files. Such data files may even be stored on difficult storage media. Therefore, the DBMS doesn't have to wait for one dist request to finish before the next one starts. In other words, the DBMS can fulfill the database requests concurrently.

Data Transformation and Presentation: The DBMS transforms entered data to conform to required data structures. The DBMS formats the physically retrieved data to make it conform to the user's logical expectations. An end user in India would expect to enter data such as June 15, 2010 as 15/06/10 whereas the same date is entered in US as 06/15/10.

Security Management: The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database which data items each user can access and which data operations (read, add, delete or modify) the user can perform. This is especially important in multi-user database system. All database users may be authenticated to the DBMS through a username and password or through biometric authentication such as fingerprint scan. The DBMS uses this information to assign privileges to various database components such as queries and reports.

Multi-user Access Control: To provide data integrity and data inconsistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database.

Backup and Recovery Management: The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery management deals with the recovery of the database after a failure such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity.

Data Integrity Management: The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data inconsistency. The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is especially important in transaction-oriented database systems.

Database Access Languages and API: The DBMS provides data access through a query language. A query language is a non-procedural language – one that lets the user specify what must be done without having to specify how is to be done. SQL is the query language and data access standard supported by majority of DBMS vendors. The DBMS also provides application programming interfaces to procedural languages such as COBOL, C. Java, etc. In addition, the DBMS provides administrative utilities used by the DBA and the database designer to create, implement, monitor and maintain the database.

Database Communication Interface: Current generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Internet Explorer, Firefox.

11. List and describe the different types of Databases.

A DBMS support many different types of Databases. Databases can be classified according to the number of users, the database locations and expected type and extent of use.

Single-user: A single user database supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A is done. A single user database that runs on a personal computer is called a desktop database.

Multi-user: A multi-user database supports multiple users at the same time. When the multi-user database supports a relatively small number of users or a specific department within an organization, it is called as **Workgroup database**. When the database is used by the entire organization and supports many users across the department, the database is known as **Enterprise Database.**

Centralized Database: A database that supports data located at a single site is called a Centralized database.

Distributed Database: A database that supports data distributed across different sites is called a distributed database.

Operational Database: A database that is designed primarily to support a company's day-to-day operations is called as Operational database.

Data Warehouse: A data warehouse primarily focuses on storing data used to generate information required to make tactical or strategic decisions. Data warehouse can store data derived from many sources.

12. What are the disadvantages of database systems?
(OR)
What are the potential costs of implementing a database system?

Database systems do carry some disadvantages:

Increased Costs: Database systems required sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software and personnel required operating and mange a database system can be substantial. Training, licensing and regulation compliance costs are often overlooked when database systems are implemented.

Management Complexity: Database systems interface with many different technologies and have a significant impact on a company's resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.

Maintaining Currency: To maximize the efficiency of the database system, we must keep the system current. Therefore, frequent updates and security measures are properly performed on all the components. Because database technology advances rapidly, personnel training costs tend to be significant.

Vendor Dependence: Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers and those customers might be limited in their choice of database system components.

Frequent upgrade/replacement cycles: DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of the versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

13. Discuss the importance of data modeling?

A good database design is the foundation for good applications. Regardless of the skills of programmers and/or the effectiveness of application generators, *we cannot develop good applications without a good database design*. And good database design uses an appropriate data model as its foundation.

The importance of data modeling cannot be overstated. Data constitute the most basic information units employed by a system. Applications are created to manage data and to help transform data into information. But data are viewed in different ways by different people. For example, contrast the (data) view of a company manager and that of a company clerk. Although both manager and clerk work for the same company, the manager is much more likely to have an enterprise-wide view of company data than the clerk.

Applications programmers have yet another view of data, being more concerned with data location, formatting, and specific reporting requirements. Basically, applications programmers translate company policies and procedures from a variety of sources into appropriate interfaces, reports, and query screens.

If we have a good blueprint, it does not matter that an applications programmer's view of the data is different from that of the manager and/or the end user. Conversely, if we do not have a blueprint, problems are likely to ensue. For instance, an inventory management program or an order entry system may not fit into the overall set of operational requirements, thereby costing the company thousands (or even millions!) of dollars.

Keep in mind that a house blueprint is an abstraction; you cannot live in the blueprint. Similarly, the data model is an abstraction; you cannot draw the required data out of the data model. Just as you are not likely to build a good house without a blueprint, you are equally unlikely to create a good database without first selecting an appropriate data model.

14. What are the Data Model basic building blocks?

The basic building blocks of all data models are entities, attributes, and relationships. An **entity** is anything, such as a person, place, thing, or event, about which data are to be collected and stored. Entities may be physical objects such as customers or products. But entities may also be abstractions such as flight routes or musical concerts.

An **attribute** is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone, customer address, and customer credit limit. The attributes are the equivalent of fields in file systems.

A **relationship** describes an association among (two or more) entities. For example, there is a relationship between customers and agents that may be described as "*an agent can serve many customers and each customer may be served by one agent*. Data models use three types of relationships: one-to-many, many-to-many, and one-to-one. Database designers usually use the shorthand notations 1:M, M:N, and 1:1 for them,

15. What is a business rule and what is its purpose in data modeling?

A business rule is a brief, precise and unambiguous description of a policy, procedure or principle within a specific organization.

Business rules, derived from a detailed description of an organization's operations, help to create and enforce actions within that organization's environment. Business rules must be rendered in writing and updated to reflect any change in the organization's operational environment.

To be effective, business rules must be easy to understand and widely disseminated to ensure that every person in the organization shares a common interpretation of such rules. Business rules describe, in simple language, the main and distinguishing characteristics of the data *as viewed by the company*. Examples of business rules are:

1. A customer may make many payments on account.
2. Each payment on account is credited to only one customer.
3. A machine operator may not work more than 10 hours in any 24-hour period.
4. A business trip destination must be at least 200 miles away for an airline ticket to be purchased.
5. A training session cannot be scheduled for fewer than 10 employees or for more than 30 employees.
6. A company aircraft must have an airframe inspection every 100 flight hours.
7. A lab cannot be scheduled for demonstration purposes more than one hour per day.
8. A customer may generate many invoices.
9. Each invoice is generated by only one customer.

Business rules are essential to database design for several reasons:

- They help standardize the company's view of data.
- They constitute a communications tool between users and designers.
- They allow the designer to understand the nature, role, and scope of the data.
- They allow the designer to understand business processes.
- They allow the designer to develop appropriate relationship participation rules and constraints.

16. How to translate business rules into data model components?

Business rules set the stage for the proper identification of entities, attributes, relationships and constraints. In the real world, names are used to identify objects. If the business environment wants to keep track of the objects, there will be specific business rules for them. As a general rule, a noun in a business rule

will translate into an entity in the model and a verb associating nouns will translate into a relationship among the entities.

To properly identify the type of relationship, we should consider that relationships are bidirectional; For example, the business rule "a customer may generate many invoices" is complemented by the business rule "an invoice is generated by only one customer".

As a general rule to properly identify the relationship type, we should ask two questions:

- How many instances of B are related to one instance of A?
- How many instances of A are related to one instance of B?

17. Abbreviate (a) CODASYL (b) SPARC (c) ANSI (d) DBTG

CODASYL: Conference on Data Systems Language
SPARC: Standards Planning And Requirements Committee
ANSI: American National Standards Institute
DBTG: Database Task Group

18. Explain the Three-Level Database Abstraction.

It is a standard database structure consisting of Three-Levels.
(i). **Conceptual Level:** It is the level at which conceptual database design is done. Conceptual Database Design involves analysis of user's information needs and definition of data items needed to meet them. The result of conceptual design is the conceptual schema, a single and logical description of all data elements and their relationships.
(ii). **External Level:** It consists of the user views of the database. Each definable user group will have its own view of the database. Each of these views gives a user-oriented description of the data elements and relationships of which the view is composed. It can be directly derived from conceptual schema.
(iii). **Internal Level:** The internal level provides the physical view of the database – the disk drives, physical addresses, indexes, pointers and so on. This level is the responsibility of physical database designers, who decide which physical devices will contain data, what access methods will be used to retrieve and update data and what measures will be taken to maintain or improve database performance.

19. What are the characteristics of a Table?

1. A table is perceived as a two-dimensional structure composed of rows and columns.
2. Each table row (tuple) represents a single entity occurrence within the entity set.
3. Each table column represents an attribute and each column has a distinct name.
4. Each row/column intersection represents a single data value.
5. All values in a column must conform to the same data format.
6. Each column has a specific range of values known as the attribute domain.
7. The order of the rows and columns is immaterial to the DBMS.
8. Each table must have an attribute or a combination of attributes that uniquely identifies each row.

20. What is meant by Functional Dependency?

The attribute B is functionally dependent on the attribute A if each value in column A determines **one and only one value** in the column B.
E.g. Sno value in the Student table determines all of the student's attribute value.

This can be represented as follows:

Sno → Sname, Class, DOB

21. What is a Key?

In the relational model, keys are important because they are used to ensure that each row in a table is uniquely identified. They are also used to establish relationships among tables and to ensure the integrity of the data. Therefore, a proper understanding of the concepts and use of keys in the relational model is very important. A key consists of one or more attributes that determines other attributes.

Superkey: An attribute (or combination of attributes) that uniquely identifies each row in a table.

Candidate Key: A minimal (irreducible) superkey. A superkey doesn't contain a subset of attributes that is itself is a superkey.

Primary Key: A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.

Secondary Key: An attribute (or combination of attributes) used strictly for data retrieval purpose.

Foreign Key: An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

22. What does it mean to say a database displays both Entity Integrity and Referential Integrity?

Entity Integrity: All primary key entries are unique and no part of the primary key may be null. Each row will have a unique identity and foreign key values can properly reference primary key.
E.g. No invoice can have a duplicate number, nor can it be null. In short, all invoices are uniquely identified by their invoice number.

Referential Integrity: A foreign key may have either a null entry as long as it is not a part of its tables primary key or an entry that matches the primary key values in a table to which it is related. (Every non-null foreign key value must reference an existing primary key value). It is possible for an attribute NOT to have a corresponding value, but it will be impossible to have an invalid entry. The enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
E.g. A customer might not yet have an assigned sales representative number, but it will be impossible to invalid sales representative number.

23. Explain about Relational Set Operators.

Relational algebra defines theoretical way of manipulating capabilities of the relational model using the eight relational operators:

1. UNION
2. INTERSECT
3. DIFFERENCE
4. PRODUCT
5. SELECT
6. PROJECT
7. DIVIDE
8. JOIN

UNION: It combines all rows from two tables excluding duplicate rows. The tables must have the same attribute characteristics (the columns and domains must be identical) to be used in the UNION. When two or more tables share the same number of columns, when the columns have the same names and when they share the same (or compatible) domain, they are said to be union-compatible.

INTERSECT: It yields only the rows that appear in both tables. As was true in the case of UNION, the tables must be union-compatible to yield valid results. For example, we cannot use INTERSECT if one of the attributes is numeric and one is character-based.

DIFFERENCE: It yields all rows in one table that are found in the other table. It subtracts one table from the other. As was true in the case of UNION, the tables must be union-compatible to yield the results.

PRODUCT: It yields possible pairs of rows from two tables also known as the Cartesian Product. Therefore if one table has six rows and the other has three rows, the PRODUCT yields a list composed of 6 X 3 = 18 rows.

SELECT: It is also known as RESTRICT, yields values for all rows found in a table that satisfy a condition. SELECT can be used to list all the rows values or it can yield only those row values that match a specified criterion. In other words, SELECT yields a horizontal subset of the table.

PROJECT: It yields all values for a selected attributes. In other words, PROJECT yields a vertical subset of a table.

DIVIDE: The DIVIDE operation uses one single-column table (i.e. column a) as the divisor and one 2-column table (i.e. columns a and b) as the dividend. The tables must have a common column (i.e. column a). The output of the DIVIDE operation is a single column with the values of column 'a' from the dividend table rows where the value of the common column (i.e. column a) in both tables match.

JOIN: It allows information to be combined from two or more tables. JOIN is the real power behind the relational database, allowing the use of independent tables linked by common attributes.

(a). **Natural Join**: A natural join links tables by selecting only rows with common values in their common attributes.

A natural join is the result of three-stage process

1. Product of the two tables.
2. Select is performed on the result of Product.
3. Project is performed on the result of select.

The final outcome of a natural join yields a table that does not include unmatched pairs and provides only the copies of the matches.

(b). **EquiJoin**: It links tables on the basis of an equality condition that compares specified columns of each table. The outcome of the equijoin does not eliminate duplicate columns and the condition or criterion used to join the tables must be explicitly defined. The equijoin takes its name from the equality comparison operator (=) used in the condition. If any other comparison operator is used, the join is called as a **theta join**.

(c). **Outer Join**: In an outer join, the matched pairs would be retained and any unmatched values in the other table would be left null. Outer joins are especially useful when we are trying to determine what values in related tables causes referential integrity problems. Such problems are created when foreign key values don't match the primary key values in the related tables.

24. What are the Codd's Relational Database Rules?

In 1985, Dr.E.F.Codd published a list of 12 rules to define a relational database system. The reason Dr.Codd published the list was his concern that many vendors were marketing products as relational even though those products did not meet minimum relational standards.

1. **Information:** All information in a relational database must be logically represented column values in rows within tables.
2. **Guaranteed Access:** Every value in a table is guaranteed to be accessible through a combination of table name, primary key and column name.
3. **Systematic Treatment of Nulls:** Nulls must be represented and treated in a systematic way, independent of data type.
4. **Dynamic On-Line Catalog Based on the Relational Model:** The metadata must be stored and managed as ordinary data, that is, in the table.

5. **Comprehensive Data Sublanguage:** The relational database may support many languages. However, it must support one well defined, declarative language with support for data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization and transaction management (begin, commit and rollback).
6. **View Updating:** Any view that is theoretically updatable must be updatable through the system.
7. **High-Level Insert, Update and Delete:** The database must support set-level inserts, updates and deletes.
8. **Physical Data Independence:** Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9. **Logical Data Independence:** Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of column or inserting columns).
10. **Integrity Independence:** All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11. **Distribution Independence:** The end users and application programs are unaware and unaffected by the data location (distributed or local database).
12. **Non Subversion:** It the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.

Rule Zero: All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

25. What is meant by Index?

An index is orderly arrangement used to logically access rows in a table. Index in the relational database environment is composed of an index key and a set of pointers. The index key is in effect, the index's reference point. More formally, an index is an ordered arrangement of keys and pointers. Each key points to the location of the data identified by the key.

DBMS uses indexes for many different purposes. Indexes can also used to retrieve data ordered by a specific attribute or attributes. For example, creating an index on a customer's name will allow retrieving the customer data alphabetically by the customer's last name. Indexes play an important role in DBMS for the implementation of primary key. To define a table's primary key, the DBMS automatically creates a unique index on the primary key column.

A table can have many indexes, but each index is associated with only one table. The index key can have multiple attributes (composite index).

26. What are homonyms and Synonyms?

Homonym: The word homonym indicates the use of same attribute name to label different attributes. For example, C_NAME is used to label customer name in CUSTOMER table and also in CONSULTANT table.

Synonym: A synonym is the opposite of homonym and indicates the use of different name to describe the same attribute. For example *car* and *auto* refer to the same object.

Both Homonyms and Synonyms should be avoided in database.