**DATA:**

* Data is Raw, unorganized facts that need to be processed.
* Data comprises facts, figures, observations, numbers, characters, symbols, images, etc.
* Data is a known facts about any entity.
* Set of Data that is processed in meaningful way according to the given requirement is called Information.
* *For Example:- 100, Amazon, R No., Name*
* Collection of Interrelated Data called Record.

Ex. R No Name Subject Marks

01 John POD 18

**DATABASE:**

Database is a collection of interrelated data i.e. *Records*.

* Database Systems are designed to manage large amount of Data.
* The Database system must ensure the safety of the Data.
* Database System applications:

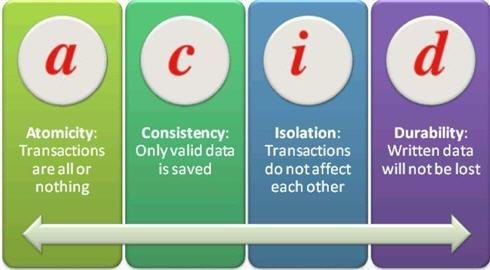
1. ***Banking:-*** *For customer info, accounts, loans, and banking transactions.*
2. ***Airlines/Trains :-*** *For Reservations and schedule Information.*
3. ***Universities:-*** *For Student information, Course registrations, Results.*
4. ***Telecommunication:-*** *For keeping records of calls made. For Billing*

**Database Purpose**

* Provides secure and survivable medium for storage & retrieval of Data.
* Data shared among several users & is persistent.
* Provides mechanism to create, access and manipulate data without compromising security and integrity of Data.
* Redundancy can be reduced.
* Inconsistency can be avoided.

**CHARACTERISTICS OF DATABASE:**

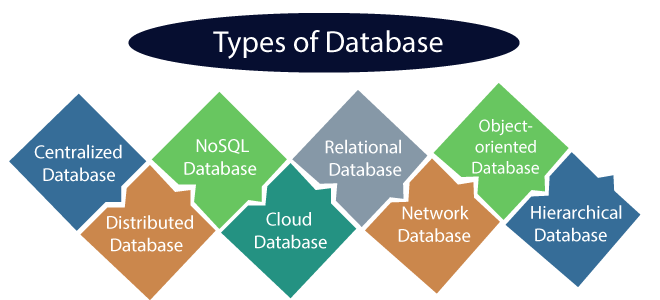
* **Real-world entity: -** We should be able to store all kinds of data that exist in this real world. Since we need to work with all kinds of data and requirements, the database should be strong enough to store all kinds of data that are present around us.
* **Relation-based tables: -** We should be able to relate the entities/tables in the database by means of relation. Ex. An employee works for a department. This implies that an Employee is related to a particular department.
* **Isolation of data and application: -** Data and applications should be isolated. Because the database is a system that gives the platform to store the data, and the data is the one that allows the database to work. Hence there should be a clear differentiation between them.
* **Less redundancy**: - There should not be any duplication of data in the database. Data should be stored in such a way that it should not be repeated in multiple tables. If repeated, it would be an unnecessary waste of DB space, and maintaining such data becomes chaos.
* **Consistency: -** Consistency is a state where data cannot be written that would violate the database’s own rules for valid data. If a certain transaction occurs that attempts to introduce inconsistent data, the entire transaction is rolled back and an error returned to the user.
* **Query Language**: - DBMS has a strong query language. Once the database is designed, this helps the user to retrieve and manipulate the data. If a particular user wants to see any specific data, he can apply as many filtering conditions that he wants and pull the data that he needs.
* **Multiuser and Concurrent Access: -** Multiple users should be able to access the same database, without affecting the other user. i.e.; if teachers want to update a student’s marks in the Results table at the same time, then they should be allowed to update the marks for their subjects, without modifying other subject marks.
* **Multiple views**: - It supports multiple view to the user, depending on his role. In a school database, Students will able to see only their reports and their access would be read-only. At the same time, teachers will have access to all the students with modification rights. But the database is the same. Hence a single database provides different views to different users.
* **Security**: - The database should also provide security, i.e.; when there are multiple users are accessing the database, each user will have their own levels of rights to see the database.  Some of them will be allowed to see the whole database, and some will have only partial rights. For example, an instructor who is teaching Physics will have access to see and update marks of his subject. He will not have access to other subjects. But the HOD will have full access to all the subjects.
* **ACID Properties: -** The database should also support the ACID property. i.e.; while performing any transactions like insert, update and delete, the database makes sure that the real purpose of the data is not lost. For example, if a student’s address is updated, then it should make sure that there is no duplicate data is created nor there is any data mismatch for that student.



**FILE SYSTEM V/S DATABASE APPROACH:**

|  |  |
| --- | --- |
| **File based system** | **Database system** |
| 1. The data and program are inter- dependent. | 1. The data and program are independent of each other. |
| 2. File-based system caused data redundancy. The data may be duplicated in different files | 2. Database system control data redundancy. The data appeared only once in the system. |
| 3. File –based system caused data inconsistency. The data in different files may be different that cause data inconsistency. | 3. In database system data always consistent. Because data appeared only once. |
| 4. The data cannot be shared because data is distributed in different files. | 4. In database data is easily shared because data is stored at one place. |
| 5. In file-based system data is widely spread. Due to this reason file-based system provides poor security. | 5. It provides many methods to maintain data security in the database. |
| 6. File based system does not provide consistency constrains. | 6. Database system provides a different consistency constrains to maintain data integrity in the system. |
| 7. File based system is less complex system. | 7. Database system is very complex system. |
| 8. To generate different report to take a crucial decision is very difficult in file-based system. | 8. The report can be generated very easily in required format in database system. |
| 9. File based system takes much space in the system, and memory is wasted in this approach. | 9. Database approach store data more efficiently it takes less space in the system and memory is not wasted. |
| 10. File based system does not provide concurrency facility. | 10. Database system provides concurrency facility. |
| 11. File based system does not provide data atomicity functionality. | 11. Database system provides data atomicity functionality. |

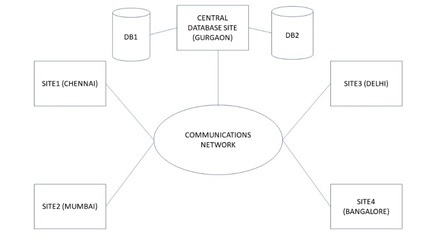
**TYPES OF DATABASE:**



1. **Centralized Database**

The information(data) is stored at a centralized location and the users from different locations can access this data. This type of database contains application procedures that help the users to access the data even from a remote location.

Various kinds of authentication procedures are applied for the verification and validation of end users, likewise, a registration number is provided by the application procedures which keeps a track and record of data usage. The local area office handles this thing

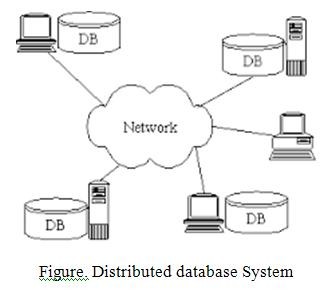


**2.DISTRIBUTED DATABASE**

Just opposite of the centralized database concept, the distributed database has contributions from the common database as well as the information captured by local computers also. The data is not at one place and is distributed at various sites of an organization. These sites are connected to each other with the help of communication links which helps them to access the distributed data easily.

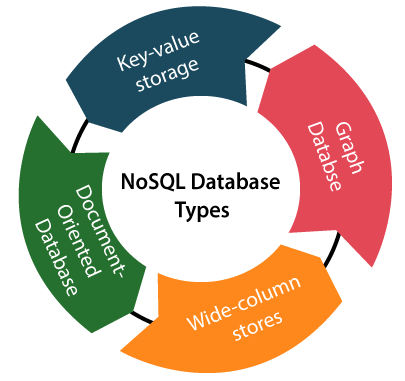
You can imagine a distributed database as a one in which various portions of a database are stored in multiple different locations(physical) along with the application procedures which are replicated and distributed among various points in a network.

There are two kinds of distributed database, viz. homogenous and heterogeneous. The databases which have same underlying hardware and run over the same operating systems and application procedures are known as homogeneous DDB, for eg. All physical locations in a DDB. Whereas, the operating systems, underlying hardware as well as application procedures can be different at various sites of a DDB which is known as heterogeneous DDB.



**3. NOSQL DATABASE**

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways. It came into existence when the demand for building modern applications increased. Thus, NoSQL presented a wide variety of database technologies in response to the demands. We can further divide a NoSQL database into the following four types:



1. **CLOUD DATABASE**

A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database. There are numerous cloud platforms, but the best options are:

Amazon Web Services(AWS) 

Microsoft Azure

Kamatera

PhonixNAP

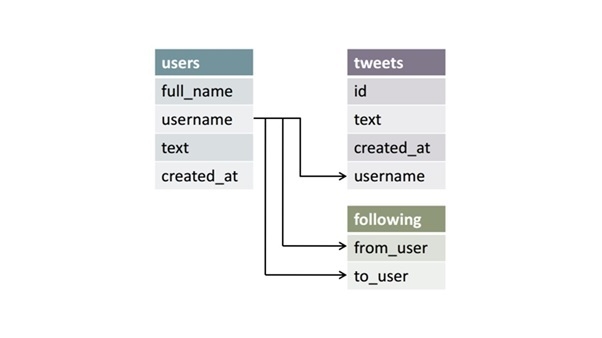
ScienceSoft

Google Cloud SQL, etc.

**5. RELATIONAL DATABASE**

These databases are categorized by a set of tables where data gets fit into a pre-defined category. The table consists of rows and columns where the column has an entry for data for a specific category and rows contains instance for that data defined according to the category. The Structured Query Language (SQL) is the standard user and application program interface for a relational database.

There are various simple operations that can be applied over the table which makes these databases easier to extend, join two databases with a common relation and modify all existing applications.



1. **NETWORK DATABASE**

It is the database that typically follows the network data model.

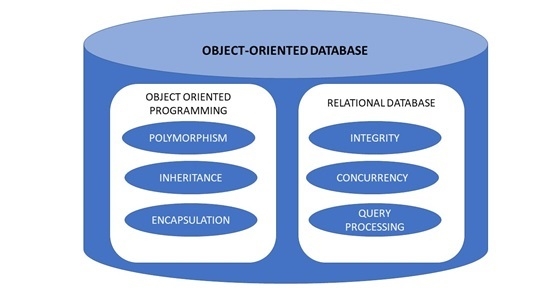
Here, the representation of data is in the form of nodes connected via links between them.

Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

1. **OBJECT ORIENTED DATABASE**

An object-oriented database is a collection of object-oriented programming and relational database. There are various items which are created using object-oriented programming languages like C++, Java which can be stored in relational databases, but object-oriented databases are well-suited for those items.

An object-oriented database is organized around objects rather than actions, and data rather than logic. For example, a multimedia record in a relational database can be a definable data object, as opposed to an alphanumeric value.



1. **HIERARCHICAL DATABASE**

It is the type of database that stores data in the form of parent-children relationship nodes. Here, it organizes data in a tree-like structure.

Data get stored in the form of records that are connected via links. Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.



**ADVANTAGES OF DATABASE**

* Reduced data redundancy
* Reduced updating errors and increased consistency
* Greater data integrity and independence from applications programs
* Improved data access to users through use of host and query languages
* Improved data security
* Reduced data entry, storage, and retrieval costs
* Facilitated development of new applications program

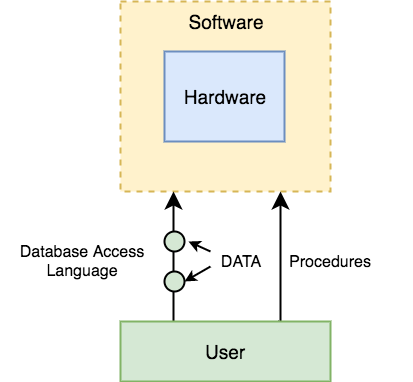
**DISADVANTAGES OF DATABASE**

* Database systems are complex, difficult, and time-consuming to design
* Substantial hardware and software start-up costs
* Damage to database affects virtually all applications programs
* Extensive conversion costs in moving form a file-based system to a database system
* Initial training required for all programmers and users

**COMPONENTS OF DATABASE**

The database management system can be divided into five major components, they are:

* **Hardware**
* **Software**
* **Data**
* **Procedures**
* **Database Access Language**
* **Users**



**HARDWARE**

* When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.
* When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

**SOFTWARE**

* The main component of a Database management system is the software. It is the set of programs which is used to manage the database and to control the overall computerized database.
* The DBMS software provides an easy-to-use interface to store, retrieve, and update data in the database.
* This software component is capable of understanding the Database Access Language and converts it into actual database commands to execute or run them on the database.

**DATA**

* It is the most important component of the database management system.
* The main task of DBMS is to process the data. Here, databases are defined, constructed, and then data is stored, retrieved, and updated to and from the databases.
* The database contains both the metadata (description about data or data about data) and the actual (or operational) data.

**PROCEDURES**

* Procedures refer to general rules and instructions that help to design the database and to use a database management system.
* Procedures are used to setup and install a new database management system (DBMS), to login and logout of DBMS software, to manage DBMS or application programs, to take backup of the database, and to change the structure of the database, etc.

**DATABASE ACCESS LANGUAGE**

* Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database.
* A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS.
* User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.

**USERS**

The users are the people who control and manage the databases and perform different types of operations on the databases in the database management system.

**There are three types of user who play different roles in DBMS:**

**1. Application Programmers:** The users who write the application programs in programming languages (such as Java, C++, or Visual Basic) to interact with databases are called Application Programmer.

**2. Database Administrators (DBA):** A person who manages the overall DBMS is called a database administrator or simply DBA.

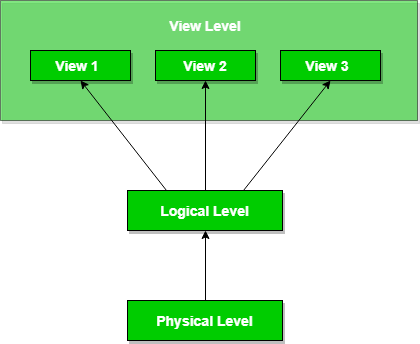
**3. End-Users:** The end-users are those who interact with the database management system to perform different operations by using the different database commands such as insert, update, retrieve, and delete on the data, etc.

**Data Abstraction**

Database systems comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

There are mainly **3**levels of data abstraction:

1. Physical
2. Logical
3. View



**Physical**: This is the lowest level of data abstraction. It tells us how the data is actually stored in memory. The access methods like sequential or random access and file organization methods like B+ trees, hashing used for the same. Usability, size of memory, and the number of times the records are factors which we need to know while designing the database.  
Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes is kept hidden from the user.

**Logical**: This level comprises of the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown.  
We can store the various attributes of an employee and relationships, e.g. with the manager can also be stored.

**View**: This is the highest level of abstraction. Only a part of the actual database is viewed by the users. This level exists to ease the accessibility of the database by an individual user. Users view data in the form of rows and columns. Tables and relations are used to store data. Multiple views of the same database may exist. Users can just view the data and interact with the database, storage and implementation details are hidden from them.

**DATABASE LANGUAGES**

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.

**Types of Database Language**

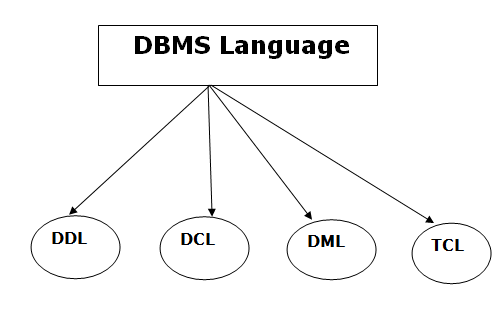
1. **Data Definition Language (DDL) :-**

* It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

**Here are some tasks that come under DDL:**

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.



**2. Data Manipulation Language**

* **DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

**Here are some tasks that come under DML:**

**Select:** It is used to retrieve data from a database.

**Insert:** It is used to insert data into a table.

**Update:** It is used to update existing data within a table.

**Delete:** It is used to delete all records from a table.

**Merge:** It performs UPSERT operation, i.e., insert or update operations.

**Call:** It is used to call a structured query language or a Java subprogram.

**Explain Plan:** It has the parameter of explaining data.

**Lock Table:** It controls concurrency.

**3. Data Control Language**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

**Here are some tasks that come under DCL:**

**Grant:** It is used to give user access privileges to a database.

**Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

**4. Transaction Control Language**

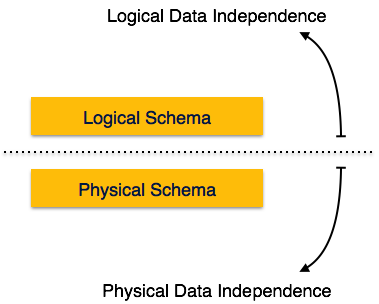
* TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

**DATA INDEPENDENCE**

* A database system normally contains a lot of data in addition to users’ data.
* For example, it stores data about data, known as metadata, to locate and retrieve data easily.
* It is rather difficult to modify or update a set of metadata once it is stored in the database.
* But as a DBMS expands, it needs to change over time to satisfy the requirements of the users.
* If the entire data is dependent, it would become a tedious and highly complex job.



**Logical Data Independence**

* Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.
* Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

**Physical Data Independence**

* All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
* For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

**DATA INTEGRITY**

* The term *data integrity* refers to the accuracy and consistency of data.
* When creating databases, attention needs to be given to data integrity and how to maintain it. A good database will enforce data integrity whenever possible.
* For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

Maintaining data integrity means making sure the data remains intact and unchanged throughout its entire life cycle. This includes the capture of the data, storage, updates, transfers, backups, etc. Every time data is processed there’s a risk that it could get corrupted (whether accidentally or maliciously

**4 Types of Data Integrity**

**1. Entity Integrity**

*Entity integrity* defines each row to be unique within its table. No two rows can be the same.

To achieve this, a primary key can be defined. The primary key field contains a unique identifier – no two rows can contain the same unique identifier.

**2. Referential Integrity**

*Referential integrity* is concerned with relationships. When two or more tables have a relationship, we have to ensure that the foreign key value matches the primary key value at all times. We don’t want to have a situation where a foreign key value has no matching primary key value in the primary table. This would result in an orphaned record.

**3. Domain Integrity**

*Domain integrity* concerns the validity of entries for a given column. Selecting the appropriate data type for a column is the first step in maintaining domain integrity. Other steps could include, setting up appropriate constraints and rules to define the data format and/or restricting the range of possible values.

**THREE LEVEL ARCHITECTURE FOR DB**

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

**Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

**Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

**User (Presentation) Tier −** End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.



**12 CODD’S RULE:**

Here is brief note on E.F Codd’s Twelve rules:

**Rule 0 − Foundation rule**

Any relational database management system that is propounded to be RDBMS or advocated to be a RDBMS should be able to manage the stored data in its entirety through its relational capabilities.

**Rule 1 − Rule of Information**

Relational Databases should store the data in the form of relations. Tables are relations in Relational Database Management Systems. Be it any user defined data or meta-data, it is important to store the value as an entity in the table cells.

**Rule 2 − Rule of Guaranteed Access**

The use of pointers to access data logically is strictly forbidden. Every data entity which is atomic in nature should be accessed logically by using a right combination of the name of table, primary key represented by a specific row value and column name represented by attribute value.

**Rule 3 − Rule of Systematic Null Value Support**

Null values are completely supported in relational databases. They should be uniformly considered as ‘missing information’. Null values are independent of any data type. They should not be mistaken for blanks or zeroes or empty strings. Null values can also be interpreted as ‘inapplicable data’ or ‘unknown information.’

**Rule 4 − Rule of Active and online relational Catalog**

In the Database Management Systems lexicon, ‘metadata’ is the data about the database or the data about the data. The active online catalog that stores the metadata is called ‘Data dictionary’. The so called data dictionary is accessible only by authored users who have the required privileges and the query languages used for accessing the database should be used for accessing the data of data dictionary.

**Rule 5 − Rule of Comprehensive Data Sub-language**

A single robust language should be able to define integrity constraints, views, data manipulations, transactions and authorizations. If the database allows access to the aforementioned ones, it is violating this rule.

**Rule 6 − Rule of Updating Views**

Views should reflect the updates of their respective base tables and vice versa. A view is a logical table which shows restricted data. Views generally make the data readable but not modifiable. Views help in data abstraction.

**Rule 7 − Rule of Set level insertion, update and deletion**

A single operation should be sufficient to retrieve, insert, update and delete the data.

**Rule 8 − Rule of Physical Data Independence**

Batch and end user operations are logically separated from physical storage and respective access methods.

**Rule 9 − Rule of Logical Data Independence**

Batch and end users can change the database schema without having to recreate it or recreate the applications built upon it.

**Rule 10 − Rule of Integrity Independence**

Integrity constraints should be available and stored as metadata in data dictionary and not in the application programs.

**Rule 11 − Rule of Distribution Independence**

The Data Manipulation Language of the relational system should not be concerned about the physical data storage and no alterations should be required if the physical data is centralized or distributed.

**Rule 12 − Rule of Non Subversion**

Any row should obey the security and integrity constraints imposed. No special privileges are applicable.