Introduction

Data: data means a known fact that can be recorded and that have some implicit meaning. E.g., the names, telephone number, address of people etc.

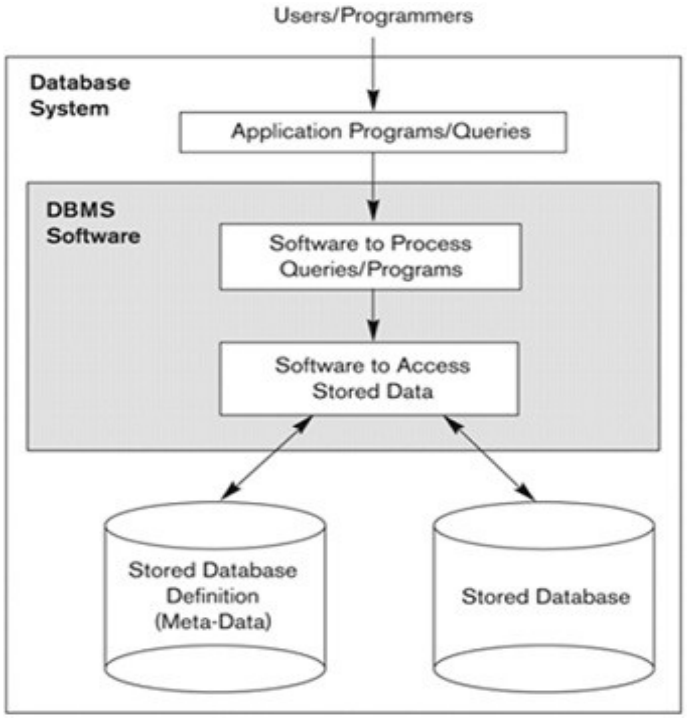
Database: a database is a collection of inter-related data with an implicit meaning.

DBMS: stands for Database Management System and it consist of 2 major components.

- The collection of inter related data, which is called as the database.
- A set of software packages or a set of software tools or programs that can access the data and process the data.

Therefore the primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Database System Environment:



The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications.

- **Defining**: a database involves specifying the data types, structure and constraints for the data to be stored in the database.
- **Constructing**: the database follows the process of storing the data in some storage medium controlled by the DBMS.
- **Manipulating**: a database includes the functions as querying the database to retrieve specific data, updating the database to reflect changes in the mini world and generating reports from the data.
- **Sharing**: a database allows multiple users and programs to access the database concurrently.
- **Protection**: includes both system protection against hardware or software malfunction / crashes and security protection against unauthorized access.
- **Maintain**: the database system by allowing the system to evolve as requirements change over time.

Database approach:

In the database approach a single repository of data is maintained that is defined once and is accessed by various users in many ways.

Characteristics: The following are the main characteristics of the database approach.

- **Self describing nature of a database system:** The database system contains not only the database itself but also a complete definition / description of the database structure and constraints. The descriptions include the structure of each file, its type and storage format of each data item and various constraints imposed on them. This information is called as **metadata** and it is stored in the DBMS catalog.
- **Insulation between program and data, and data abstraction:** in the DBMS approach if we change the structure of a file then it will not require to change all programs that access the file as the structure of the data files are stored in the DBMS catalog separately from the access programs. This property is called as **program–data** independence.

  An operation is specified in two parts

  - The interface / signature of an operation include the operation name and data type of its argument.
  - The implementation / method of the operation are specified separately and can be changed without affecting the interface.
  - User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This property is called as the program operation independence.
- **Data abstraction**: the characteristic that allows program – data independence and program – operation independence is called as data abstraction. Data abstraction means, DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented means it hides storage and implementation details that are not of interest to most database users.
- **Support of multiple views of data**: different users of database require a different perspective or view of the database. A **view** may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. A multiuser database whose users have a variety of distinct application must provide facilities for defining multiple views.
- **Sharing of data and multiuser transaction processing**:
  - The DBMS must include concurrency control software to ensure that several users trying to update the same data must be done in a controlled manner so that the result of the updates is correct.
  - A transaction is an executing program or process that includes one or more database records. Each transaction is supposed to execute a logically correct database access, if executed in its entirety without interference from other transactions. The DBMS must enforce several transaction properties.

Advantages of DBMS:

The following are the advantages of DBMS:

- **Controlling redundancy:**

- The DBMS avoids unnecessary duplication of data and effectively reduces the total amount of data storage required.
- It also eliminates the extra processing necessary to trace the required data in a large mass of data.
- Another advantage of avoiding duplication is the elimination of the inconsistencies that to be present in redundant data files.
- Any redundancies that exist in the DBMS are controlled and the system ensures that these multiple copies are consistent.

- **Shared data**: a database allows the sharing of data under its control by any number of application programs or users.
- **Restricting unauthorized access**: data is vital importance to an organization and may be confidential. Such confidential data must not be accessed by unauthorized persons. For e.g., financial data is often considered confidential and hence only authorized persons are allowed to access such data. The database administrator (DBA) who has the ultimate responsibility for the data in the DBMS can ensure that proper access procedures are followed including proper authentication schemes for access to the DBMS and additional checks before permitting access to sensitive data. Different levels of security can be implemented for the various types of data and operations.
- **Providing backup and recovery**: the backup and recovery subsystem of the DBMS is providing the facilities for recovering from hardware / software failures. For e.g., if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.
- **Integrity**: data integrity means that the data contained in the database is both accurate and consistent. Therefore data values being entered for storage could be checked to ensure that they fall within a specified range and are of the correct format. For e.g., the value of the age of an employee may be in the range of 18 and 60. Another type of constraint specifies the uniqueness of data item values such as "every student must have a unique value for roll number".
- **Providing storage structure for efficient query processing**: DBMS must provide specialized data structures to speed up disk search for the required data. Indexes that are used for disk search are typically based on the tree data structure or hash data structure.

  The query processing and optimization module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structure.

- Representing complex relationships among data: a DBMS must have the capability to represent a variety of complex relationships of data that are interrelated in many ways as well as to retrieve and update related data easily and efficiently.

Database Users:

Database users are those who really use and take the benefits of database. There will be different types of users depending on their need and way of accessing the database.

- **Application Programmers -** They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.
- **Sophisticated Users:** They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.
- **Specialized Users -** These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.
- **Stand-alone Users -** These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces.
- **Naive users:** Any user who does not have any knowledge about database can be in this category. There task is to just use the developed application and get the desired results. For example: Clerical staff in any bank is a naïve user. They don't have any DBMS knowledge but they still use the database and perform their given task.
- **Database Administrators:** The administration and maintenance of database is taken care by database Administrator–DBA. A good performing database is in the hands of DBA. A DBA has many responsibilities:
    - **Installing and upgrading the DBMS Servers: -** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions comes in the market or requirement. If there is any failure in up gradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.
    - **Design and implementation: -** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc for the database.
    - **Performance tuning:** - Since database is huge and it will have lots of tables, data, constraints and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in fraction of seconds.
    - **Migrate database servers: -** Sometimes, users using oracle would like to shift to SQL server. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.
    - **Backup and Recovery: -**Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA.

Data/objects should be backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.

- **Security: -** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.
- **Documentation: -** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.

Database system concepts and architecture:

- **Introduction:** Earlier architectures used mainframe computers to provide the main function for all functions of the system, including user application programs as well as all the DBMS functionality. The reason was that most users accessed such systems via computer terminals that did not have processing power and only providing display capabilities. Now a day's a large number of centralized mainframe computers are being replaced by hundreds of distributed workstations and personal computers connected via communication networks to various types of server machines.

Data models:

- A data model is a collection of concepts that can be used to describe the structure of a database where by structure of database, we mean the data types relationships and constraints that should hold for the data.
- Most data models also include a set of basic operation or basic data model operation such as insert, delete, modify or retrieve any kind of objects.

Categories of data models:

- The data model is organized according to the types of concepts they use to describe the database structure.
  - **Logical / conceptual / high level data model:** it provides the concepts that are close to the way many users perceive data. It describes what data are stored in the database and what relationship exists among the data. It uses the concept such as entities, attributes and relationships.
    Entity: it represents a real world object / concept such as an employee, student or project….
    Attribute: it represents some properties that describe the entity such as employee's name, mobile etc…
    Relationship: a relationship is the association among two or more entities.
  - **Physical / low level data model:** It provides the concept that describes the details of how data is stored in the computer.
  - **Representational / Implementation data model:** between the above two data model, this level provides concepts that may be understood by end users but are not too far removed from the way data is organized within the computer.

Data abstraction: a DBMS provides users with a conceptual representation of data that does not include many of the details how data is stored or how the operations are implemented. It provides user with an abstract view of the data that is the system hides certain details of haw data are stored and maintained that are not needed by most database users.

Database schema: the description of the database or the overall design of the database is called as the database schema, which is specified during database design and is not expected to change frequently.

Database state / snapshot / Instance: the data in the database at a particular moment in time is called as the database state.

Every time we insert or delete a record or change the value of a data item in a record, we change one state of the database into another state.

Metadata: it is the data about data means the description of schema constructs and constraints. Metadata is the information such as the structure of each file, the type and storage format of each data item and various constraints on the data.

Three–schema architecture: The three schema architecture can be defined at the following three levels:

- **Internal level:** the internal level has an internal schema, which is using the physical data model and describes the complete details of physical data storage and access paths of the database.
- **Conceptual level:** the conceptual level has a conceptual schema which hides the details of physical storage structure and concentrates on describing entities, data types, relationships, user operations and constraints.
- **External / view level:** it includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from the user group.

  The DBMS must transform a request specified on an external schema into a request against the conceptual schema and then into a request on the internal schema for processing over the stored database and it must be reformatted to match the

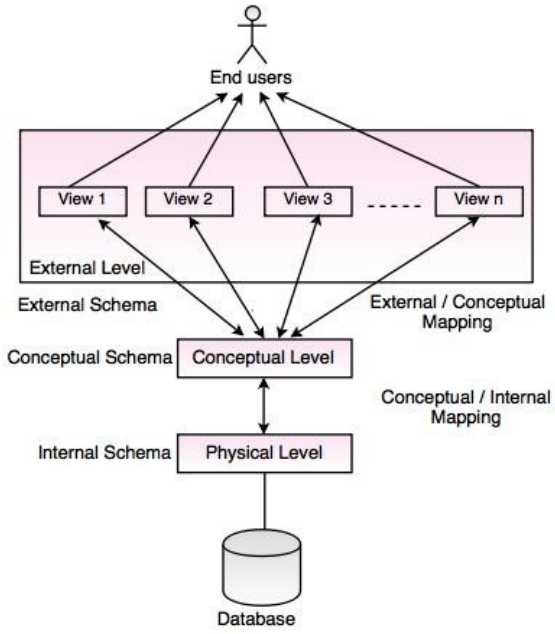
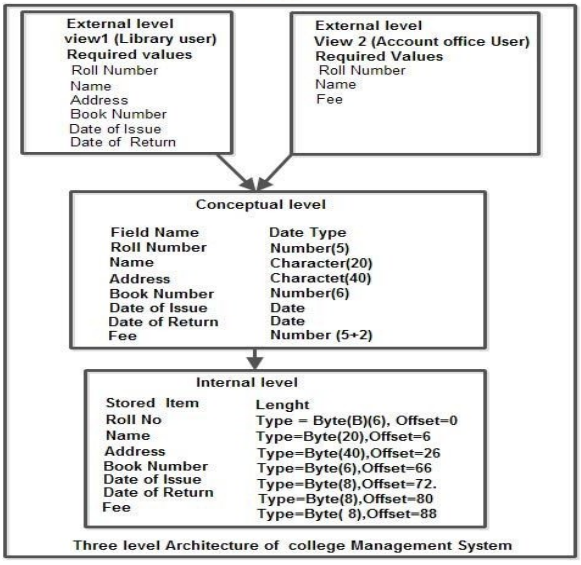


Fig. Three Level Architechture of DBMS



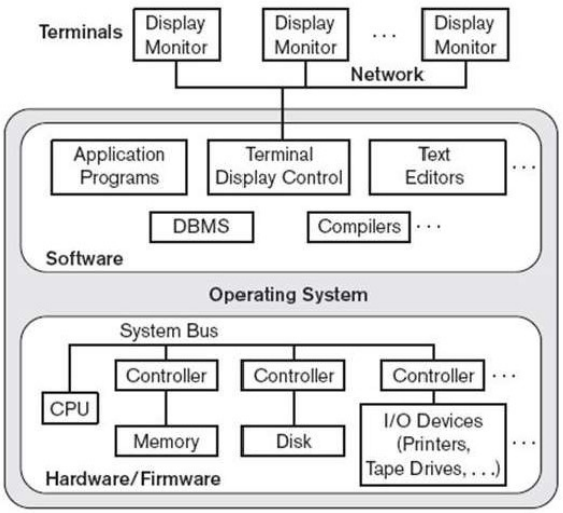
user's external view.

The process of transforming requests and result between levels are called mappings.

Data independence: It is the concept which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

- **Logical data independence:** It is the capacity to change the conceptual schema without having to change the external schema or application programs. We may change the conceptual schema to expand the database by adding a data item, to change constraints or to reduce the database and that will not affect the external schema. Only the view definition and the mappings need to be changed.
- **Physical data independence:** It is the capacity to change the internal schema without having to change the conceptual schema. For e.g., the storage structure or devices used for storing the data could be changed without necessitating a change in the conceptual view / external view.

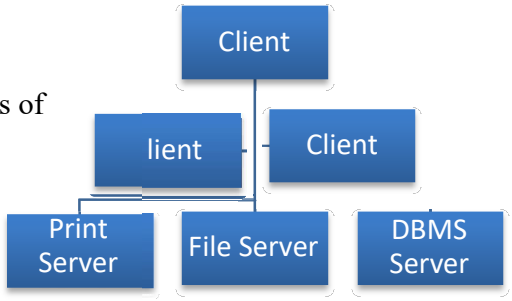
Centralized DBMS architecture:



Earlier architectures used mainframe computers to provide the main processing for all functions of the system, including user application programs and user interface programs as well as the DBMS functionalities.

All the processing was performed remotely on the computer system and only display information and controls were sent from the computer to the display terminals through network.

Basic Client / Server Architecture: The concept of client / Server architecture assumes a framework that consists of many PCs and workstations as well as a smaller number of mainframe computers connected via local area network and other types of networks. A client in this framework is typically a user machine that Provides user interface capabilities and local processing.

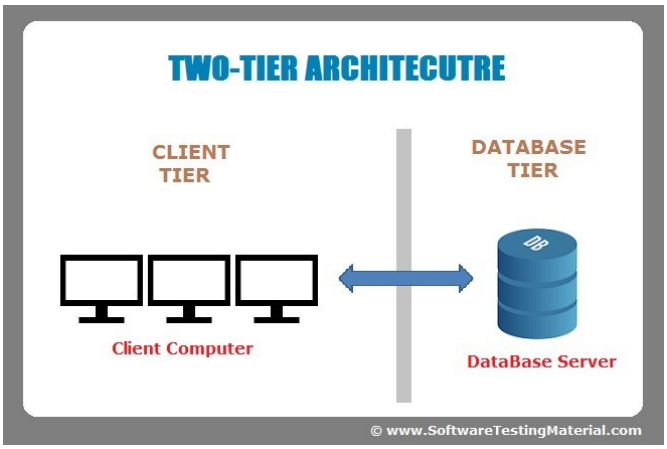


The server is a machine that can Provide services to the client machines, such as file access, printing and database access etc…

Two tier Client / Server Architecture:-

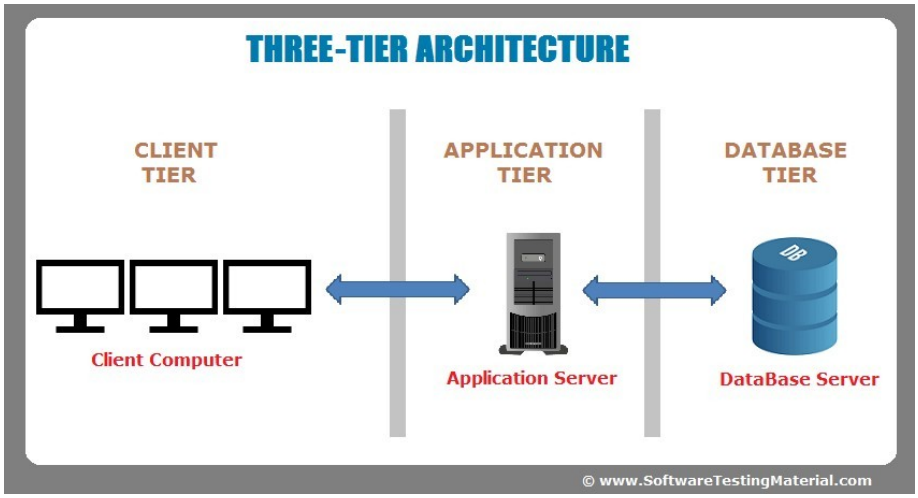
- It is called as two-tier architecture because the structure components ara distributed over two systems client and server.

- In this architecture the user interface program/application program can run on client machine.
- The query and transaction functionality remained on the server side.
- When DBMS access is required, the program establishes a connection to the DBMS .Once the connection is established, the client program can communicate with the DBMS.
- A standard called open database connectivity(ODBC) provides an application programming interface(API),which allows the client side program to call the DBMS.



Three-tier Client/Server Architecture:-

- An intermediate layer called as application server or web server is lies between the client and the database server.
- The web server play an intermediary role by storing business rules / procedures / constraints that are used to access the data from the database server.
- It can also improves database security by checking client's credentials before forwarding a request to the database server.
- The web server accepts request from the clients, process the request and sends database commands to the database server and then passing the processed data from the database server to the clients, where it may be processed further and filtered to be presented to users in GUI format.



The Entity-Relationship(ER) Model:-

- The database is used to store information that is useful for an organization and represents this information; some means of modeling is used.
- The ER model was developed to facilitate the database design that represents the overall logical structure of a database.
- The ER model is very useful is mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

- The database structure, employing the ER model is usually shown pictorially using entity-relationship(ER)diagram ,which shown how the schema for the database aplication can be displayed by means of the graphical notation .
- The ER data model employs 3 basic notations:
  Entity sets,relationship sets and attributes.

i. Entity sets:-

- An entity is a "thing" or "object" in the real world that is distinguished from all other objects. For example, each student in the student database is an entity.
- An entity has a set of properties and the values of the set of properties may uniquely identify an entity.For example , a student may have a roll number property whose value uniquely identifies that student.
- Entity set:- An entity set is a set of entities of the same type that share the same properties, or attributes.For example the set of all students in the student database can be defined as an entity set student.
- A database thus includes a collection of entity sets,each of which contains any number of entities of the same type.

ii. Attributes:-

- An entity is represented by a set of attributes.
- Each entity has a value for each of its attributes. For example possible attributes for the student entity are roll-no, name, age etc.
- Domain/value set:- For each attribute, there is a set of permitted values, called the domain or value set, of that attribute . For example, the domain of the attribute roll-no might be the set of all integers from 1 to 11.
- **Types of attributes:-**
  The following are the different types of attributes:-

  a) Simple and composite attributes:-
  The simple attributes have not been divided into subparts. For example, the roll number, age cannot be divided into supports, hence called as simple or atomic attributes.
  The composite attributes can be divided into subparts or attributes. This can form a hierarchy. For eg:,an attribute name could be structured as a composite attribute consisting of first_name, middle_name and last_name as 3 sub-ports.

  b) Single-valued and multi-valued atributes:-
  Single-valued attributes have single value for a particular entity. For example , the roll number attributes for a specific student refers to only one roll number.
  Multi-valued attributes means an attributes has a set of values for a specific entity. For example, a student may have zero, one or several phone numbers, and different employees may have different number of phones. This type of attribute is said to be multivalued.

  c) Stored and derived attributes:-

Stored attribute are the attributes from which we can derive another attribute. For example, the date-of –birth attribute from which we can derive the attribute age.

Derived attribute are the attributes that is derived from another attribute. For example, the age attribute can be derived from the attribute date –of-birth.

d) Complex attributes:-

Composite and multivalued attribute nested to form a complex attribute.

For example, if a person can have more than one residence and for each residence can have multiple phones, or attributes address phone can be specified as complex attribute.

iii. Keys:-

- A key is a single attribute or combination of two or more attributes of an entity set that is used to identify one or more instances of a set.
- A key allows us to identify a set of attributes that is used to distinguish entities from each other.
- The following are the different types of keys used:-

a. Primary key:-

It is a key that uniquely identifies an entity within an entity set.

The attribute roll-number is unique and will identify an instance of the entity set STUDENT. Such a unique entity identifier as STUDENT roll-number is referred to as a primary key.

b. Candidate key:-

There may be two or more attributes or combinations of attributes those uniquely identify an instance of entity set.These attributes or combinations of attributes are called as candidate keys.

For example ,the attributes roll-number and regd-no both can uniquely identifies an STUDENT entity.

c. Alternate key:-

One of the candidate key can be used as primary key.

The remaining candidate key could be considered alternate key.

d. Super key:-

If we add/remove additional attributes to a primary key, the resulting combination would still uniquely identifies an instance of the entity set .Such keys are called as super keys.

For example , the attribute set {roll no} is a primary key, that uniquely identifies an entity .However {roll no,name,age} is a super key , because by removing name and age or both from the set still leaves with a super key.

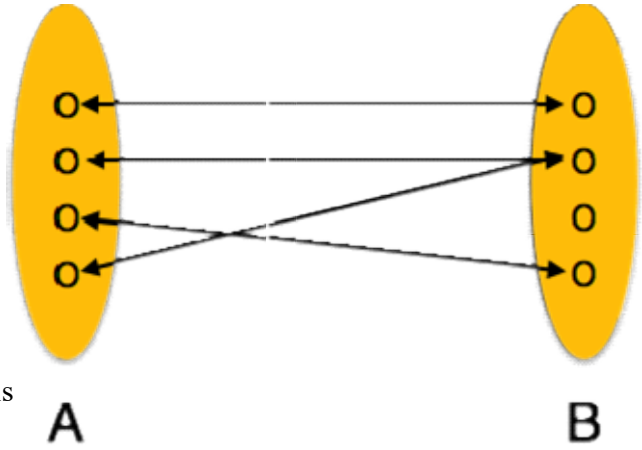
e. Secondary key:-

A secondary key is an attribute or combination of attributes that may not be a candidate key but that classifies the entity set on a particular characterstic.

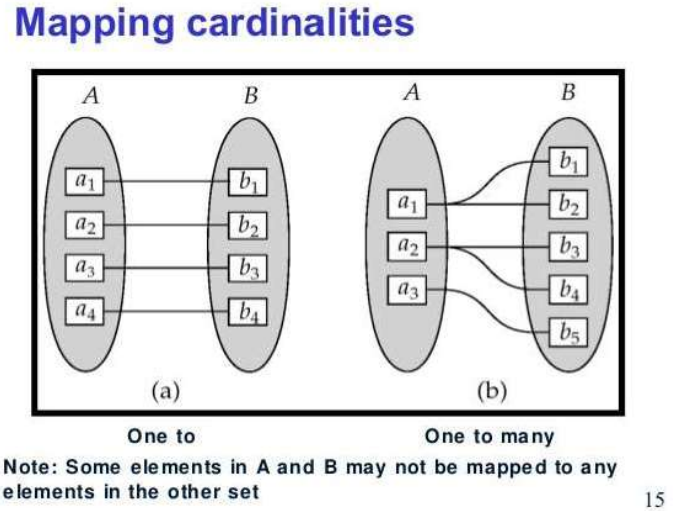
For example, an entity set EMPLOYEE having the attribute department which identifies by its value all instances of EMPLOYEE who belong to a given department.

iv. Relationship sets:-

- A relationship is an ssociation among several entities .Relationship type(R) between n entity E1,E2,..... En.
- A relationship set is a set of relationships of the same type .Formally, it is a mathematical notation of n>=2 entity sets.
- If (E1,E2,E3,…….En) are entity sets then a relationship set R is a subset of { (e1,e2,…..en)|e1
  Where (e1,e2,e3…en) is a relationship.
- For example ,for the two entity sets EMPLOYEE and DEPARTMENT , we can define the relationship type works-for, which associates each employee with the department for which the employee works.



- The association between the entity sets is referred to as participation i.e the entity sets EMPLOYEE , DEPARTMENT participate in the relationship set work_for. Mathematically , the entity sets E1,E2,E3,…..En participate in the relationship set R.
- The function that an entity plays in a relationship is called the entity's role.The role name signifies the role that a participating entity from the entity type plays in each relationship instance ,and helps to expain what the relationship means.
- **Recursive relationship set:-** When the entity sets of a relationship set are not distinct, that is the same entity set participating in relationship set more than once in different roles.In this type of relationship set ,it is called as the recursive relationship set.
- **Degree of relationship:-** Degree of the relationship type is the name of participating entity type in the relationship.
- **Mapping cordinalities:-(Cordinality ratios)**
  Mapping cordinality express the number of entity to which another entity can ecpress the number via a relationship set.
  For example , a binary relationship set R between entity sets A and B , the mapping cardinalities must be one of the following:-



A. One –to-One(1:1):-
An entity in A is associated with at most one entity in B, and an entity in B is associated

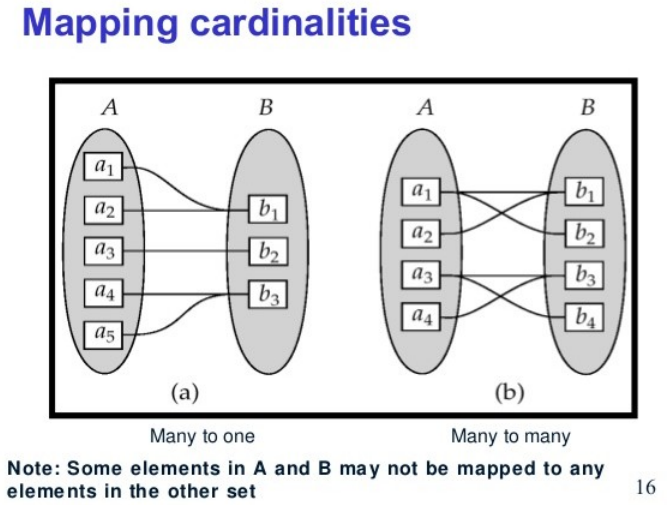
with at most one entity in A.

B. One –to-Many:-

An entity in A is associated with any number (zero or more) of entity in B, An entity in B ,however can be associated with at most one entity in A.

C. Many –to-One:-

An entity in A is associaated with at most one entity in B. An entity in B , however can be associated with any number of entities in A.



D. Many-to-Many:-

An entitty in A is associated with any number(zero/more) of entities in B, and an entity in B is associated with any number (zero/more) of entities in A.

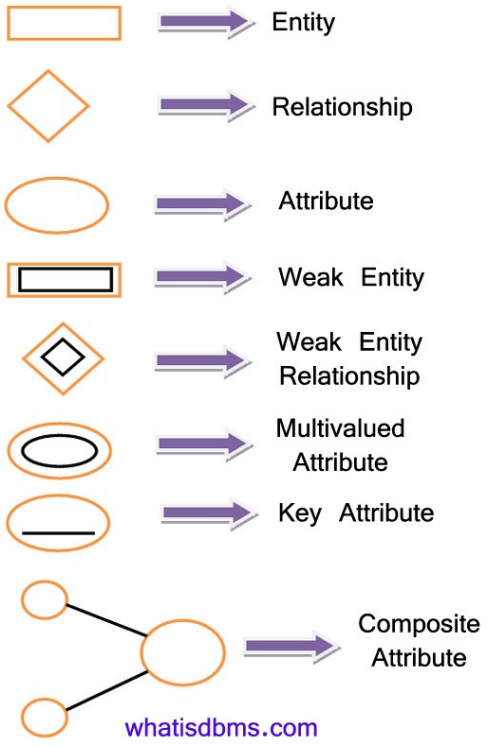
Participation Constraints:-

The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in atleast one relationship in R.

The participation of an entity set E in a relationship set R is said to be partial if only some in E participate in relationship in R.

Entity –Relationships Diagrams:-

An ER diagram can express the overall logical structure of a database graphically.The following are the different notations/components that are used in the ER diagram.



Strong Entity Type and Weak Entity type:-

- A regular entity type or strong entity type does have sufficient attribute to form a key attribute to form a key attribute /primary key .
- A weak entity type do not have sufficient attribute to form a key attribute .
- Entities belonging to weak entity type are identified by being related to a another entity type known as owner

entity type.

- A weak entity type always has a total participation with respect to its identification relationship,because a weak entity cannot be identified without an owner entity.
- For example ,consider the entity type DEPENDENT,related to EMPLOYEE,which is usedto keep the details of the employee's department. The attributes of DEPENDENT are Name, DOB, Sex and relationship to the employee.
- A weak enity type (in our ex. DEPENDENT) has a partial key(suppose name) which is a set of attribute that can uniquely identify the weak entities with the help of the strong entity type(Ex. EMPLOYEE)
- In ER-diagram, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines.
- The partial key attribute in underline with a dashed or dotted lines.

Extended ER Modeling:-

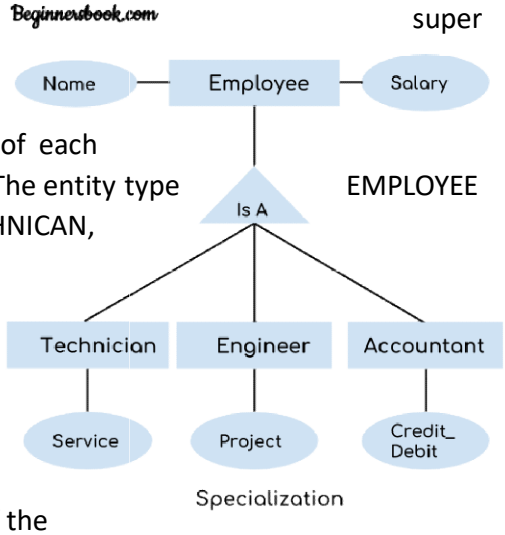
ER modeling concepts are sufficient for representing many traditional database application, but there are some more complex application are present such as telecommunication, GIS (Geographic Information System) ,CAD/CAM . These type of data database required complex requirement.

This led to the development of ad itional semantics data modelling concepts such as the EER model with tri-corporation with the ER model.

The extended features are specialization, generalization and aggregation.

Super Class and Sub Class relationship:-

- This is the inheritance concept, where we can say that all unity that is a member of a subclass which inherits all the attributes of the entity as a member of the super class and it also inherits all the relationship in which the super class participates.
- For example, the entity type EMPLOYEE describes the type of each employee entity, and also refers to the COMPANY database. The entity type EMPLOYEE (super class) has some sub groupings such as ENGINEER, TECHNICAN, SECRETARY (subclass).
- Here we can say that EMPLOYEE/ENGINEER, EMPLOYEE/TECHNICIAN and EMPLOYEE/SECRETARY ARE the super class/subclass relationships.



Specialization

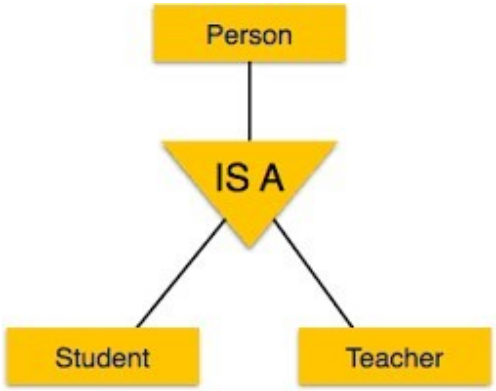
SPECIALIZATION:-

Specialization is the process of defining a set of subclasses of an entity type, this entity type is called as the super class of the specialization.

- For example, {ENGINNER, TECHNICAN, SECRETARY} is a specialization of the EMPLOYEE super class that distinguishes among EMPLOYEE entities based on the job type of each EMPLOYEE entity.
- In ER-diagram, specialization is depicted by a triangle component labeled ISA .The label ISA stands for "is a" and represents, for example, that a TECHNICIAN "is an" EMPLOYEE.

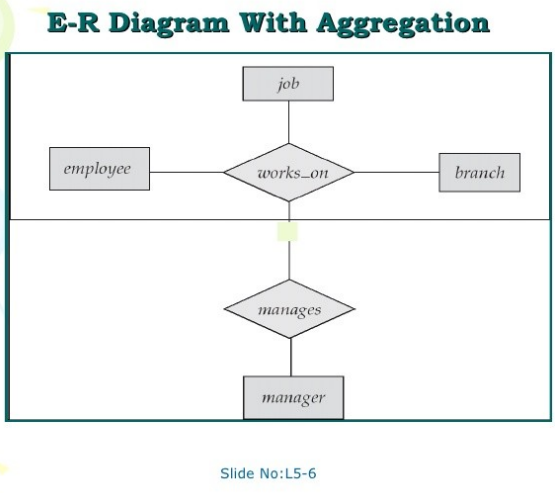
GENERALIZATION:-

- Generalization is the process of identifying the common features of the different entities i.e the subclass and generalizes them into a single super class.
- For example, consider the two entity types ENGINNER, TECHNICIAN which has several common attributes and they can be generalized to form the super class EMPLOYEE.
- A double line pointing to the generalized super class represents a generalization, where as simply arrow pointing to the specialized subclasses represents specialization.
- Generalization is a bottom up approach.



AGGREGATION

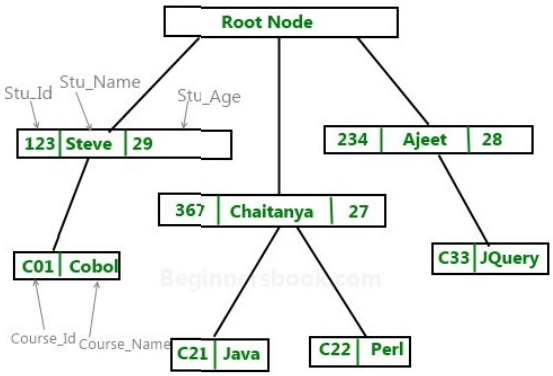
- The aggregation is the concept which expresses relationships among relationships.
- Aggregation is an abstraction through which relationships are treated as higher level entities.
- For example, the relationship set works-on (relating the entity sets EMPLOYEE, DEPARTMENT and JOB) as a higher level entity set called works-on. Such an entity set is treated in the same marks as in any other entity set. We can then create a binary relationship manages between works-on and manager to represent who manages what task.



DATA MODELS

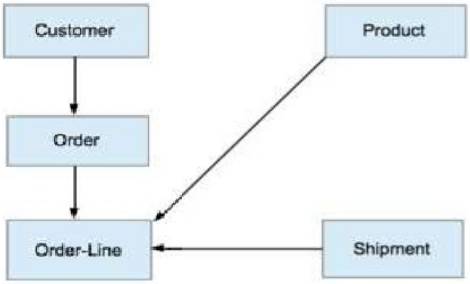
HIERARCHICAL DATABASE APPROACH/MODEL:-

- In the hierarchical database approach/model the data are in the hierarchical relationship.
- The data are stored in paper and file.
- Each customer section would contain folders for individual orders, and the orders would list each item being purchased.
- To store or retrieve data, the database system must start at top
- The difficulty of searching for item in the middle/bottom of the hierarchy.



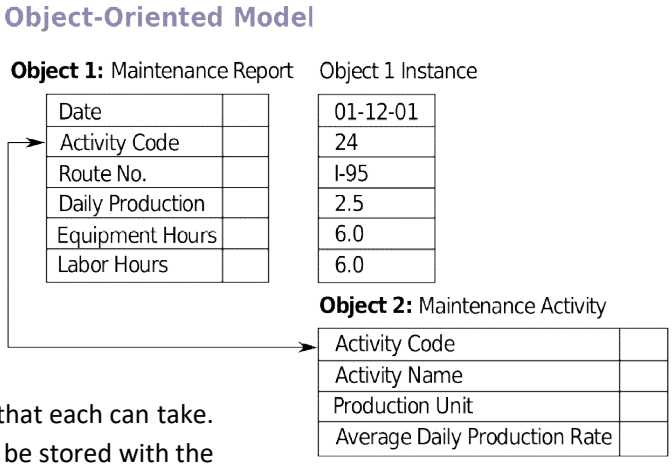
NETWORK DATABASE APPROACH/MODELS:-

- The network model is named from the network of connection between the data elements.
- The primary goal is to solve the hierarchical problem of searching for data.
- The item are now physically separated, they are connected by arrows.
- The model solves the search problem but the cost is high.



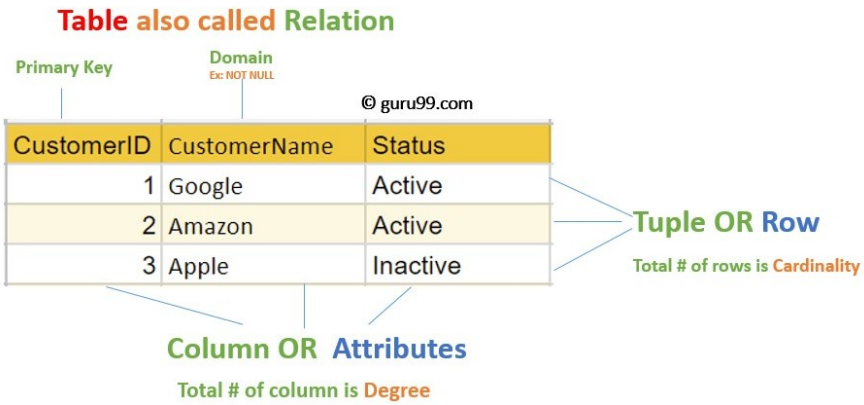
OBJECT ORIENTED DATABASE/APPROACH:-

- An object oriented database model is a new and evolving method of organizing data.
- An object has 3 major components:-
  - ✓ A name
  - ✓ A set of properties/attributes
  - ✓ A set of methods/functions
- The properties describe the object just as attribute describe an entity in the relational database.
- Methods are short programs that define the actions that each can take.
- For example, the code to add a new customer would be stored with the customer object.



RELATIONAL DATABASE APPROACH/MODELS:-

- The relational data model was first introduced by Ted Codd of IBM Research in 1970.
- The basic building blocks in the model is tables/relations.
- The relational model represents the database as a collection of relations, and each row in relations represents a collection of related data values.



- In the relational model, each row in the table represents a fact that typically corresponds to a real world entity/relationships.
- In the formal relationship relational model terminology, a row is called as tuple, a column header is called as attribute and the table is called as relation.
- The data types describing the type of data of values that can appear in each column is represented by a domain of possible values.
- Informally, a table is an entity set and a row is an entity.

CODD'S 12 RULES:-

Dr. Codd is an IBM Researcher, who first developed the relational data model in 1970's.

In 1985 he published a list of 12 rules that define an ideal relational database and has provided a guideline for the design of all relational database systems.

RULE :01 (INFORMATION RULE)

- All information in a relational database including table names, column names is represented by values in tables.
- The description of the tables and attributes definitions, integrity constraints everything is displayed in the tables.
- The simple view of data speeds up design and learning.

RULE: 02 (GUARANTEED ACCESS RULE)

- Every piece of data in a relational database, can be accessed by using a combination of a table name, a primary key using that identifies the row and a column name which identifies the cell.

RULE: 03 (SYSTEMATIC TREATMENT OF NULL RULE)

- A field should be allowed to remain empty. This involves the support of null values, which is distinct from an empty string or a number with a value of zero.
  This cannot be applied to primary key.

RULE: 04 (ACTIVE ON-LINE CATALOG BASED ON THE RELATIONAL MODEL)

- The description of a database and its contents are database tables and therefore can be queried on-line via the data manipulation languages (DML).

RULE: 05 (COMPREHENSIVE DATA SUB-LANGUAGE RULE)

- The RDBMS may support several languages, But at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity and transaction control.

RULE: 06(VIEW UPDATE RULE)

- Data can be preserved in different logical combination called views.
- Each view should support the same full range of data manipulation that has directed access to a table available.

RULE: 07(HIGH LEVEL, INSERT UPDATE AND DELETE)

- The RDBMS supports insertion, updation and deletion at a table level.
- This rule state that insert, update, delete operation should be supported for any retrievable set rather than just for a single row in a single table.

RULE: 08 (PHYSICAL DATA INDEPENDENCE)

- The execution of requests of application programs or queries is not affected by changes in the physical data access and stored methods.
- Database administrator can make changes to the physical access and storage methods, which improves performance and do not require changes in the application programs or requests.
- Here , the user specifies what he wants and need not worry about how the data is stored.

RULE: 09 (LOGICAL DATA INDEPENDENCE)

- Logical changes in table and views such as adding or deleting columns or changing field lengths need not necessitate modifications in the program or in the format of request/queries.
- For example, adding an attribute / column to the base table should not affect the program or the interactive commands that have no use for the new attribute.

RULE: 10 (INTEGRITY INDEPENDENCE)

- Like table and view definition, integrity constrains are stored in the on-line catalogue and can therefore be changed without necessitating changes in the application programs.
- The database languages (like SQL) should support constraints on the user input that maintain database integrity.

RULE: 11 (DISTRIBUTION INDEPENDENCE)

- Application programs and the request/queries are not affected by the changes in the distribution of physical data.
- This improves system reliability since application program will work even if the program and the moved to different sites.

RULE: 12 (NON-SUBVERSION RULE)

- If the RDBMS has a language that accesses the information of a record at a time , this language should not be used to bypass the integrity constraint.

RELATIONAL MODEL CONCEPTS

- A relational database consists of a collection of tables, formally known as relations and each of which is assigned a unique name.
- Formally, a row is called as tuple, a column header is called as attribute and the table is called as relation.
- Informally, a table is an entity set and a row is an entity.
- **DOMAIN:-**For each attribute, there is a set of permitted values called the domain of that attribute. For the attribute, branch name, for example the domain is the set of all branch names and it should be atomic.
- **ATTRIBUTE:-**The relation/ table having certain column headers and these column headers are called as attributes and these attributes having certain permitted value called as the domain of that attribute.
- **TUPLE:-**A row is specified as the tuple of the table. A tuple variable is a variable that stands for a tuple.
- **RELATION:-**A relation is thought of as a table of values, each row in the table represents a collection of related data values.
- **RELATION SCHEMA:-**
  - A relation schema is made up of a relation name and a list of attributes.
  - For example, we use STUDENT_SCHEMA to denote the relation schema for the relation STUDENT. Thus, STUDENT_SCHEMA=(ROLL, NAME, DEPT)
  - Mathematically,

    R = Relation

    $A_1, A_2, \ldots A_n$ = Attributes

    Each attribute A_i is the name of the role played by some domain D in the relation schema R. D is called the domain of A_i and is denoted by Dom (A_i).
  - A degree of a relation is the number of attributes n of its relation schema.
  - RELATION STATE:-
    - A relation state r of the relation schema R ($A_1, \ldots, A_n$) also denoted by r(R), is a set of n tupples $r=\{t_1,t_2,\ldots,t_n\}$.

- Each n tuple t is an ordered list of n values $t=<V_1,V_2,...,V_n>$. Where each value V_i is an element of $dom(A_i)$.

CONSTRAINTS AND KEYS:-

- Key is a constraint which can be enforced to the attribute, which is used to distinguish each and every tuple is a relation.
- Or, we can say that a key is a single attribute or combination of two or more attribute of an entity set that is used to identify one or more instances of the set.
- The attribute STUDENT.ROLL uniquely identifies an instance of the entity set STUDENT.
- The value 101 for the attribute STUDENT.ROLL uniquely identifies the student Sachin.

1. **PRIMARY KEY:-**
   - Primary key is used to uniquely identify a row of data in a table. This can be a single column of a table or a combination of more than one column.
   - The value off this attribute should unique.
   - It should not be null value.
   - For example, Let us consider a student relation having attributes ROLL_NO, NAME, AGE, and ADDRESS. In this relation more than one student cannot have the same ROLL_NO. Therefore, ROLL_NO is an attribute which is uniquely distinguish each and every tuple in this relation.
2. **CANDIDATE KEY:-**
   - These may be two or more attributes or combination of attributes that uniquely identify an instance of an entity set. These attributes or combination of attributes are called Candidate key.
   - The minimal super key is called as Candidate key.
   - For example, the attributes ROLL_NO and REGD_NO both can uniquely identifies an STUDENT entity.
3. **ALTERNATE KEY:-**
   - One of the candidate keys will be used as the primary key.
   - The remaining candidate keys would be considered as Alternate Keys.
4. **SECONDARY KEY:-**
   - A secondary key is an attribute or combination of attributes that may not be a candidate key but that classifies the entity set on a particular characteristic.
   - For example, the entity set/ relation EMPLOYEE with the attribute DEPARTMENT where DEPARTMENT attribute is not a candidate key for the relation EMPLOYEE, since it cannot uniquely identify an individual employee. However, the DEPARTMENT attribute is a secondary key that identify all employees belonging to a given DEPARTMENT.
5. **SUPER KEY:-**
   - If we add/remove additional attributes to a primary key, the resulting combination would still uniquely identifies an instance of the entity set. Such keys are called as Super key.

- For example, the attribute set {ROLL_NO} is a primary key that uniquely identifies an entity. However {ROLL_NO, NAME, AGE} is a super key, because by adding the name and age the resulting key still uniquely identifies an instance of the entity set.
- **ENTITY INTEGRITY CONSTRAINT:-**
  - The entity integrity constraint states that no primary key value can be null.
  - This is because the primary key value is used to identify individual tuples in a relation. Having null values for the primary key implies that we cannot identify some tuples.

REFERENCE INTEGRITY CONSTRAINTS:-

- To establish a parent-child relationship between 2 tables having a common column, we make use of referential integrity constraint.
- To implement this we should make the column of the parent table as primary key and the same column in the child table as a foreign key referring to the corresponding parent key.
- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.

6. FOREIGN KEY:-

A key column in a table that identifies records in a different table is called a foreign key.

The Relational Algebra and Relational Calculus

Refer from your class notes

RELATIONAL DATABASE DESIGN BY ER AND EER TO RELATIONAL MAPPING:--

There are steps of an algorithm to create a relation schema from an entity-relationship(ER) or an enhanced ER (EER) schema.

ER- to –Relational Mapping Algorithm:-

The following are steps of an algorithm for ER-to-relational mapping.

STEP:1(MAPPING OF REGULAR ENTITY TYPES)

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E and includes only the simple component attributes of a composite attributes.
- Choose one of the key attributes of E as a primary key for R.
- For example ,we have to create EMPLOYEE , DEPARTMENT and PROJECT in the relational database schema (i.e designed in the next page) to correspond to the regular entity types EMPLOYEE , DEPARTMENT and PROJECT of the ER diagram from the COMPANY DATABASE (i.e designed in the previous section)

- For example we have to choose ENO, DNO and PNUMBER as primary key for the relations EMPLOYEE, DEPARTMENT and PROJECT respectively.

STEP:2 (MAPPING OF WEAK ENTITY TYPES)

- For each weak entity type W in the ER schema with owner entity type E, create a relation R and includes all simple attributes (or simple components of composite attributes) of W as attributes of R.
- Includes the foreign key attributes of R the primary key attributes of the owner entity type E.
- The primary key of R is the combination of the primary key of the owner entity type (E) and partial key of the weak entity type (W) if any.
- For example we create the relation DEPENDENT to correspond to the weak entity type DEPENDENT. The primary key of the DEPENDENT relation is the combination of { EENO ,NAME } because NAME is the partial key of DEPENDENT.

STEP:3 (MAPPING OF BINARY 1:1 RELATIONSHIP TYPES)

- For each binary 1:1 relationship type R in the ER schema, identity the relations S and T that correspond to the entity types participating in R.
- One approach for this is the foreign key approach where choose one key of the relations –S, say and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1;1 relationship type R as attributes of S.
- For example we map 1:1 relationship type MANAGES by choosing the participating entity types DEPARTMENTS to serve in the role of S, because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation ENO as foreign key in the DEPARTMENT relation and rename its MGRENO.

STEP: 4 (MAPPING OF BINARY 1: N RELATIONSHIP TYPES)

- For each binary 1: N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R, this is done because each entity instance on the N-side is related to at most one instance on the 1: side of the relationship type. Include any simple attributes of the 1: N relationship type as attributes of S.
- For example The CONTROLS relationship is mapped to the foreign key attributes DNUM of PROJECT, which references the primary key DNUMBER of the DEPARTMENT relation.

RULE: 5 (MAPPING OF BINARY M:N RELATIONSHIP TYPES): For any binary M:N relationship type R , create a new relation S to represent R. Include as foreign key attributes in S the primary key of the relation that represents the participating entity types: their combination will form the primary key of S

RULE: 6 (MAPPING OF MULTIVALUED ATTRIBUTES)

- For each multi valued attributes A, create a new relation R. R will include an attributes corresponding to A , plus the primary key attributes K as a foreign key in R.
- The primary key of R is the combination of A and K .
- If the multi valued attribute is composite, we include its simple components.
- For example the primary key of DEPT_LOCATION is the combination of {DNUMBER<DLOCATION}.

RELATIONAL DATABASE DESIGN:-

NORMALIZATION:-

From the ER model a good relational database can be designed. But in the ER model there may be some amount of inconsistency, ambiguity and redundancy. To reduce this type of problem the normalization process is required.

INFORMAL DESIGN GUIDELINES FOR RELATIONAL SCHEMAS :(Why normalization is required)

The following are the some informal design guidelines for the relational schemas:-

i. **Semantics of relation attributes :-(guideline -1)**
   - Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relation types into a single relation.
   - **Insert**

ii. **Redundant information in tuples and update anomalies (irregularities):-**
   - Design the relation schema so that there should not be any redundant (duplicate) information.
   - Design the relation schema so that insertion of new data, deletion / updation of existing data should not be difficult.
   - GUIDELINE -2:- Design the relation schema so that no insertion, deletion or modification anomalies (irregularities) are present in the relations. If any anomalies are present, note them clearly, and make sure that the programs that update the database will operate correctly.

iii. **Null value in tuples :- (guideline -3)**
   - As far as possible, avoid placing attributes in a relation whose values may frequently be null. If nulls are unavoidable, make sure that they apply in exceptional cases and do not apply to a majority of tuples in the relation.

iv. **Generation of spurious tuples :-(guideline -4)**
   - Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys are generated.
   - Avoid relations that contain matching attributes that are not primary key or foreign key combinations, because joining on such attributes may produce spurious tuples.

FUNCTIONAL DEPENDENCIES:-

MEANING: - Suppose a relation / table R contains of two attributes X and Y , we can say that Y is functional depends upon attributes X if and only if X uniquely determines the value of Y.

i.e for the two tuple t_1 and t_2 in the relation R

if $t_1[X] = t_2[X]$

then $t_1[Y] = t_2[Y]$

- So, this means the values of Y component of a tuple is determined by the values of X component of a tuple.
- We can say that there is a functional dependency from X to Y is functionally dependent on X.
- The abbreviation for functional dependency is FD or f.d and it can be represented by X→Y.
- The set of attributes X is called the left-hand side of the FD, and Y is called the right –hand side.

Example

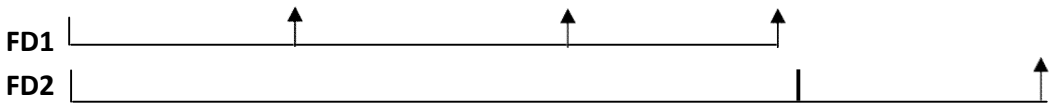
- lets us consider the following example :-

EMPLOYEE				
ENO	**ENAME**	**ADDRESS**	**DOB**	**AGE**

Here, by using the key attributes ENO we can derive the other attributes ENAME, ADDRESS and DOB and by using the ENO and DOB we can derive the attribute AGE.
It can be represented by
FD1: ENO → {ENAME, ADDRESS, DOB}
FD2: {ENO, DOB} → AGE

- The above two functional dependency FD1 and FD2 can be represented graphically as :-

EMPLOYEE				
ENO	**ENAME**	**ADDRESS**	**DOB**	**AGE**



- Each FD is displayed as a horizontal line. The left hand side attributes of the FD are connected by vertical lines to the lines represented the FD, while the right-hand side attributes are connected by arrows pointing towards the attributes.
- **Definition :-** A functional dependency, denoted by X→Y, between two sets of attributes X and Y that are subset of R, specifies a constraint and that is for any two tuples t_1 and t_2 that have $t_1[X] = t_2[X]$ then they must also have $t_1[Y] = t_2[Y]$.

FULL FUNCTIONAL DEPENDENCY:-

- A functional dependency (FD) X→Y is a full functional dependency (FFD) if removal of any attributes A from X means that the dependency does not hold any more.
- {ENO, DOB} → AGE
  Here, the combination of ENO and DOB uniquely identifies the AGE of the employee. Here age cannot be determined by the use of DOB or ENO only.

PARTIAL FUNCTIONAL DEPENDENCY:-

- A Functional dependency (FD) X→Y is partial functional dependency if some attributes A ∈ X can be removed from X and the dependency still holds.
- For example, {ENAME, ENO} → ADDRESS
  Here, if we remove ENAME then ENO→ADDRESS still holds.

INFERENCE RULES FOR FUNCTIONAL DEPENDENCY: -

MEANING: -

- Suppose, F is set of functional dependencies that are specified on relation schema R. It is not possible to specify all possible functional dependencies for a given situation .
- For example if each department has one manager so that DEPT_NO uniquely determines MANAGER_ENO (DEPT_NO →MGR_ENO),
  And a manager has a unique phone number called MGR_PHONE (MGR_ENO→MGR_PHONE), then these two dependencies together imply that MGR_PHONE (MGR_ENO →MGR_PHONE), then these two dependencies together imply that DEPT_NO →MGR_PHONE. This is an inferred FD and need not be explicitly stated in addition to the two given FDs.
- Therefore formally it is useful to define a concept called closure that includes all possible dependencies that can be inferred from the given set F.

DEFINATION

The set of all dependencies that includes F as well as all dependencies that can be inferred from F is called the closure of F, it is denoted by F^+.

Example

F = ENO → {ENAME, DOB, ADDRESS, DNUM}

DNUM → {DNAME, ADDRESS}

F^+= ENO → {DNAME, DMGRNO}

To determine a systematic way to infer dependencies we must discover a set of inference rules that can be used to infer new dependencies from a given set of dependences F.

There are six inference rules are present. These are: -

ARMSTRONG'S AXIOMS OR INFERENCE RULES: -

I. **IR1(REFLEXIVE RULE): -**
For a relation R having attributes X and Y.
If $X \supseteq Y$, then $X \rightarrow Y$
Proof: - Suppose $X \supseteq Y$ and two tuples t_1 and t_2 exist in some relation, such that, $t_1[X]=t_2[X]$ then, $t_1[Y]=t_2[Y]$, because $X \supseteq Y$
Hence, $X \rightarrow Y$ must hold in this relation.

II. **IR2(AUGMENTATION RULE): -**
For a relation R having attributes X and Y
If $\{X \rightarrow Y\}$ then $XZ \rightarrow YZ$
Proof: - Let $t_1[X]=t_2[X]$ ------ **(1)**
Then, $t_1[Y]=t_2[Y]$ ------- **(2)**
And we can say that $t_1[XZ]=t_2[XZ]$ --------- **(3)**
From equation 1 and 2, we can say that, $t_1[Z]=t_2[Z]$ ------ **(4)**
From equation 2 and 4 we can say that, $t_1[YZ]=t_2[YZ]$
So from the above equation we can say that $\{X \rightarrow Y\}$, then $XZ \rightarrow YZ$.

III. **IR3(TRANSITIVE RULE): -**

For a relation R having attributes X and Y ,
If $\{X \rightarrow Y, Y \rightarrow Z\}$ then, $X \rightarrow Z$
Proof: -
Assume that $X \rightarrow Y$ and $Y \rightarrow Z$ both hold in relation R then
we can say that, $t_1[X]=t_2[X]$ and $t_1[Y]=t_2[Y]$
and we can say that $t_1[Z]=t_2[Z]$
hence, $X \rightarrow Z$ must hold in relation R.

IV. **IR4(DECOMPOSITION /PROJECTIVE : -**
For a relation R having attributes X and Y,
If $\{X \rightarrow YZ\}$ then, $X \rightarrow Y$
Proof: -
Given, $X \rightarrow Y$--------- **1**
$YZ \rightarrow Y$ from IR1---------- **2**
Thus from 1 &2 , we can say that $X \rightarrow Y$

V. **IR5(UNION / ADDITIVE RULE): -**
For a relation R having attributes X and Y,
If $\{X \rightarrow Y, X \rightarrow Z\}$ then, $X \rightarrow YZ$

Proof

Given X→Y and X→Z ---------- 1

X→XY from IR2 ---------- 2

XY→YZ from IR3 ----------3

X→YZ using IR3 AND EQUATION 2 AND 3

VI. IR6 (PSEUDOTRANSITIVE RULE) : -

For a relation R having attributes X and Y,

If {X→Y, WY→Z} then WX→Z

PROOF

Given

X→Y ---------- 1

WY→Z ---------- 2

WX→WY (using IR2)---------- 3

WX→Z (USING IR3 AND EQUATION 2 AND 3)

CLOSURE:-

A systematic way to determine the addition fractional dependencies is first to determine each set of attributes X that appears in the left hand side of some functional dependencies in F and then to determine the set of all attributes that are dependent on X.

Thus, for each set of attributes X, we determine the set X^+ of attributes that are functionally determined by X based on F, X^+ is called the closure of X under F.

Example-

F = ENO → ENAME

DNO →{DNAME, DLOCATION}

X^+ (CLOSURE SET) ={ ENO} +={ENO, ENAME}

{DNO} +={DNO, DNAME, DLOCATION}

COVERED BY FUNCTIONAL DEPENDENCY: -

A set of functional dependency F is said to cover another set of functional dependencies E if every FD in E is also in F^+ that is if every dependency in E can be inferred from F then we can say that E is covered by F.

EQUIVALENT: -

Two sets of functional dependencies E and F are said to be equivalent if $E^+=F^+$. Hence equivalence means that every FD in E cab be inferred from F and every FD in F can be inferred from E that is E is equivalent to F if both the condition E covers F and F covers E hold.

- Normalization is the process of efficiently organizing data by analysing and decomposing the complex relation to form a simple relation.
- Normalization is the process of analysing the given relation schemas based on their functional dependencies and primary keys to achieve the desired properties like
    - ✓ Minimizing redundancy
    - ✓ Minimizing the insertion, deletion and update anomalies.
- Normalization is carried out for the following reasons: -
    - ✓ To structure the data between relations as maintenance is simplified.
    - ✓ To allow data retrieval at optimal speed.
    - ✓ To reduce the need to restructure relations as application requirements arise.
    - ✓ To improve the quality of design for an application
    - ✓ To minimizing the insertion, deletion, and update anomalies.
    - ✓ To structure the data between relations so that data redundancy can be minimized
- Normal form was first introduced by E.F.Codd and R.Boyce in 1972. The following are the five types of normal form.

➢ **FIRST NORMAL FORM (1NF): -**

- A relation is said to be in first normal form, if the domain of an attribute must include only atomic (simple, individual) values.
- It disallows multi-valued and composite attribute.
- Steps to convert the relation into 1NF: -
    - ✓ Eliminate duplicate columns from the same relation.
    - ✓ Create a separate relation for each set of related data and identify each set of related data with a primary key.

➢ **SECOND NORMAL FORM (2NF): -**

- A relation is in 2NF if: -
  It is in 1NF
  It includes only that attribute which is fully dependent on its primary key.
- Steps to convert a relation to its 2NF is: -
    - ✓ Find and remove the fields that are related to the only part of the primary key.
    - ✓ Group the removed items in another relation.
    - ✓ Assign the new relation with a primary key

Definition: A relation schema R is in 2NF if every nonprime attribute A in R is fully dependent on the primary key of R.

➢ **3RD NORMAL FORM(3NF): -**

- A relation is said to be in 3NF if: -
  It is in 2NF and
  There exists no transitive functional dependency.
- **Definition:** A relation schema R is in 3NF if it satisfies 2NF and no nonprime attributes of R is transitively dependent on the set of primary key.

- **Transitive dependency:** X→Y in a relational schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X→Z and Z→Y holds .

BOYCE –CODD NORMAL FORM: -(BCNF)

- A relation is said to be BCNF if and only if all the determinants are candidate keys.
- **Definition:** A normalized relation schema R(A,F)

Where, A = set of attributes

F= set of functional dependencies is in BCNF if for every FD in F^+ of the form X →Y where X belongs to A. and y belongs to A, Z is a super key of R.

RELATIONAL DECOMPOSITION: -

For a good database design the database schema must satisfy the normal form by decomposing the relation schema and must satisfy certain properties: -

- Dependency preservation property
- Lossless / non additive join property

Dependency preservation property: -

- By using the functional dependencies, the universal relation schema R is divided/decomposed into a set of relation schema D ={R_1, R_2 ……,R_n}
- This property states that each functional dependency X→Y specifies in F that appeared directly in one of the relation schema R must be appeared in one of the relation in the decomposition D.
- So that no dependency are lost.
- It is sufficient that the union of the dependencies that hold in the individual relation in D be equivalent to f of R.
- Suppose
  For a relation R ={A_1 ,A_2 ,…..A_n}
  The set of FD in F (X →Y)
  Decomposition (D) = {R_1, R_2 ……,R_n}
  Then the property says that
  $(R_1(F) \cup R_2(F) \cup \ldots\ldots\ldots \cup R_n(F))^+ = F^+$

Lossless/non additive join property: -

- This property ensures that no spurious tuples are generated when a natural join operation is applied to the relations in the decomposition.
- Let R be the relation schema
  F be the set of functional dependencies on R.
  R_1 and R_2 be the decomposition of R

r be the relation instance with schema R

Then we can say that the decomposition is a loss less decomposition if

$R1 \bowtie R2 = R$

- If we project r onto R_1 and R_2 and compute the natural join of the projection results, we get back exactly r.
- The decomposition that is not a lossless decomposition is called as **lossy decomposition.**

MULTIVALUED DEPENDENCIES: -

- The multivalued dependency relates when more than one multi valued attributes exist.
- Functional dependency X→Y relates one value X to one value Y while multivalued dependency X→>Y defines a relationship in which a set of values of attributes Y are determined by a single value of X.

Refer your class notes for example

- **Definition :**

  The multivalued dependency X→>Y is said to hold for a relation R(X,Y,Z) if t_1 and t_2 are two tuples in R that have the same values for attributes X and therefore with $t_1[X] = t_2[X]$ then R also contains tuples t_3 and t_4 such that : -

$t_1[X] = t_2[X] = t_3[X] = t_4[X]$

$t_3[Y] = t_1[Y]$ and $t_3[Z] = t_2[Z]$ then

$t_4[Y] = t_2[Y]$ and $t_4[Z] = t_1[Z]$

In others words

$t_1 = [X, Y1, Z1]$ AND

$t_2 = [X, Y2, Z2]$

Then there must be tuples t_3 and t_4 such that

$t_3 = [X, Y_1, Z_2]$ AND

$t_4 = [X, Y2, Z1]$

TRIVIAL FUNCTIONAL DEPENDENCY

A functional dependency FD: X→Y is called trivial if Y is a subset of X or X and Y together form a relation R.

In a relation like EMP(EMPID,ENAME) has a relationship between EMPID and ENAME in which EMPID uniquely determines the value of ENAME, the dependence of ENAME on EMPID is called trivial FD since the relation EMP cannot be decomposed any further.

- **FORTH NORMAL FORM (4NF): -**

A relation R is in 4NF with respect to a set of dependencies F(that includes functional dependencies and multivalued dependencies) and if, when multivalued dependency X →>Y holds then either

- The dependency is trivial or
- X is a candidate key for R

- **FIFTH NORMAL FORM(5NF)/PROJECT JOIN**
  - The aim of fifth normal form is to have relations that cannot be decomposed further.
  - A relation R is in 5NF with respect to a set F of functional multivalued dependency if for all join dependencies at least one of the following holds : -
    A) R(R1,R2,R3,……Rn) is a trivial join dependency.
    B) Every R_i is a candidate key for R.

Storage Strategies and File Organizations

Data Storage:-

The collection of data that makes up a computerised database must be stored physically on some computer's storage medium. Computer storage media forms a storage hierarchy that includes two main categories.

(i)**Primary Storage:-**This categories includes storage media that can be operated on directly by the CPU, such as the computer main memory and smaller But faster Cache memories.

*Primary storage usually provides fast access to data but is of limited storage capacity.

(ii)**Secondary Storage:-**

*This category includes magnetic disks, optical disks, and tapes.

*Those devices usually have a larger capacity, cost less and provide slower access to data than primary storage.

Indexing Structures for files:-

Indices:-

*Database indices play the same role as the book.

*For example to retrieve an account record given the account number, the database system would look up an index to find in Which disc block the corresponding Record resides and then fetch the disc block, to get the Account record.

*There are two basic kinds of Indices:-

(i) **Ordered indices: -** Based on a sorted Ordering of the values.

(ii) **Hash indices: -** Based on a uniform distribution of values across a range of Buckets.The bucket to which a value is assigned is Determine By a function Called as Hash function.

Ordered Indices:-

* To gain fast Random Access to record in a file we can use index Structure.

* Each Index structure is associated with a particular Search key.

* Ordered index Stores the values of the search Keys In sorted Order and Associates with each search key the records that contain it.

*Two types of Ordered index Are used:-

A) Primary Index/Clustering index.

B) Secondary Index/Non-clustering Index.

(a) Primary Index/Clustering index:-

*If the file containing the records is sequentially ordered, then a clustering index is an index Whose Search key Also defines the sequential Order of the file.

*The search key of Clustering index is often the primary key.

*Two types of primary index we can use:-

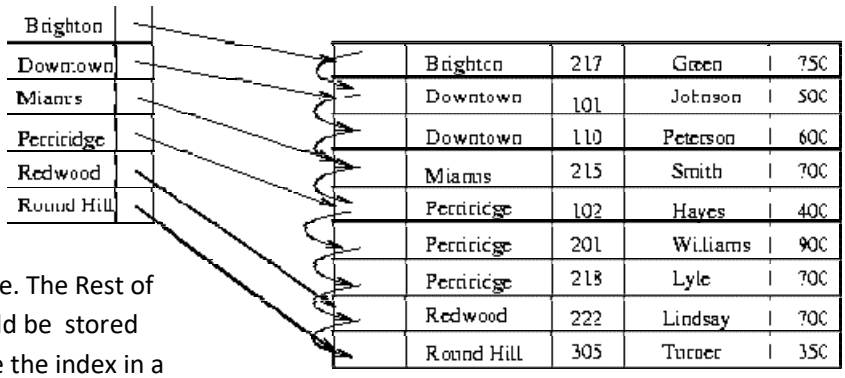
-Dense Index

-Sparse Index

Dense Index:-

*An index record appears for every search key Value in the file.

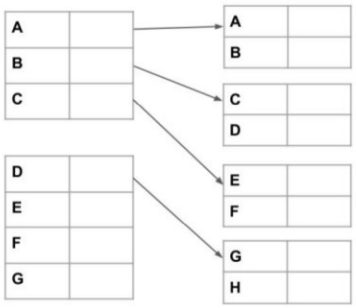
*In a dense Clustering index the index record Contains the search Key value And a pointer to the first Data record with that Search key value. The Rest of the records with the same search Key value would be stored sequentially after the First record, since, because the index in a clustering one, Records are stored on the same search key.



Sparse Index:-

*An index record appears for only name of the search key values are called as sparse index.

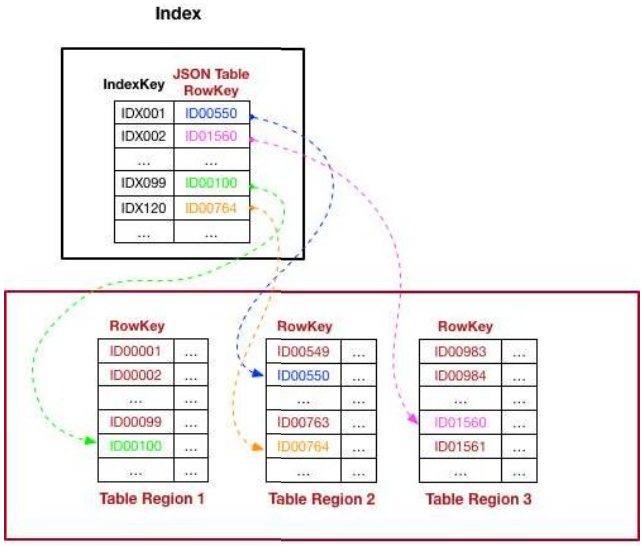
→It is generally faster to locate a record if we have a dense index rather than a sparse index.



Sparse Index

→Sparse indices require less space and they impose less maintenance overhead for insertions and deletions.

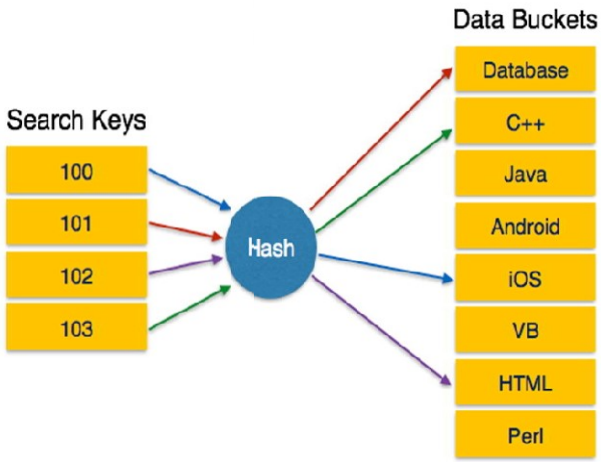
b) Secondary Index/Non-clustering Index:-



*secondary indices must be dense, with an index entry for every search key value, and a pointer to every record in the file.

*A secondary index, if the search key is not a candidate key, it is not enough to point to just the first record with each search key value. The remaining records with the same search key value could be anywhere in the file, since the records are overloaded by the search key of the clustering index, rather than by the search key of secondary index. Therefore the secondary index must contain pointers to all the records.

(ii)Hash Indices/Hashing Techniques:-



*Another type of primary file organisation is based on hashing, which provides very fast access to records on certain search conditions .this organisation is usually called a hash file.

*The search condition must be an equality condition on a single field, called the hash field of the file. It is called also as key field/hash key.

*In this technique a function called a hash function or randomizing function, that is applied to the hash field value of a record and yields the address of the disk block in which the record is stored.

*Two types of hashing techniques are there:

A) Internal hashing.

B) External hashing.

A) Internal hashing:-

*The internal hashing is implemented as a hash table through the use of array of records.

*Suppose that the array index range in form 0 to n-1, then we have n slots whose address corresponds to the array index.

*We choose a hash function that transforms the hash field value into an integer between 0 and n-1.

*One common hash function is the h (k) =k mod n function which returns the remainder of an integer hash field value k after division by n; this value is then used for the record of address.

B) External hashing:-

*hashing for disk files is called external hashing.

*The target address space is made of buckets, each of which holds multiple records.

*A bucket is either one disk block or a cluster of contiguous blocks.

*A hashing function maps a key into a relative bucket number, rather than assign an absolute block address to the bucket. A table maintained in the file header converts the bucket number into the corresponding disk block address.

*The collision problem is less severe with buckets, because as many records as will fit in a bucket can hash to the same bucket without causing problems.

Query Processing and Optimization

Query Processing:-

*Query processing refers to the range of activities involved in extracting data from a database.

*A query expressed in a high level query language such as SQL must first be scanned, parsed and validated.

Steps while processing high level query:-

*when a query written in a high level language than first it is scanned

*the scanner identifies the language taken such as SQL keywords, attributes and relation name

*the parser checks the query syntax to determine whether the query is formulated according to the syntax rules (rules of grammar) of the query language.

*then the query must be validated, by checking that all attributes names and relation name are valid and semantically meaningful name.

*an internal representation of the query is created usually pre data structure called query tree or a graph data structure called query graphs.

*then the DBMS must then make an execution strategy /plan. A query typically has possible execution strategy and the process of choosing a suitable one for the processing a query is called as query optimization.

*then basing upon the query execution plan the code generator generates the code to execute the plan.

*finally the runtime database processor has the task of running the query code whether in compiled/interpreted mode.

*if the runtime error occurs an error message is generated by the runtime data base processor.

Query optimization:-

*we do not expect users to write the query to show that they can be processed efficiently .rather we expect a system to construct a query execution plan that minimizes the cost of query execution .this is where the query optimization to play.

*Query optimization is the process of selecting the most efficient query execution plan from among the many strategies is usually possible for processing a given query especially if the query is complex.

*There are two main techniques for implementing query optimization:-

→The first technique is heuristic rules for ordering the operation in a query execution strategy. This involves mains query tree/ query graph.

→The second technique involves systematically estimating the cost of different execution plan/strategy and choosing the best with the lower cost estimation.

Translating SQL queries into Relational Algebra:-

Heuristic Rules in Query Optimisation:

- The optimisation technique that apply heuristic rules to modify the internal representation of a query ,which is usually in the form of a query tree/ query graph data structure to improve the expectation performance.
- The parser pf a high-level query first generates an initial representation which is then optimised according to the Heuristic rules.

Query Tree:-

- A query tree is a tree data structure that corresponds to a relational algebra expression or extended relational algebra expression.
- A query tree represents the input relation of the query as leaf nodes of the tree ,and represents the relational algebra operations as internal nodes.
- An execution of the query tree consists of executing an interval node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.
- The execution terminates when the root node is executed and produces the result relation for the query.

For Examples refer your class notes

Query Graph:

- Query graph is used to represent a relational calculus expression.
- Relations in the query are represented by relation nodes which are displayed as single circles.
- Constant values, typically from the query selection conditions are represented by constant nodes ,which are displayed as double circles.
- Selection and join condition are represented by the graph edges.