Database Management System

Mr. Rama Krushna Padhy
Computer Science (Diploma)

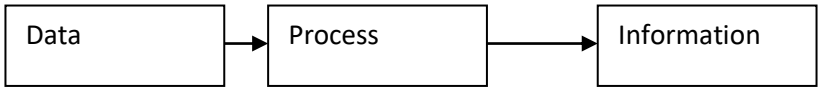
MODULE – 1

❖ Data

Any raw fact figure which has some meaning with respect to this real world is called as data, which can be recordable. Ex: Mobile NO: 23, room etc.

❖ Information

- It is a refined data.
- When data becomes processing or meaningful called information.



❖ Difference between data and information

Data:

- Data is raw fact and figure.
- Data is not significant to a business and of itself.
- Data are atomic level pieces of information.
- Data does not help in decision making.

Information:

- Information is a process form of data.
- Information is significant to a business and of itself.
- Information is a collection of data.
- Information helps in decision making.

❖ Database:

- Collection of interrelated data is called database.
- The main feature of data in a database are:
  I. It is well organized.
  II. It is related.
  III. It is accessible in different orders.
  IV. It is stored only once in particular order in an organized form.

❖ Database management system(DBMS):

D=Data: Any raw fact figure

B=Base: where stores data

M=Management: Managing the database, which includes insertion, deletion, updating, sorting etc.

S= System: Software that helps to manage the database.

❖ DBMS:

- It is a collection of interrelated data and programs to access those data.
- It is a computerized record keeping system.
- It is software which assists in maintaining and utilizes a database.

- The goal of DBMS is to provide a way to store and retrieve database information in a convenient and efficient manner.

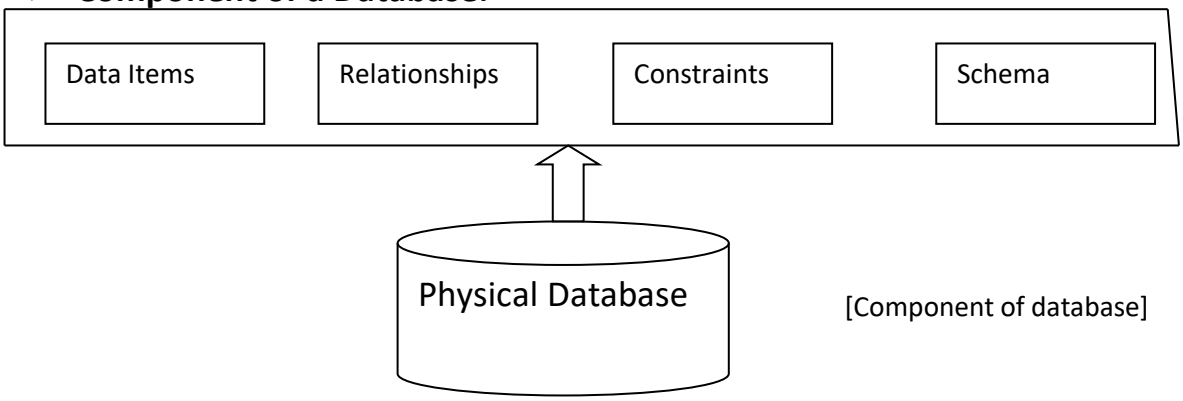
❖ Advantages of DBMS:

- Controlling redundancy.
- Integrity can be enforced.
- Inconsistency can be avoided.
- Data can be shared.
- Restricting unauthorized access (Security).
- Solving enterprise requirement.
- Providing backup and recovery.
- Accuracy can be achieved.

❖ Disadvantage of DBMS:

- Complexity
- Cost of DBMS, hardware cost etc.
- Size

➢ Component of a Database:

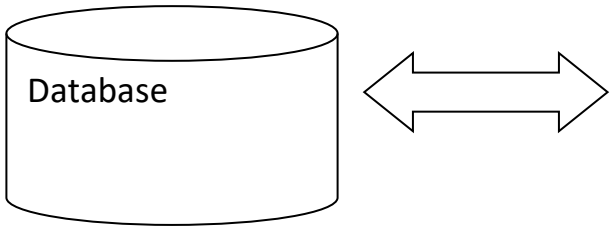


[Component of database]

A database consists of the following and components.

I. Data items: It is a distinct piece of information.
II. Relationship: It represents a communication between various data elements.
III. Constraint: These are predicates that define correct database state.
IV. Schema: It describes the organization of data and relationship within the database.

➢ Functions of find services of DBMS:



[Services provided by DBMS]

Data catalog management
Data manipulation management
Transaction management
Concurrency control
Recovery management
Security management
Language interface
Storage management
Data access

i. **Data catalog management:**

Data catalog or data dictionary is a system database that contains descriptions of the data in the database (Metadata).

ii. **Data manipulation management:**

A DBMS furnishes users with the ability to retrieve, update and delete existing data in the data base.

iii. **Transaction management:**

- A transaction is sequence of database operations, carried out by a single user or applications program, which access or changes the content of the database.
- When DBMS does a commit, the changes mode by transaction are mode permanent.

iv. **Concurrency control:**

- It co- ordinate the actions of database manipulation processes that operate concurrently and access shared data.
- The goal of this is to allow concurrency while maintaining the consistency of the shared data.

v. **Recovery management:**

- The DBMS provide mechanism for backing up data periodically and recovering from different types of failures.
- This prevents the loss of data.

vi. **Security management:**

- Security refers to the protection of data against unauthorized access.
- It ensures that only authorized users are given access to the data in the database.

vii. **Language interface:**

DBMS provides languages for definition and manipulation of data in the database.

viii. **Store management:**

- It provides a mechanism for management of permanent storage of data.
- Internal schema defines have data should be stored by storage management, mechanism and storage management interfaces with the operating system to access the physical storage.

ix. **Data access:**

DBMS provides data access via structured Query Language (SQL).

❖ **Data dictionary:**

It is a Meta data (Data about Data) repository defined as “centralized repository of information about data such as meaning, relationships other data, origin, usage and format”.

Benefits of data dictionaries

The data dictionary helps to maintain a better database because it

a. Avoids data inconsistency
b. Avoids data redundancy
c. Improve data quality

Types of the data dictionary:

There are two types of data dictionaries

i. **Active dictionary** is integrated into the DBMS will be updated automatically when the data structure is changed and always remains correct.
ii. **Passive data** dictionary is not integrated into the DBMS. It will be implemented using a separate program changes in database will not be reflected automatically. In DBMS, data dictionary is often called system catalog.

❖ Database users:

The main objective of the database system is to store new data in the database and retrieve data from the database. So there must be some users for storing and retrieving information from the database.

1. **Naive/ end user:**
   They are normal users, interacting with the database directly on online or indirectly through some interface.
2. **Application Programmer:**
   These are creators of applications interface. By using different language, they are creating application programs.
3. **Database administrator (DBA):**
   DBA is a person who has get central control over the database and responsible for implementing the administrative decision.

❖ Data independence:

- It is the ability of change the schema at one level of database system without affecting the schema at higher level.
- There are two types of data independence.

1. **Logical data independence:**
   It is the capacity to change the conceptual schema without affecting the schema at logical level
2. **Physical data independence**:
   It is the capacity to change the internal schema without affecting the schema at logical level.

❖ Schema and instances:

The overall design of a database is called as schema of database.
Ex: Schema for student database is

Roll NO	Name	Branch	Semester

The collection of information stored in a database at a particular moment is called as an instance of a database. The plan for a view is called subschema. It is a subset of schema and inherits the same property that a schema.

❖ Database language:

- Database languages are used to create and maintain database on computer.
- Query is request to the database by using some language called Query language.
- In database, we are using SQL structured Query language as query language.
- It can be categorized as:

a. Data definition language (DDL):

- It is a language that allows the users to define data and their relationship to other types of data.
- It is mainly used to create files, databases, data dictionary and tables within databases.
- Example of DDL are:

i. CREATE:

To create schema objects in the database.
Syntex: CREATE table <table-name>
(attribute datatype (width), attribute datatype(width)..);
Ex: CREATE table Customer
(CustID char(4), C-Name char(20), C-Add char(25), C-Mob number(10));

ii. ALTER:

It modify the structure of the database.
ALTER table <table-name>
Modify (attribute datatype (width), attribute datatype(width)..);
ALTER table <table-name>
add (attribute datatype (width), attribute datatype(width)..);

iii. **DROP:**

It delete object from the database.
Syntex: Drop table <table-name>

iv. **RENAME:**

It renames the object of a database.
Syntex: RENAME <source table-name> to <destination table-name>;

b. Data manipulation language(DML):

- It provides a set of operations a support the basic data manipulation operations on the data held in the database.
- It allows users to insert, update, delete, and retrieve data from the database.
- Example of DML are:

i. **DELETE:**

It removes records/rows from the table
Syntax: Delete from <table name>
Where <col- name = value>;

ii. **INSERT:**

It add new rows of data into table.
Syntax: INSERT into <table name> values (values, values,..);

Ex: INSERT into Customer values (3,'sam','BBSR', 5000);

iii. **UPDATE:**

It change column values in existing rows of a table.

Syntax: UPDATE <table name>

Set <col-name = value>

Where <col-name = value>;

Ex: UPDATE Customer set C-Bal = 10000 where Cust-ID = 151;

iv. **SELECT:**

it retrieve data from one or many table.

Syntax: SELCT * from <table name>;

c. Data control language (DCL):

- It controls the access to data and the database.
- Example of DCL are:
  - **i. GRANT:** Gives privileges to the user.
  - **ii. REVOKE:** Take away privileges from the user.
  - **iii. COMMENTS:** Add a comment to the data dictionary.

❖ Comparison of file management System with DBMS:

TRADITIONAL FILE:

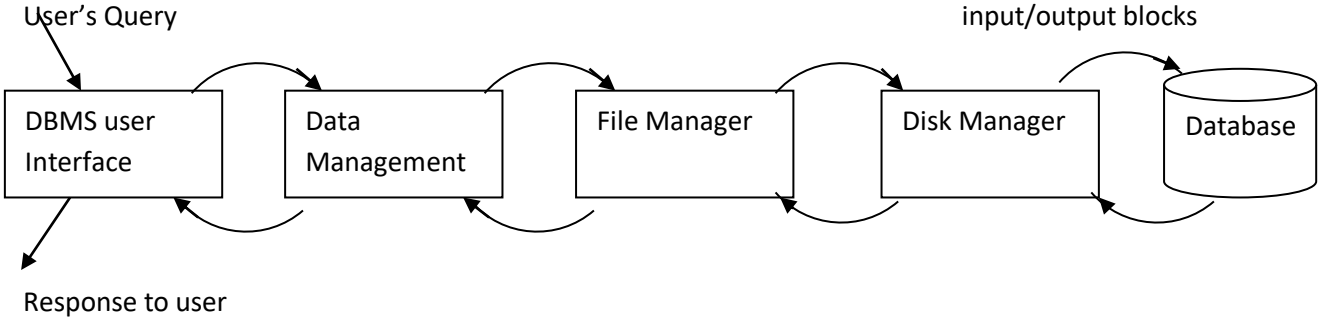
i. Small system
ii. Relatively chip
iii. Few 'files'
iv. Files are files
v. Simple structure
vi. Redundant data
vii. Change of inconsistency
viii. Isolated data
ix. Single user.
x. No security

DATABASE MANAGEMENT SYSTEM:

i. Large system
ii. Relatively expensive
iii. Many 'files'
iv. Files are table
v. Complex structure
vi. Reduced redundancy
vii. Consistency
viii. Data can be shared
ix. Multiple user
x. Security

❖ Procedure for database access:

- Any access to the stored data is done by the data manager.
- The steps involved in database access can be summarized as in fig.



- A user's request for data is received by the data manger, which determines the physical record required.
- The data manager sends the request for a specific physical record to the file manager.
- The file manager decides which physical blocks of secondary storage devices contains the required record and sends the request for the appropriate block the disk manager.
- A block is a unit of physical input/output operation between primary/secondary storage.
- Disk manager retrieves the block and sends it to the file manager, which sends required record to the data manager.

❖ Type of data system:

The DBMS can be classified according to the No. of users and the database site locations. There are

a) On the basis of No. of users.
b) On the basis of site selection.

a) On the basis of No. of users:

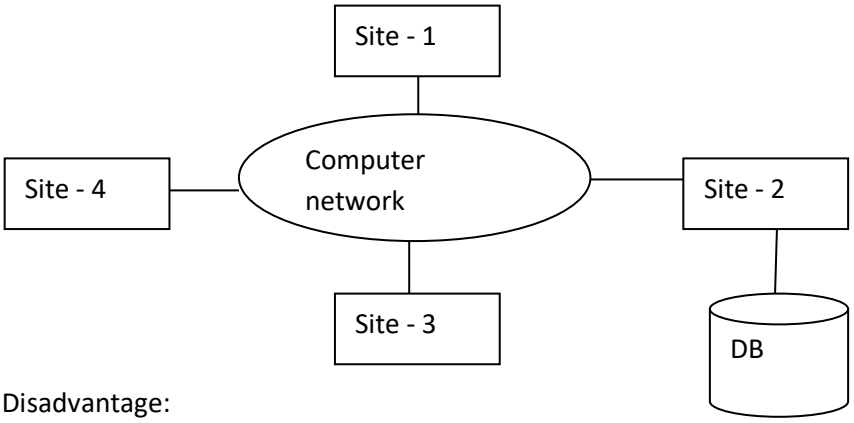
I. Single user:
In single user system, the database resides on one computer and is only accessed by one user at a time. This one user may design, maintain and write database programs.

II. Multi-user DBMS:
In this system, the data are bath integrated and shared.
A database is integrated when the same is not recorded in two places.

b) On the basis of site selections:

i) Centralized DBMS:

- It consists of a single processor together with its associated data storages devices and other peripherals.
- It is physically confined to a single location.
- Data can be accessed from the multiple sites with the use of computer network while the database is maintained at the central site.



- Disadvantage:
  - Communication cost from terminal to the central site can be expensive.
  - When the central site computer or database system goes down, then user is blocked from using the system units system comes block.

ii) Parallel database system:

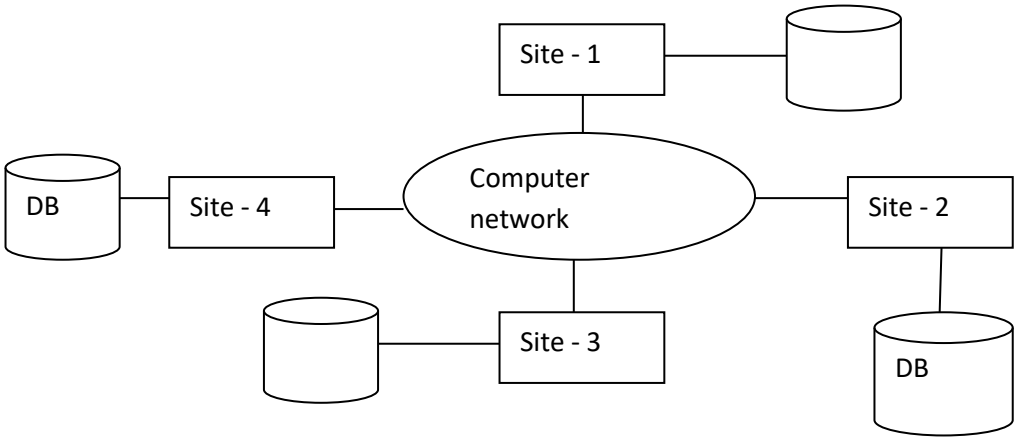
- It consists of a multiple CPU data storage disk in parallel.
- It improves processing and input/output speeds.
- Advantage:
  - Parallel database systems are very useful for the applications that have to query extremely large database.
  - In a parallel database system, the through put and the response time are very high.
- Disadvantage:
  - It affects speed up of processing time.
  - Slowdown the execution process.

iii) **Distributed database system:**

- A logically interrelated collection of shared data physically distributed over a compiler network is called a distributed database.
- The software system that permits the management of the distributed database and makes the distributed transparent user's called as distributed DBMS.
- Advantages:
  - A single database (server) can be shared across several client system (application).
  - It provides local autonomy
  - It provides greater efficiency and better performance.

- Disadvantages:

  Recovery form failure is more complex.



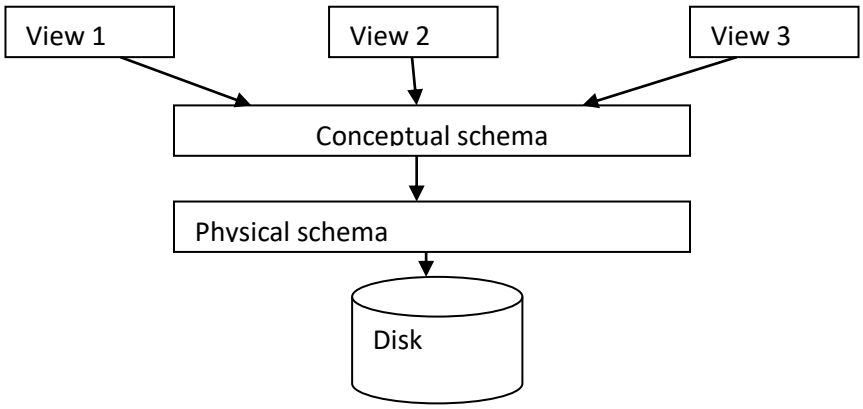
iv) **Client/server DBMS:**

- Client/server architecture of database has two logical components namely client and server.
- Application and tools of DBMS run on one or more client platforms, while DBMS software resides on the server.
- The server computer is caused back end and the clients computer is called front end.
- The server and client computers are connected into a network.
- Advantages:
  - It has less expansive platforms to support application.
  - It is more flexible.
  - Response time and through put is high.
  - A single database (server) can be shared across several distinct clients.
- Disadvantages:
  - Programming cost is high, particularly in initial phase.
  - Block of management tools for diagnosis performance monitoring and security.

Data abstraction:

- The purpose of database system is to provide the users with an abstract view of data.
- The system sides certain details of how data are stored and maintained.
- They are 3 different level of abstraction they are
  a) Physical level of abstraction.
  b) Logical level of abstraction.
  c) View level of abstraction.
- Data is described in 3 levels of abstraction as below figure.

View level



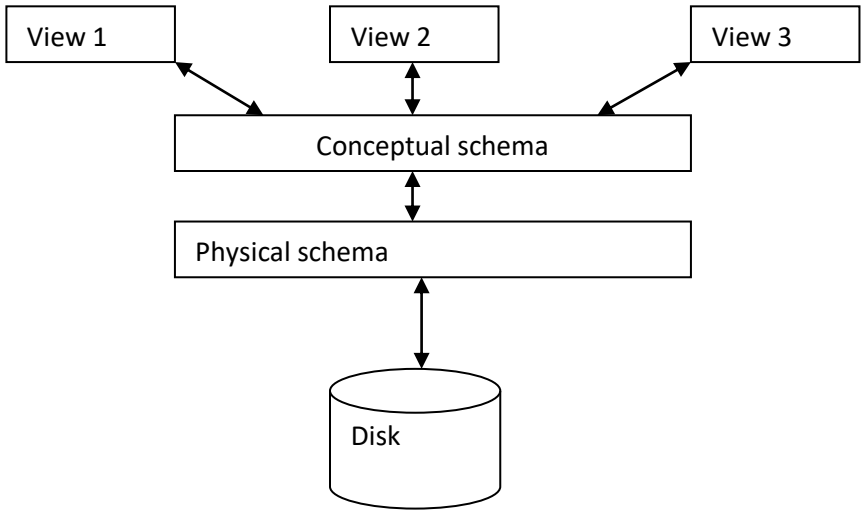
[Database levels]

- **Physical schema:**
  - This is a very low level representation of database.
  - It describes how data are physically stored in a database.
  - It also caused as internal view of database.
- **Conceptual schema:**

  It describes what data are stored are in a database and what kind of relation exists between them. It is also caused logical or conceptual view of database.
- **External schema:**
  - This is the highest level of abstraction, describes only a part of an entire database.
  - It is also caused view level or external view of database.
  - The objective is to simplify the interaction user with database system.

❖ Three level Architecture/ Three –fire architecture for DBMS:

The architecture for DBMS is divided into three levels.



[Three level architecture]

1. **External level or view level (Highest level):**

- ❖ The user's view of the database i.e. the way individual user sees the data.
- ❖ Consists of a No. Of different external user of the database.
- ❖ Describe part of the database for a particular group of user's part of database from certain users.
- ❖ Different user may have different representations of the same data.
- ❖ Ex: one user may view dates in the form (day, month, year). While another may view data as (year, month, day).

2. **Conceptual level or logical level (middle level):**

It is concerned with the following events.

- ❖ Logical structure of the entire database as seen by DBA.
- ❖ What data is stored in the database?
- ❖ The relationship among the data.
- ❖ An abstract representation of the entire information inside of the database.
- ❖ It also contains authorization and validation procedures.

3. **Internal level or storage level (lower level):**

It is concerned with the following events.

- ❖ Physical representation of the database on the computer.
- ❖ Flow the data is stored in the database.
- ❖ Storage space allocation for data and indexes.
- ❖ Record placement.
- ❖ Data compression encryption.
- ❖ Mapping:
  - o DBMS is responsible for mapping between three types of schema.
  - o Two mapping are required in a database system. They are

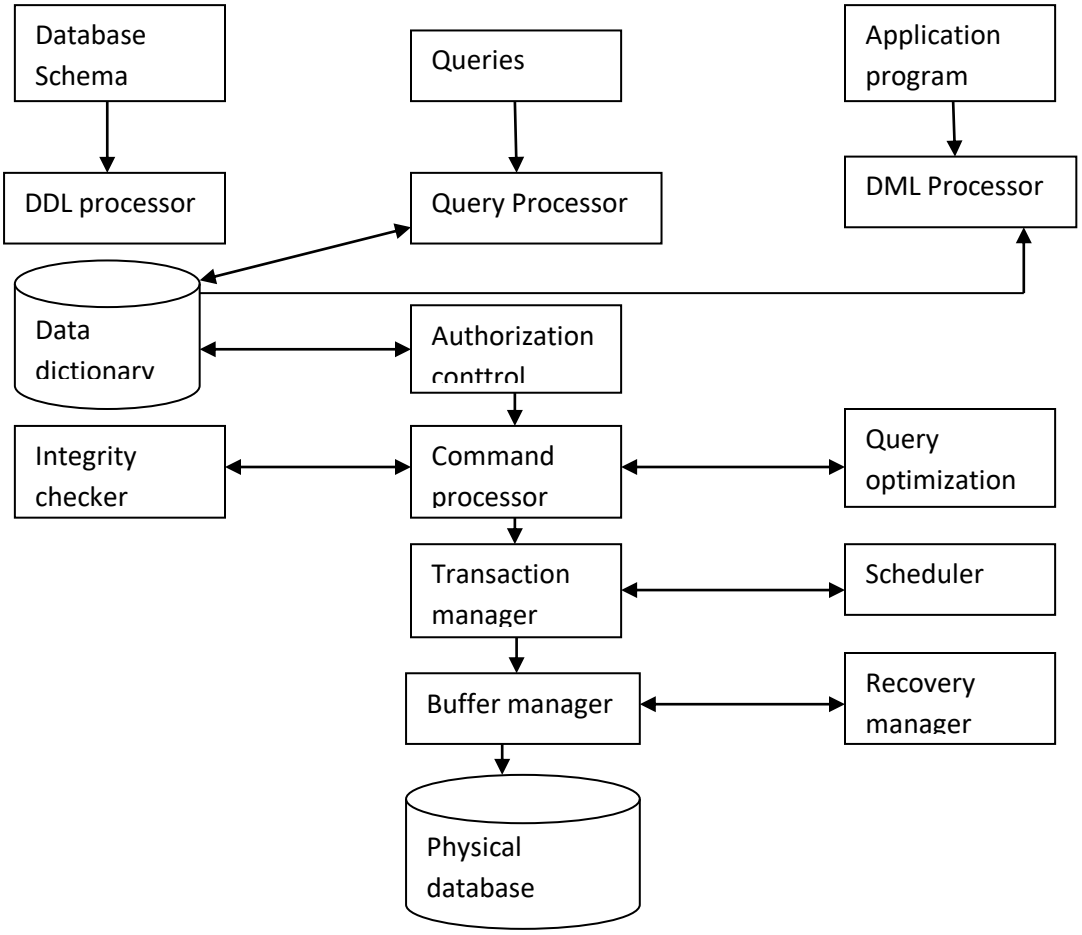
A. **External conceptual mapping:**

- ❖ Each external schema is related to conceptual schema by external or conceptual mapping.
- ❖ A mapping between external/ conceptual views gives the correspondence among records and relationships of the external and conceptual.
- ❖ External view is an abstraction of the conceptual view, which in the turn is an abstraction of internal view.
- ❖ There is a mapping from a particular logical record in the external view to one or more conceptual records in the conceptual view.

B. **Conceptual/internal mapping:**

- ❖ Mapping between the conceptual and internal level specifies the method of deriving the conceptual record from the physical database.
- ❖ It enables DBMS to find the actual record in physical storage that constitute a logical record in conceptual schema.
- ❖ DBMS accepts SQL commands generated form a verify of user interfaces, produces query evaluation plans, execute these plan against the database and returns the answer.

❖ Structure and components of DBMS:



Main components of DBMS are as follows:

i. **Query processor:**

- It transforms user queries into a series of low level instruction.
- It used to interpret the online user's query and convert it into series of operations in a form capable of being sent to the run time data manager for execution

ii. **Run time Database manager:**

- It is the central software component of the DBMS, which interfaces with user submitted application programs and queries.
- It handles database access at runtime.
- It converts user's query from user's logical view to the physical file system.
- It enforces constraints to maintain the consistency and integrity of the data, as well as its security.
- It also performs backup and recovery operations.

It is also referred as database control system and has the following components

a) Authorization control:
   This module checks the authorization of user's in terms of various privileges to users.
b) Command processor:
   It processes the queries passed by authorization control module.
c) Integrity checks:
   It checks the integrity constraints so that only valid data can be entered into the database.
d) Query optimizer:
   It determines an optional strategy for the query execution.
e) Schedule:
   It provides an environment in which multiple users can work a same piece of data at the same time i.e. it supports concurrency.
f) Transaction manager:
   It ensures that the transactions properties should be maintained by the system.
g) Data manager:
   It is responsible for actual bonding of data in the database. It provides recovery to the system in case of failure through recovery manager. It is also responsible for transfer of data between main memory and secondary storage which is done through buffer manager.

Data models:

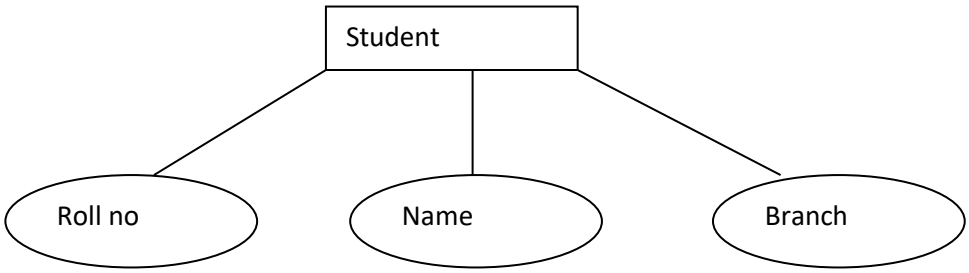
- Organizing the structure of a database is the data model.
- A model is a representation of reality, real world, objects and events, their associations.
- Data model can be defined as "an integrated collection of concepts for describing and manipulating data; relationship between data and constraints on the data in an organization".
- Data model can be grouped into two types
  1. Physical data model.
  2. Logical data model.

Physical data model:

- It describes how data is stored in the computer, representing information such as record structures, record ordering and access path.
- The most common physical data model are
  - Unifying model
  - Frame memory model

Logical data model:

- ❖ It focuses on the logical nature of the data representation.
- ❖ It concerned with what has represented in the database.
- ➢ **Object based logical model:**
  - It uses concepts such as entities, attributes and relationship.
  - An entity is a distinct object (person, place, concept, and event) in the organization that is to the represented in the database.
  - An attributes is a property that describes some aspect of object.
  - A relationship is an association between entities.
    - **Entity relationship model:**
      - ♦ Peter chen first introduced E-R data model in 1976.
      - ♦ E-R model normally represented by E-R diagram.
      - ♦ It is a graphical representation of entities and their relationship in a database structure.
      - ♦ For the database designer, the utility of E-R model is
        - ✓ It maps well to the relational model. The constructs used in ER model can easily be transformed into relational tables
        - ✓ It is simple and easy to understand with a minimum of training so, it is easy to communicate the design to the end user.
        - ✓ It can be used as a design plan to implement in specific database management software
        - ✓ Example of E-R diagram is as in fig



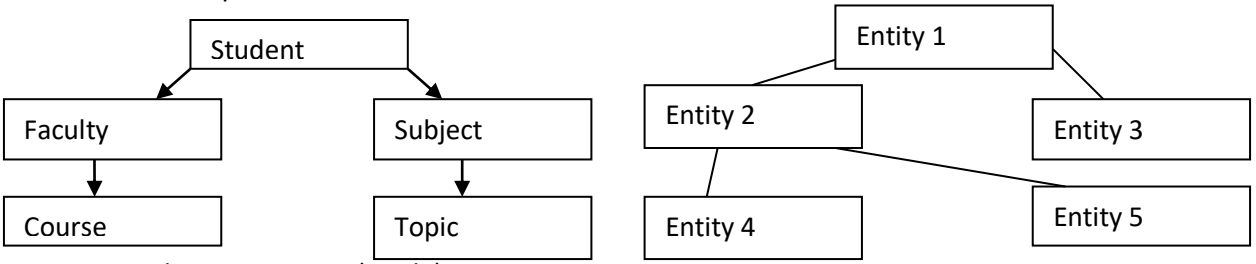
♦ Advantage:
- ✓ Straight forward relation representation
- ✓ Easy conversion for E-R to other data model.
- ✓ Graphical representation for better understanding.

♦ Disadvantage:
- ✓ Popular for high- level database design.
- ✓ There is no industry standard notation for developing E-R diagram.

- **Object – oriented model:**
  - ♦ It represents on entity as a class.
  - ♦ It is based on collection of objects.
  - ♦ An object which container same types of values and methods are grouped together into class.
  - ♦ An objects contains body of codes that operate an object, these body of code is called methods.
  - ♦ An object contains values stored in instance variables within objects.
  - ♦ Advantages:
    - ✓ Capacity to bundle large No. of different data types.
    - ✓ Data access to maintain.
    - ✓ Object – oriented features improve productivity.
  - ♦ Disadvantage:
    - ✓ Difficult to maintain.
    - ✓ Not suitable for all application.

Record based logical model:

- ❖ It is used to specify the overall logical structure of the database.
- ❖ It is so named because the database is structure in fixed format records of several types.
- ❖ Each record types defines you fixed No. of fields, or attributes and each field is usually of a fixed length.
- ❖ The three most widely accepted record based data models are Hierarchical, Network, Relational
  - ➢ **Hierarchical model:**
    - This model is like a structure of a tree with the records forming the nodes and fields forming the branches of the tree.
    - The different element (example- records) present in the hierarchical tree structure have parent – child relationship.
    - A parent element has many parent elements.
    - In this model, database is represented by records and links to establish the relation between different records.
    - The structure of this model books like a hierarchy as in fig.
    - Example:



Level 0 : root parent (mode)

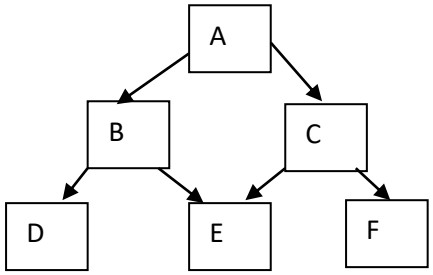
Level 1 : root children (segment)

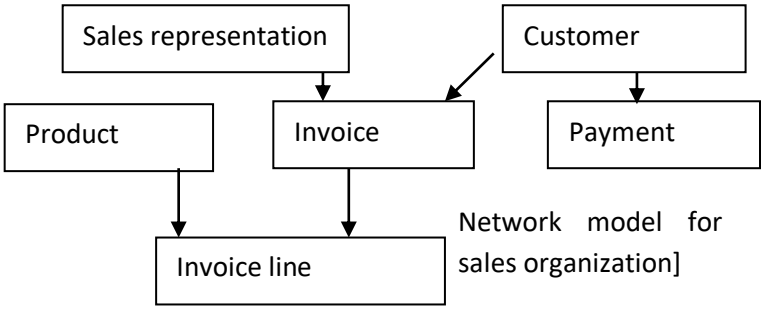
Level 2: segment (level 1 children hierarchical model)

- Advantage:
  - The design of a hierarchical database is simple.
  - It offers data security.
  - The child segments always automatically referred by its parents, so this model promotes data integrity.
  - It is very efficient.
- Disadvantage:
  - It is quite complex to implement.
  - Marinating the database and the applications can become very difficulty.
  - Complexity in programming.
  - It is not suitable for all the cases.

➢ **Network data model:**

- It is similar to hierarchical model except that a record can have multiple parents.
- It has 3 basic components such as record type, data items and links, here a relationships is a set, composed of at least two record types.
  - First record type is called owner record i.e. parent.
  - Second record type is called a member records i.e. child.
- The connection between owner and its member records is identified by a link called set – name, which is used to retrieved and manipulate data.
- It facilitative N: M (many to many) relationship.
- A set represent one- to -many (1: M) relationship between owner and the member.





Network model for sales organization]

[Network data model]

Level 0: owner

Level 1: owner/member

Level 3: member

- Advantages:
  - Simple and easy to design.
  - Facilitating more relationship types.
  - Provide database integrity.
  - It is based on universal standards formulated by DBTG.
- Disadvantages:
  - System complexity.
  - Not a user – friendly design and is highly skill – oriented system.

- It doesn't provide structural independence.

➢ **Relational data model:**

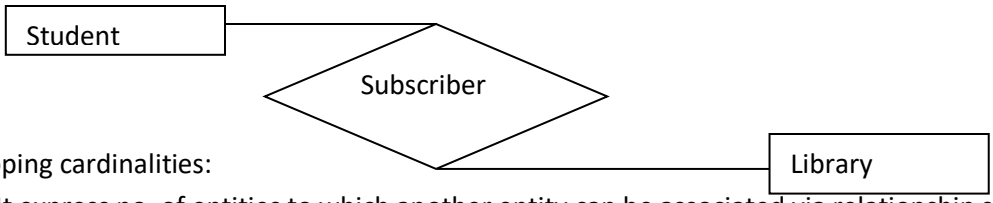
- It is implemented using relational database management system (RDBMS).
- Data is represented by using tables, consists of rows and columns.
- Each and every column header is called as attributes, every row is called tuples, is stores information of an entity and a table is called relation.
- The No. of attributes is called degree or arity of a relation and the No. of tuples is known as cardinality of relations.
- It was first introduced by E.F.Code of IBmin 1960. It laid 12 rules for relational database.
- A DBMS that employee's relational model (12 rules) is called RDBMS.
- Ex:

  Student……..->Entity

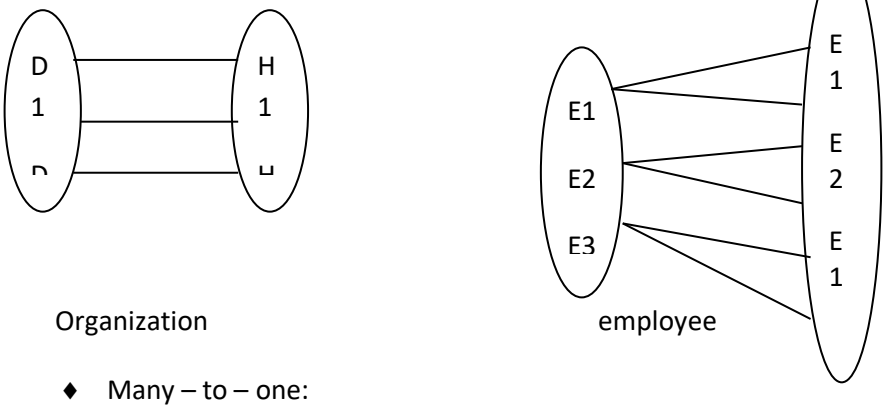
Roll no	Name	Branch	Semester
1001	Sona	C.S.E	4^{th}
1002	Amit	C.S.E	2^{nd}
1003	Vivek	I.T.	3^{rd}
1004	Sachin	C.S.E	4^{th}

- Advantages:
  - It is structural independence.
  - Database design, implementation, maintenance and usage are much easier.
  - It provides powerful, flexible and easy to use query facility.
- Disadvantages:
  - Need powerful computing hardware and data storage devices.
  - Easy – to – design capability leading to poor design and results into slower system, degraded performance and data corruption.
- Basic terminology used in E – R model:
  - Entity:
    Any real world object is called as entity, which has some meaning and some properly.
  - Entity set:
    It is set of entities of some kind that share the same properties is called entity set.
  - Weak entity:
    Entities which are dependent is called weak entity set.
  - Strong entity set:
    Independent entities are called strong entity set.
  - Attributes:
    The properties of an entity is called an attributes each entity must have a value a for each of its attributes for each attributes, these is a set of permitted value called domain.
- Attributes are classified into following types:
  - Simple attributes:
    It is composed of single component with an independent existence. It is also called as atomic attributes.
    Ex: Sex, Age, Salary

- Composite attributes:
  It is composed of multiple components, each with an independent existence. It is further divided into smaller component.
  Ex: Address, can be composed of state, city etc.
- Single – valued attributed:
  It holds a single valued for a single entity.
  Ex: Room No, Roll No, etc
- Multi – valued attributed:
  It holds multiple values for a single entity.
  Ex: Hobby, e- mail, etc
- Derived attributes:
  It represents a value that is derivable form the value of other related attributes.
  Ex: Age, net salary etc.
- Null attributes:
  The attribute whose value is null is called null attributes.
- Relationship:
  It is an association among different entities.
  - ♦ A relationship set is a set of relationship of some kind.
  - ♦ The associated between entity set it referred to as participation.
  - ♦ The No. of entity sets that participate in a relationship set is called as degree of the relationship set.
  - ♦ Ex:
    Student and library entity denote the relationship subscribed by as in fig.

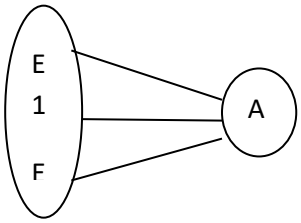


- Mapping cardinalities:
  - It express no. of entities to which another entity can be associated via relationship set.
  - The mapping cardinality must be one of the following
    - ♦ One – to – one:
      In this relationship, one instance of entity is related to another instance of the entity.
      Ex: One department can have only one employment as its HOD and one employee can becomes HOD of only one department.
      [Department HOD one – to – one relationship]
    - ♦ One – to – many:
      In this relationship, one instance of an entity is related to multiple instance of another entity.
      Ex: One organization can have many employees but one employee can work only for one organization.



Organization employee

- Many – to – one:
  This is reverse of one – to – many relations i.e. an entity is associated with at must one entity.
  Ex: Many employees work in one organization

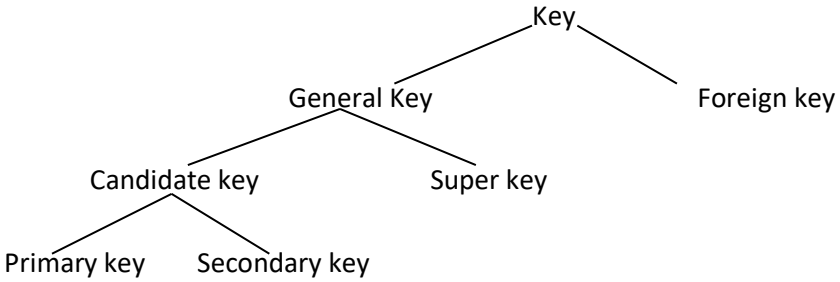


[Employee organization many – to - one]

- Many – to – many:
  Here, multiple instance of one entity is related to multiple instance of another entity.
  Ex: One Student in roll for many courses and one course in enroll for many courses and one course enroll for many Students.

- **Keys:**

The key is a field which is used to distinguish each and every entity in the entity set.

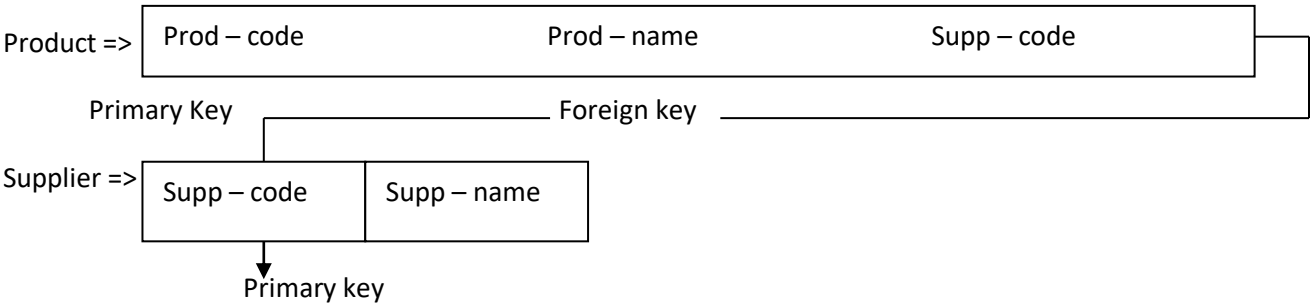


- Super Key:
  It is a set of one or more attributes, that taken collectively to distinguish each and every entity in an entity set.
  Ex: For an entity set employees, set of attributes (em - name, address) can be considered to be a super key.

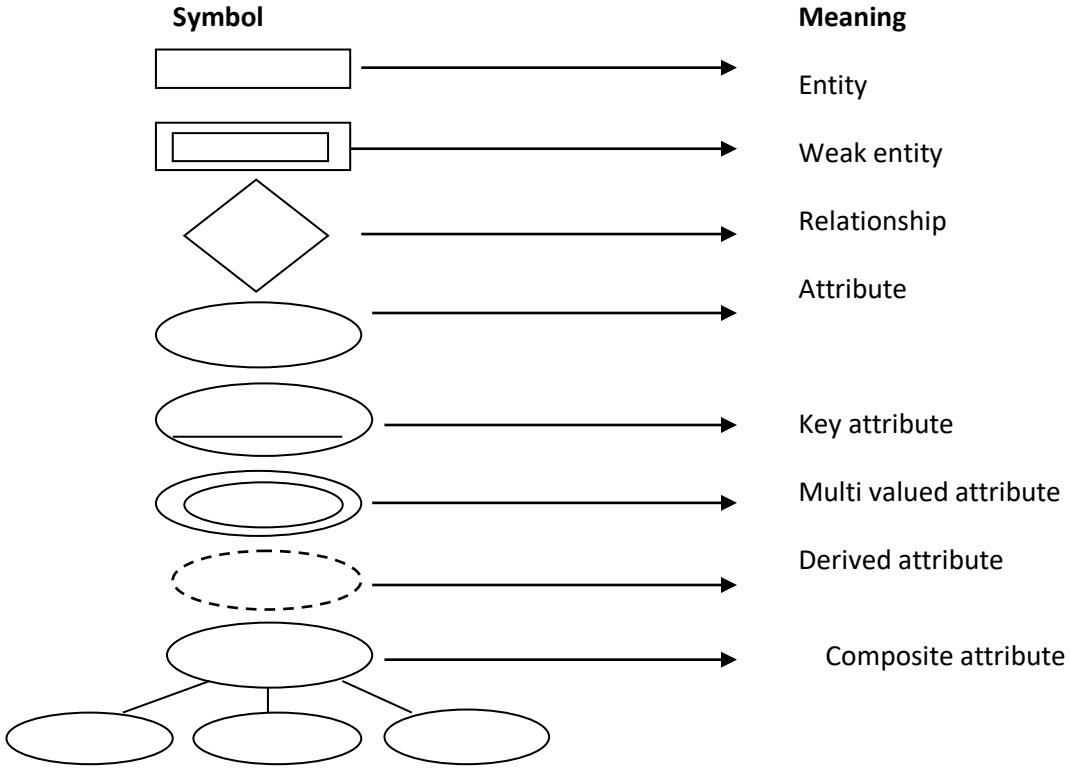
- Candidate Key:
  It is an attribute are set of attributes that uniquely identifies a record.
- Primary Key:
  It uniquely identifies each record. In a table and must never be the same for 2 records.

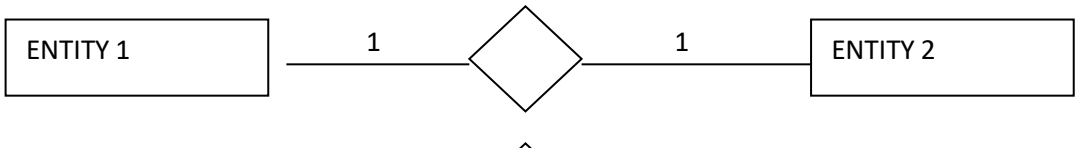
Ex: emp – code cab be primary key for the entity set employee.

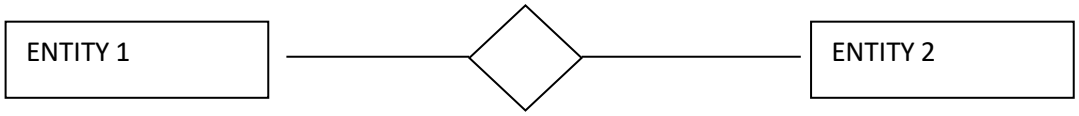
- Alternate/ Secondary Key:
  One of candidate key of table is chosen to be a primary key. Then the remaining candidate keys are called as alternate key.
- Foreign Key:
  The key which is the primary key is another table is called foreign key. It is also referred as reference key.



E – R diagram Notation:







1 4

M 1

M M

Step in E – R model:

Step 1: identify the entities.

Step 2: find the relation among different entities.

Step 3: identify the key attributes for every entity.

Step 4: identify the other relevant attributes.

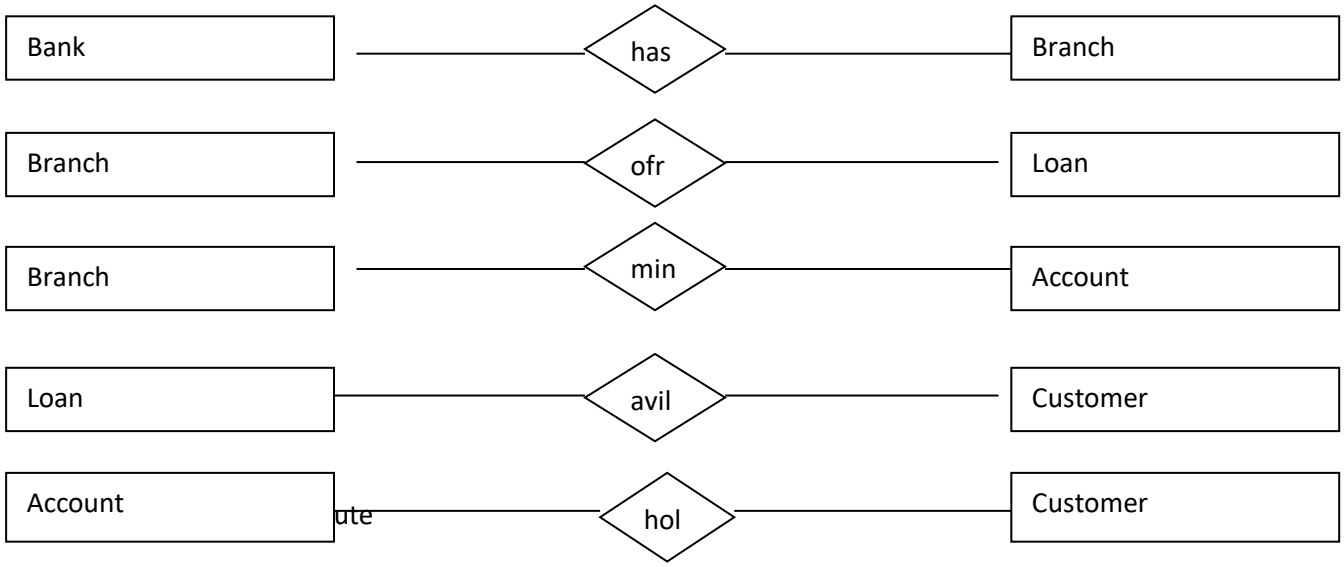
Step 5: complete E – R model by using all above steps.

Example: Banking System

Solution: **Step 1**: identify the entities

- Bank Branch Loan
- Account Customer

Step 2: Find relationship



Bank -> bank code

Branch-> branch number

Customer -> customer number

Loan -> loan number

Account -> account number

Step 4: Identify relevant attributes

Bank -> name, address

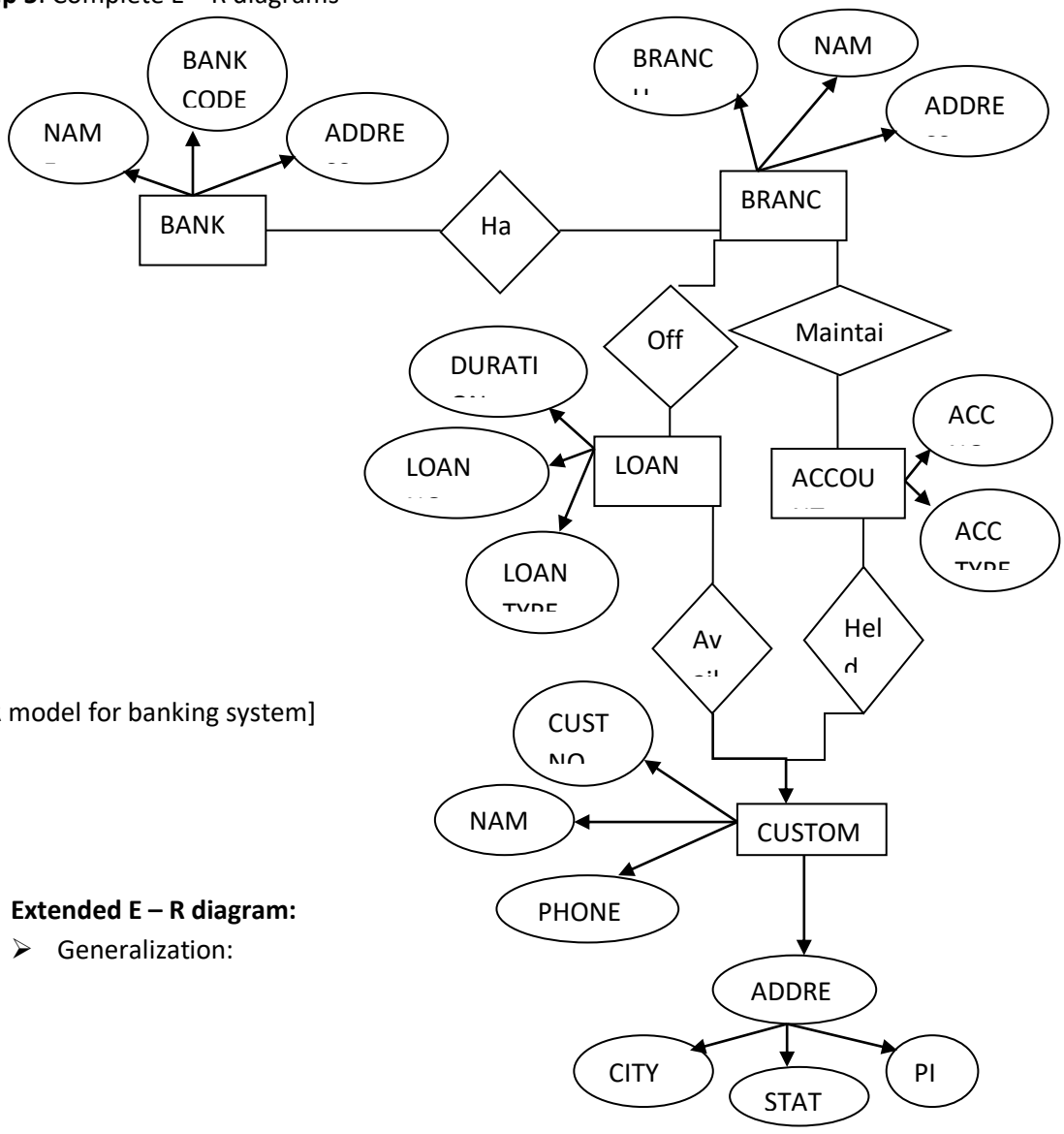
Branch -> name, address

Customer -> name, phone, address

Loan -> loan type, duration

Account -> account type, declaration.

Step 5: Complete E – R diagrams



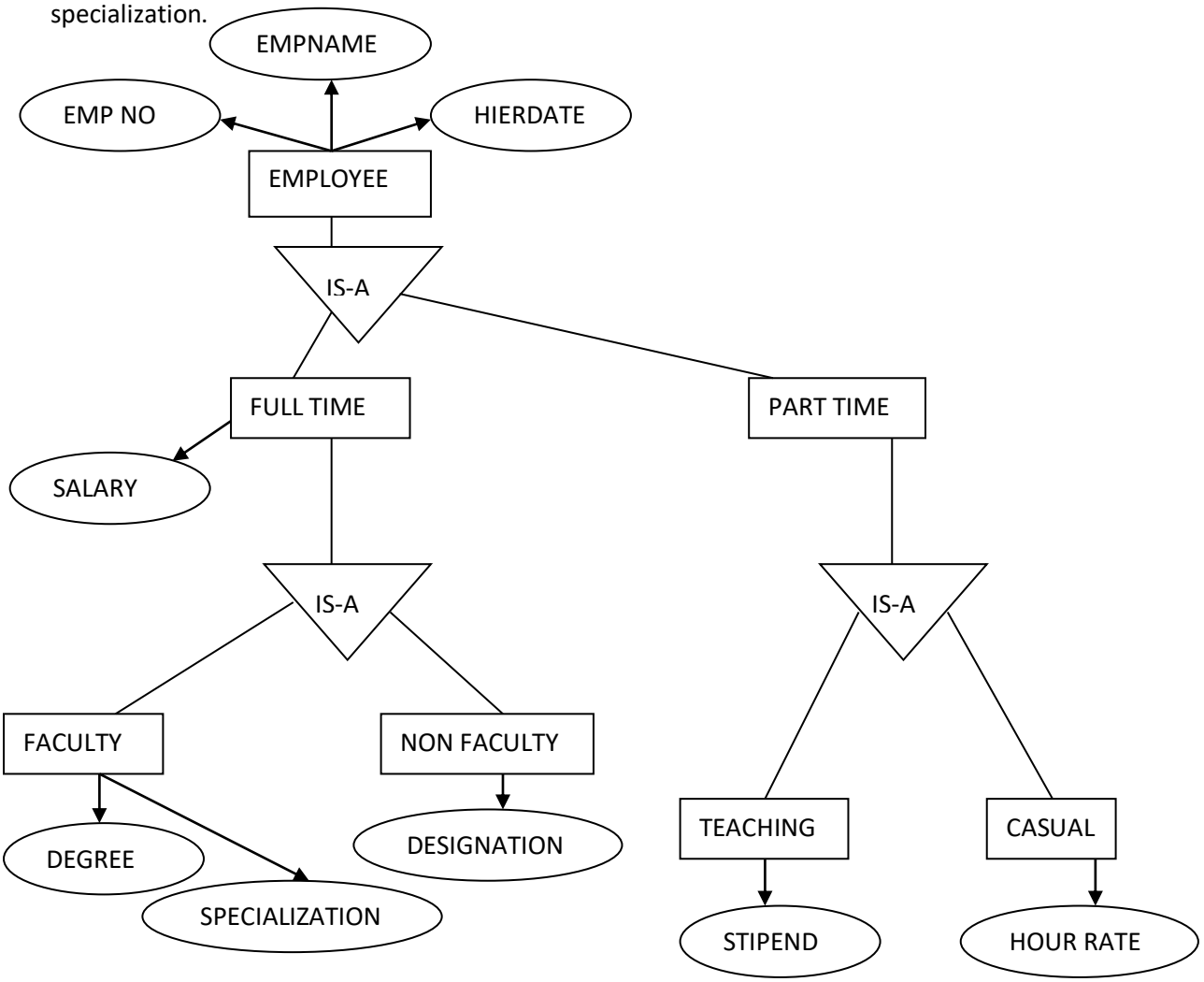
[ER model for banking system]

- **Extended E – R diagram:**
  - Generalization:

- A set of two or more entities that share common attributes can be generalized into a higher level entity type called super – type or generic entity.
- It emphasizes the similarities among lower level entity sets and to hide difference.
- It is denoted through a triangle component labeled “ISA”.
- It is bottom – up approach.

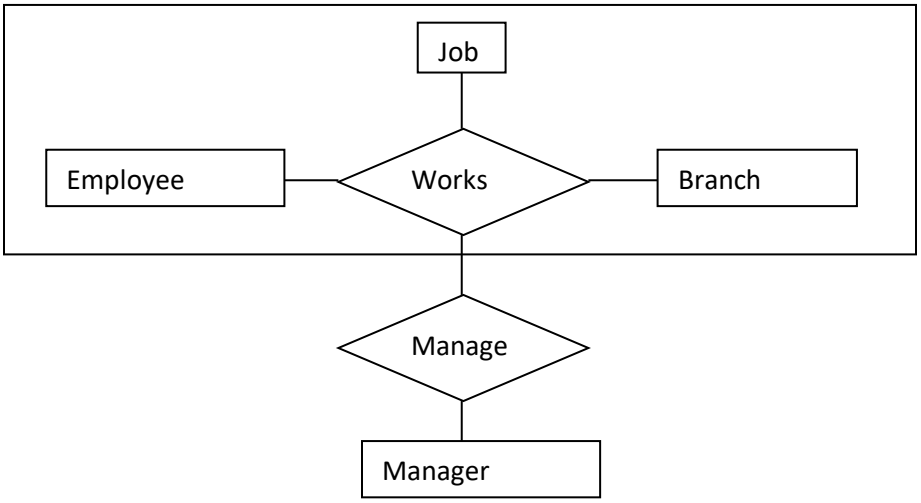
➢ Specialization:

- It is process of taking subset of higher level entity set to form lower level entity set.
- It emphasizes the differences among entities in the super class.
- It is top – down approach.
- It defines a set of sub classes of an entity type, which called as super class of the specialization.



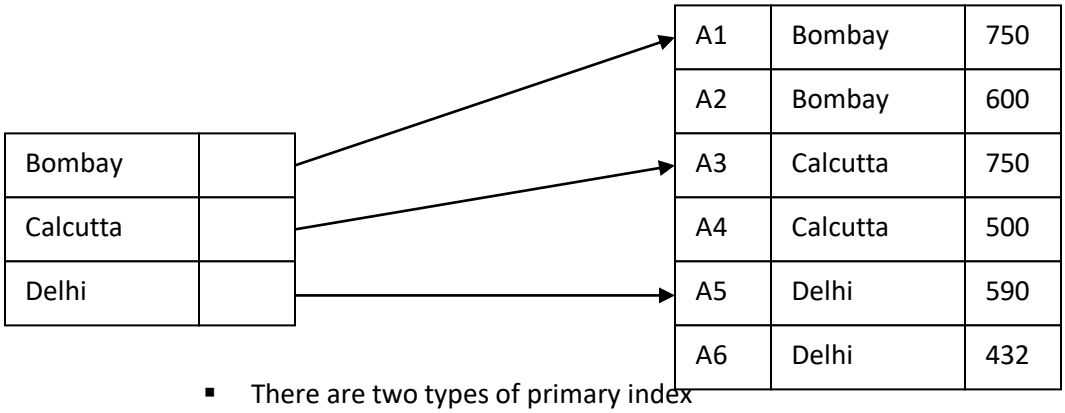
➢ Aggregation:

- It is the process of compiling information on an object, there by abstracting a higher level object.
- It is denoted by a rectangular box by grouping together entity.



- **Integrity constant:**
  - All the data entering into the database must be validation for its correctness.
  - It is implemented in RDBMS through 3 – type of integrity checks.
  - There are described below
    - Domain integrity:
      It involves the implementation of business domain specific rules.
    - Entity integrity:
      It is implemented using the primary key constant. It refers the physical entity. Attributes uniquely identifies the physical entity.
    - Referential integrity:
      - It is implemented using foreign key constraint.
      - It includes cascading update and delete which ensures in the primary table.
- Referential integrity enforces the following 3 rules
  - We may not add a record to employee table unless the dept – no attribute points to a valid record in the dept – table.
  - If a primary key for a record in the dept – table changes, all corresponding records in the employee table must be modified using cascading update.
  - If a record in the dept table is deleted all corresponding records in the employee table must be deleted using a cascading delete.
- Database audit:
  - It is a lay – out of all changes to the database along with information such as which user performs the changes and when the change was performed.
  - It is used for the security purpose in several ways.
- Indexing
  - An index is the way by which the user can easily retrieve the data from the database.
  - It is a sorted list of keyword and to access records in a file.

- An index contains a pair of consisting of index value and list of pointers to disk block for the records that have the same index value.
- An index file is very small compared to a data file.
- Also index entries are ordered.
- In case an index file is very large we can create a multi- level index i.e. index on index.
- There are two basic kinds of indexes.
  - Ordered index:
    - ♦ It is based on sorted ordering of the values.
    - ♦ It is used to gain first random access to the records in a file.
    - ♦ Each index structure is associated with a particular search key.
    - ♦ An ordered index stores the values of the search key in sorted order and associates each search key.
    - ♦ Basically there are two type of ordered index are used
      - ➢ Primary index
        - The index on a primary key is called a primary index.
        - The files with a primary key index on the search key are called as index sequential file.
        - It has two columns:
          The ordering key field and a block address for that key field in the data file.



Bombay	
Calcutta	
Delhi	

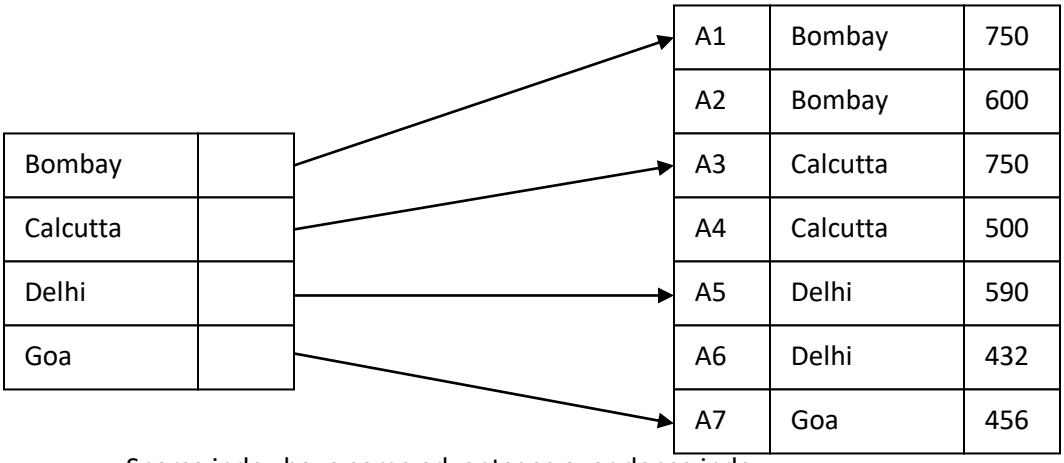
A1	Bombay	750
A2	Bombay	600
A3	Calcutta	750
A4	Calcutta	500
A5	Delhi	590
A6	Delhi	432

        - There are two types of primary index
          - Dense index
          - Sparse index

✓ Dense index:
  - An index record appears for every search key values in the file.
  - In a dense primary 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.

✓ Sparse index:
  - An index record appears for only some of the search key values are called as sparse index.

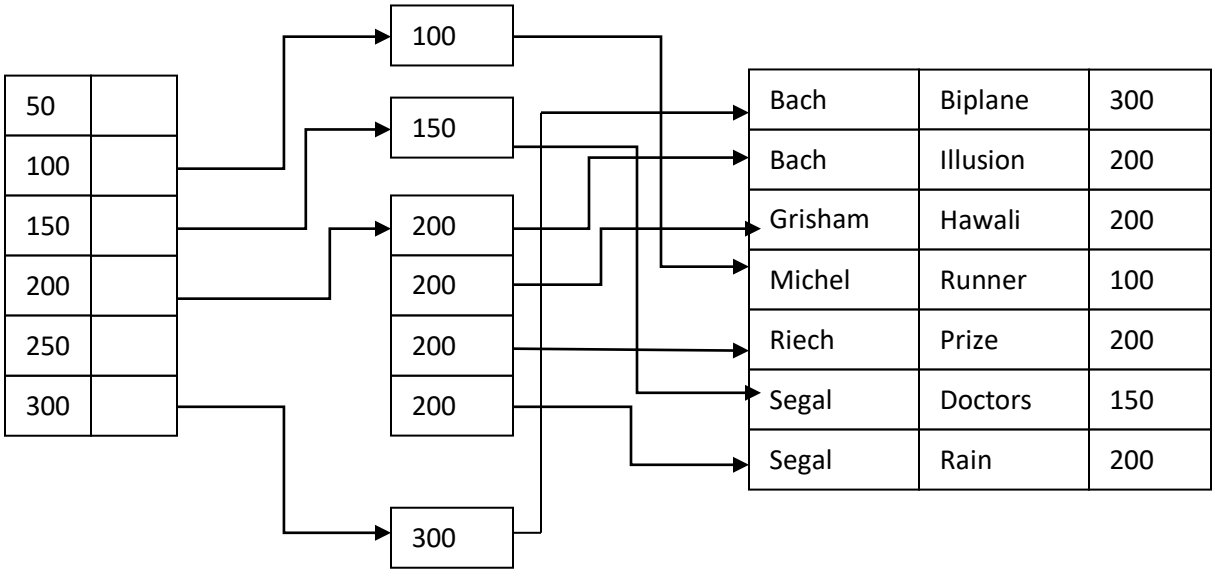
- Here, the searching will start for particular records pointed by that index entry, and follow the pointers in the file until we find the desired record.



- Sparse index have some advantages over dense index
  - ❖ They required less space.
  - ❖ They impose less maintenance overhead for insertions and deletions.

➢ Secondary index

- A secondary index is a file that contains records contains a secondary index field value which is not the ordering field of the data file and a pointer to the block that contains the data record.
- It can be defined on an alternate key or non key attributes.
- Although a data file can have only one primary index it can have many secondary indexes.



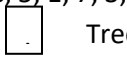
- Consider the book catalogs a secondary index on price will have price pointed by successive values in the index are not ordered sequentially.
- The secondary index must contain pointers to all records on the file.

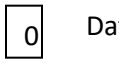
- Hashed index:
  It is based on the values being uniformly distributed using a mathematical function called hash function.

- There are several techniques for implementing ordered indexing and hashing each should be evaluated 1st before choosing one.
- Evolution should be based on the following factors:
  - Access type:
    It could include finding records with a specific attribute value or fall in a specified range.
  - Access time:
    It is the time taken to find a particular data item or a set of item using given technique.
  - Insertion time:
    It is time taken to insert a new record.
  - Deletion time:
    This is the time taken to delete a record.
  - Space overhead:
    It means the addition of space occupied by the index structure.

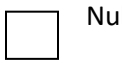
❖ **B – tree indexes:**

- A B^+ tree index takes the form of a balanced tree in which every path from the root to the tree leaf is of the same length.
- Each non – leaf node in the tree has between 'n/2' and 'n' children (n/2 = C = n), where 'n' is fixed for a particular tree.
- It creates performance overhead on insertion and deletion and adds space overhead.
- It is a multilevel index.
- The minimum value that a leaf can contain is (n-1)/2.
- The range of values in each leaf does not overlap.
- The length of every path from the roof to a leaf node is the same.
- Ex:
  Insertion in B^+ tree:
  8, 5, 1, 7, 3, 12, 9, 6, of order 3

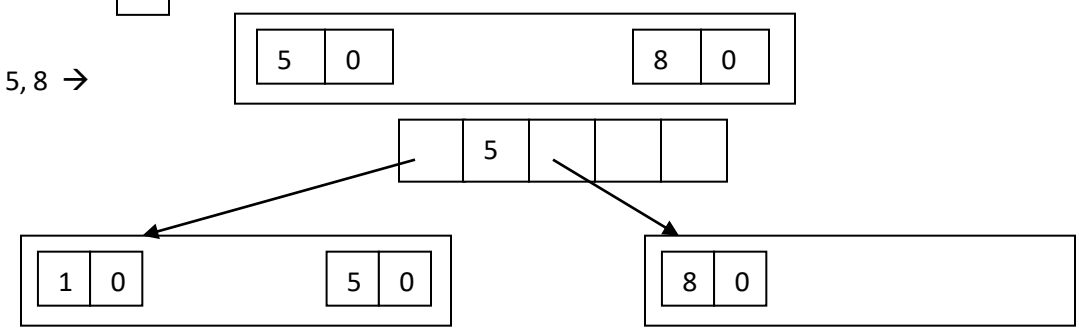

Tree node pointer.



Data pointer

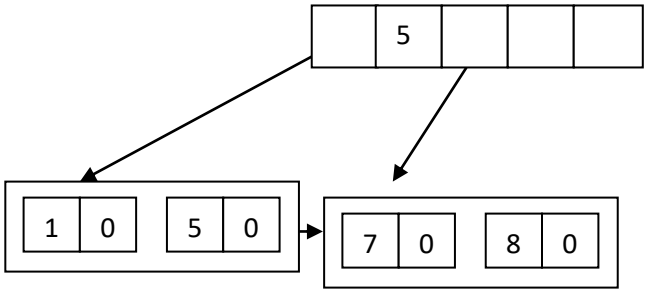

Null tree pointer

5, 8 →

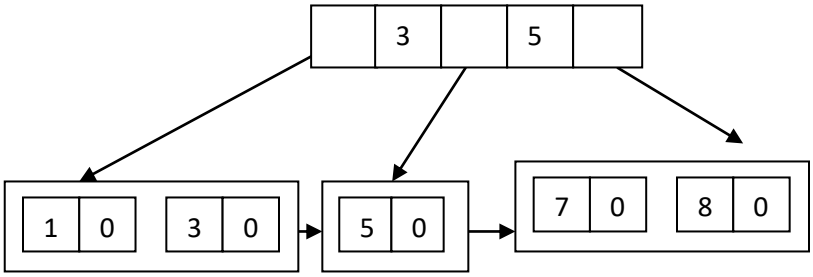


1→

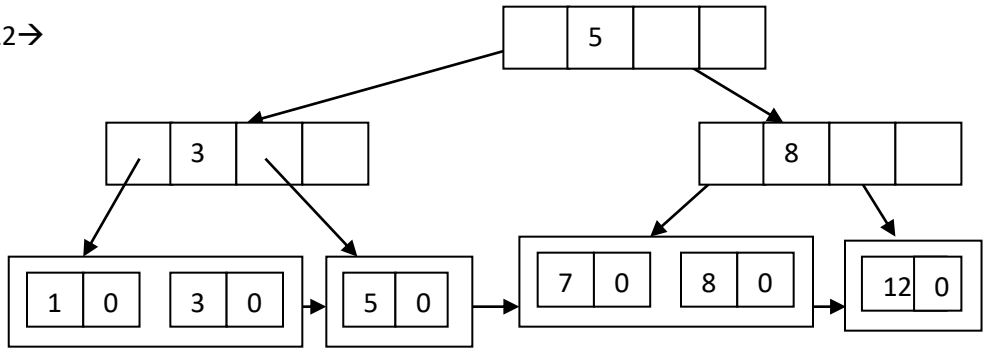
7→


5
1 0
5 0
7 0
8 0

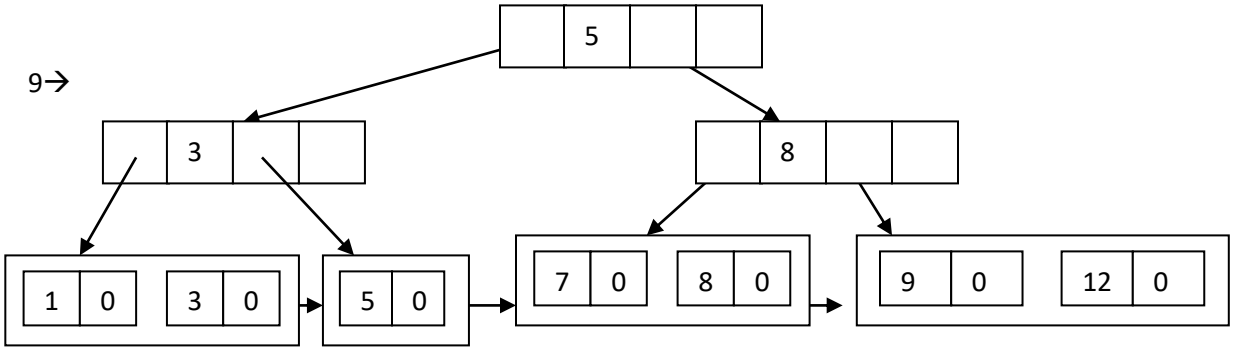
3→


3
5
1 0
3 0
5 0
7 0
8 0

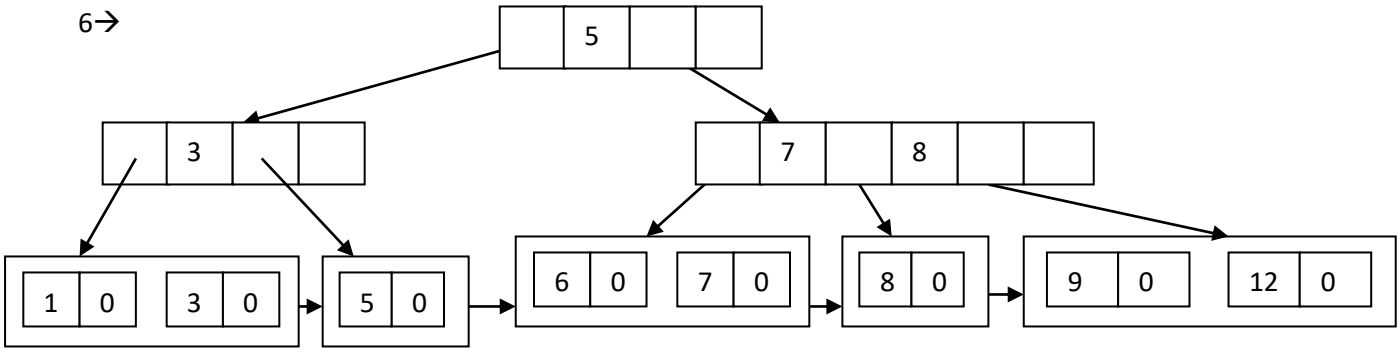
12→


5
3
8
1 0
3 0
5 0
7 0
8 0
12 0

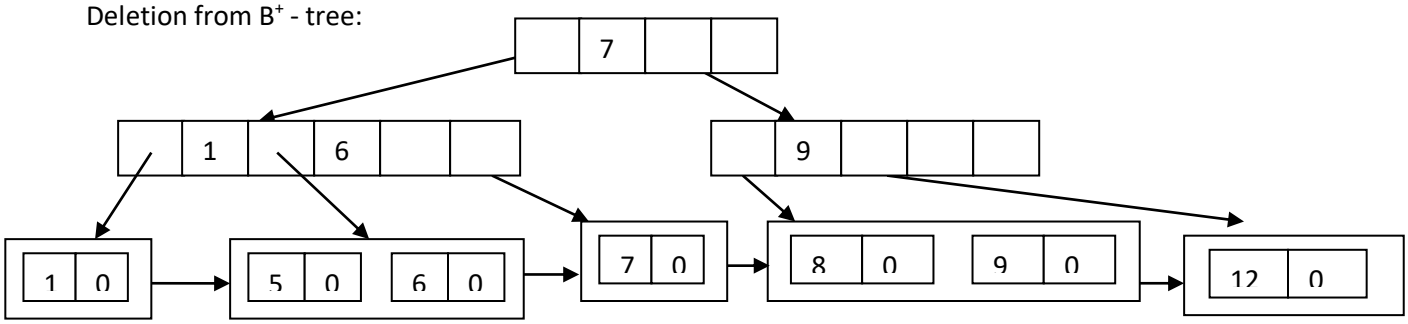
9→


5
3
8
1 0
3 0
5 0
7 0
8 0
9 0
12 0

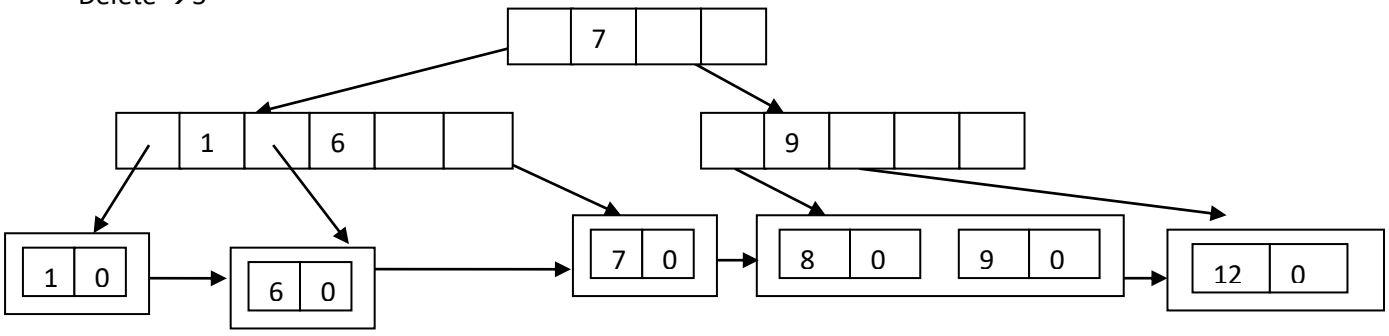
6→



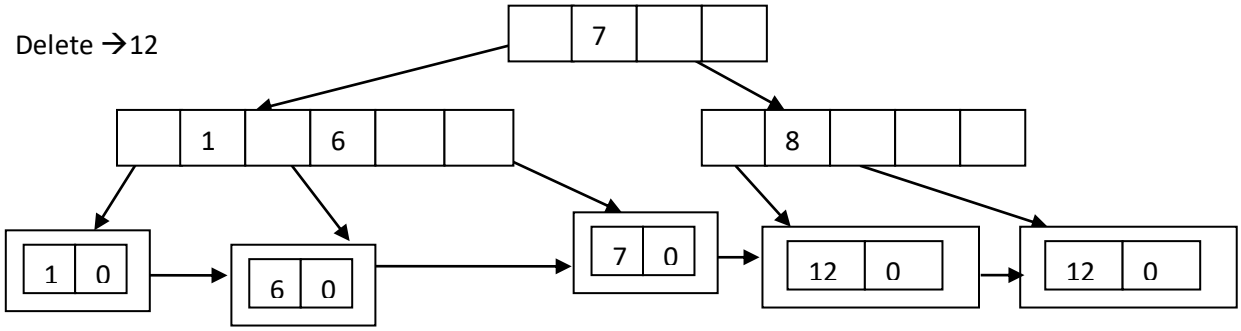
Deletion from B^+ - tree:

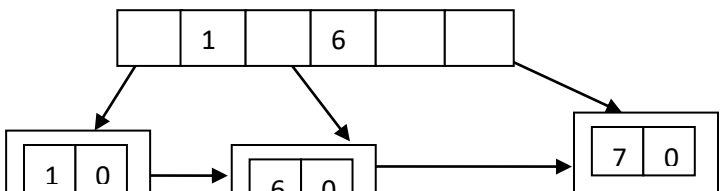


Delete →5



Delete →12





Delete → 9

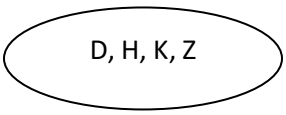
- **B – Tree:**
  - It is known as balanced sort tree which is useful for external sorting.
  - The height of the tree must be kept to a minimum.
  - The leaves of the tree must all be on the same level.
  - There must be no empty sub trees above the leaves of the tree.
  - The search time is $0(\log_2 n)$
  - There is no redundant storage of search key value because B – Tree stores each search key value in node.
- **Advantage:**
  - It is good for dynamic problem because there is no overflow.
  - As data is retrieved via index it is always presented in order.
  - There is predictable access time from any retrieval.
  - It is completely balanced.
- **Disadvantage:**
  - It is not appropriate for every small table.
  - The index can use considerable disk space.

Example:

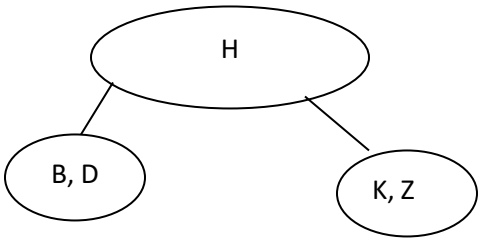
Insertion in B – Tree:

{D, H, K, Z, B, P, Q, E, A, S, W, T, C, L, N, Y, M} or order 5 order 5 means 5-1=4 I.e. insert at most 4 keys at the leaf node.

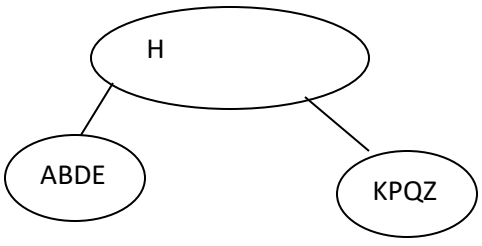
D, H, K, Z →



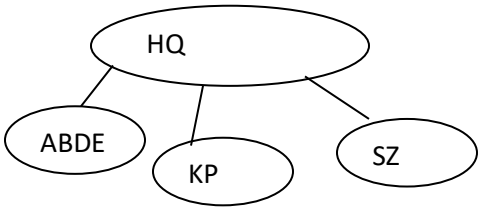
B →



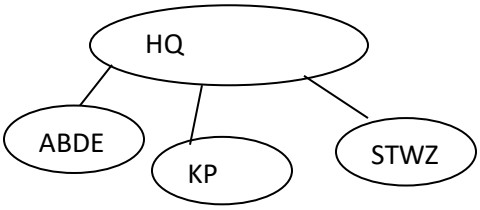
P, Q, E, A →



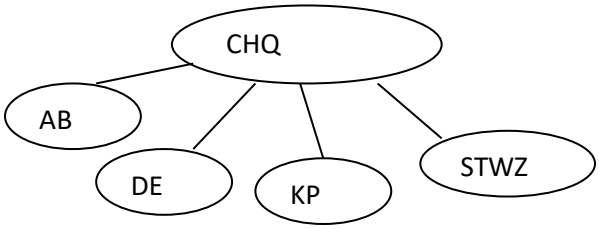
S →


HQ
ABDE
KP
SZ

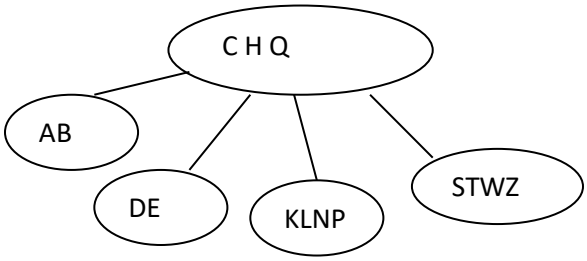
WT →


HQ
ABDE
KP
STWZ

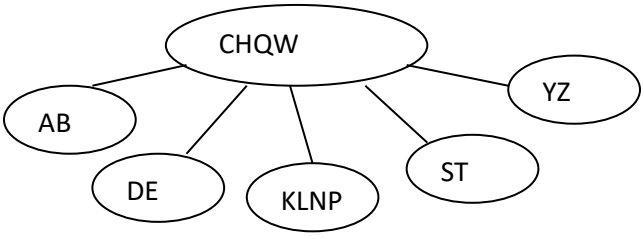
C →


CHQ
AB
DE
KP
STWZ

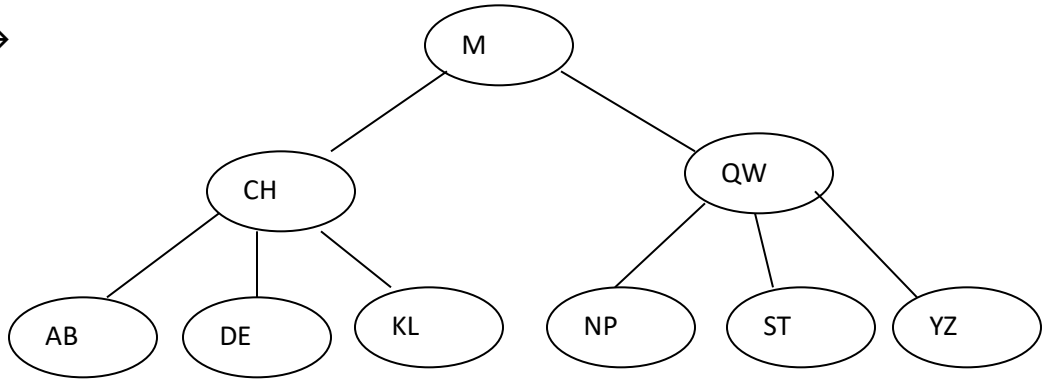
L, N →


C H Q
AB
DE
KLNP
STWZ

Y →


CHQW
YZ
AB
ST
DE
KLNP

M →


M
CH
QW
AB
DE
KL
NP
ST
YZ

- **Hashing:**
  - In a hash file, records are not stored sequentially in a file instead a hash function is used to calculate the address of page in which the record is to be stored.
  - The field on which hash function is calculated is called as hash field and if that field acts as the key of the relation then it is called as flash key.
  - Records are randomly distributed in the file so it also called as Random or Directed files.
  - Commonly some arithmetic function is applied to the hash field so that records will be evenly distributed throughout the file.
  - Advantage:
    When tuples are retrieve based on an exact match on the hash field value, particularly if the access order is random.
  - Disadvantage:
    When tuples are retrieved based on a range of value for the hash field. For example, retrieve all students whose name begins with the 'R'.

Query Language **Module - 2**

A query language is a language through which we communicate the database. When we want to read or write something into from existing database, then use query language.

❖ Relational calculus:

- In relational calculus, a query is specified in a single statement without specified any order or method for retrieving query result.
- Here, user has to specify what is required and need no to specify how to obtain it.
- **Tuple relational calculus (TRS):**
  - It is a non – procedural query language describes the desired information without giving a specific procedure.
  - TRC is based on specifying a no. of tuple variable.
  - The expression of tuples calculus re constructed form following elements.
    - Tuple variation:
      Each tuple variable is constrained to range over some named relation
    - Condition:
      Condition of the form x * y
      Where * is any one of relational operators =, <, >, <=, >= and x and y are the expression.
    - Well – formed formulas (WFFS):
      - WFFS is the sequence of symbols, which can be ensured by stating some rules for the construction of WFFS.
      - WFF is constructed from conditions, Boolean operators (AND, OR, NOT) and quantifiers ($\exists$,) according to below rules.
      - F1: Every condition is a WFF.
      - F2: If f is a WFF, then so are (f) and not (f).
      - F3: If F and q are WFFS, then so are (f and g) and (f or g).
      - F4: If F is WFF in which T occurs as a free variable, then $\exists$T(f) and T(f) are WFFS.
      - F5: Nothing else is a WFF.
      - A simple TRC query is of the form.
      - {t|COND (t)}
        Where t is a tuple variable and COND (t) is a conditional expression involving t. the result is the set of all tuple t that specify COND (t).
        Example:
        Consider the following relations
        Employee (emp –code, empname, designation, mgr – no, dab, sal, comm, dept - code)
        Department (D – code, D – name, Location)

Find all employees where salary is above Rs. 6000.

{t|employee(t) and t, salary>6000}

For retrieve some attributes, we can write

{t. emp – name, t.salary|employee(t) and t.salary>6000}

❖ Domain Relational calculus (DRC)

- Domain variables range over domains rather than relations.
- Expressions of the domain calculus are constructed form the following elements.
  - Domain Variable:
    Each domain variable s is constrained to range over some specified domain.
  - Conditions:
    Conditions can take two forms
    - X*Y, where x and y domain variables.
    - Membership conditions of the form x(term, term,..) here, x is a relation and each "term" is a pair A;r where A is an attribute of x and r is either a domain variable or a constant.
  - Well formed formulas(WFFS)
    WFFS formed in accordance with rules F1- F5
    All expression in DRC is expressed as
    {<x1, x2, x3, …., xn>|p(x1, x2, x3, …, xn)}
    Where <x1, x2, x3, …., xn> is a domain variable and (x1, x2, x3, …, xn) is formula.
- Example:
  - Find employee name and salary, where salary is above 6000
    {<n, s>|<n, s> Ɛ employee s>6000}
  - Branch:

B – Name	B – City	Asset

  - Customer:

C – Name	C – Street	C – City

  - Loan:

L – No	B – Name	Amount

  - Deposition:

C – Name	Acc – No

  - Borrower:

C – Name	L – No

i. Find out B – name, L – No and amount for loans over $1200.
Ans: TRC: {t|t∈ loan ^ t(amount)> $1200}

ii. Find out loan no. from each loan of an amounts $ 1200.
Ans: TRC: {t| ∃ s∈ loan(t[L - No] =S[L – No]^s[amount])>$1200}

DRC :{<l>| ∃ b, a(<l, b, a>)∈ loan, a>$ 1200}

iii. Find out name of all customers who have a loan from Berhampur branch.

Ans: TRC: {t| ∃ s ∈ Borrower t[C – name]=s[C – name]^ ∃ U ∈ loan(U[L – No]=S[L – No]^ U[B – Name]="Berhampur")}

DRC: {<a>| ∃ |<l,e>∈ borrower ^ ∃ b, a<l, b, a> ∈ loan ^ b="Berhampur"}

iv. Find out all customers who have an amount at the bank but have a loan at the bank.

Ans: TRC: { t| ∃ u ∈ Depositer(t[C – name]=u[C – name]^ ∃ s ∈ Borrower (t[C – Name]=S[C – Name])}

❖ Relational algebra:

- Relational algebra is a procedural query language.
- It consists of set of operations that take one or two relations as an input and produce a new relation as their result.
  - **Unary:**
    The operations which are carried out by taking one relation is known as unary operation.
    - **Selection (⌐):**
      - It selects tuples (rows) that satisfy a given condition.
      - It extracts specific tuples form a relation. It is also known as restriction operation.
        It is denoted by lower case Greek latter sigma (⌐).
        R= ⌐B(p)
        Where
        R is resultant relation.
        B is the condition.
        P is the table name.

Roll No	NAME	BRANCH	SEMESTER	ADD	AGE
2400	Rajiv	I T	4th	BBSR	23
2401	Dibya	C SE	4th	BAM	22
2402	Sherya	C SE	3rd	BBSR	24
2403	Sneha	IT	3rd	BAM	21

i. Find out those tuples of student relations where the branches are IT.
⌐ Branch = 'I.T'(Student)

ii. Find all tuples in which age of student is more then 21
⌐age>21(Student)

iii. Find all tuples where address is BBSR and age>23.
⌐address="BBSR"^ age>23(student)

- **Projection (π):**

It is a uninary operation because it operator on only one relation.
It extracts attributes (columns) from a relation

It denoted by upper case Greak latter Pi(**π**).

The general form of projections is

π

<attribute list>(R)
Where π is the projection

R is the relation
And <attribute list> is list of attributes of relation.

Example

i. Find out rollno and name of student

π rollno, name(student)

ii. find out customer name and bank balance amount from deposit relate.

π

C- name, balance(deposit)

- **Rename:**
  - ♦ it is denoted by lower case Greek letter (ρ)
  - ♦ it is used to assign a new name to the given relation.

Example:

A student relarion such as student(Roll No, name, branch, address)
We want to findout name and address of student and assign a new name as stud - addrs (name, address)

Stud – addr(name, address)(ρ name, address(student))

- **composition of selection and projection:**

1. find out rollno and name of student wrere branch is CSE.

π

rollno, name (σ branch="CSE"(Student))

2. find out roll and name from student where branch is IT and age >22.

π

roll, name(σ branch = "IT" ^ age > 22(Student))

- **Binary:**

The operation which is operating on two or three relation is known as binary relation.

- Set operation:
  - ♦ Union(U):

    It is a binary operation which takes two relation as on input and produces a resultant relation as an output.
    It is denoted by the symbol u.
    Duplicated tuples would be eliminated.

    Example:
    XUY which includes all tuples that is either in X or in Y in with X and Y.
  - ♦ Intersection(∩):

    It is a binary operation which takes two relations as an input and produces a resultant relation as an output.
    It is denoted by the symbol ∩ .
    It allows to findout, the common tuples from with table.

    Example: X∩ Y which includes the common tuples in X and Y relation.
  - ♦ Difference (-):

    It is a binary operation which takes two relations as an input and produces a resultant relation as an output.
    It is denoted by join symbol (-).
    It allows to find tuples that are in one relations but not in another relation.

    Example: X-Y which includes tuples in x but not in Y.

Example: X

Emp no	E name
1	Soumya
2	San
4	Nitin

Y

Emp no	E name
2	San
5	Sai
3	Om

xuy

Emp no	E name
2	San
5	Sai
3	Om
4	Nitin
5	Sai

x∩y

Emp no	E name
2	San

x−y

Emp no	E name
1	Soumya
4	Nitin

- **Join operation:**
  - ♦ It is a binary operation which takes two relations as an input and produces a resultant relation as an output.
  - ♦ It s denoted by join symbol (lXl)
  - ♦ It is used while retrieving the information from multiple tables.
  - ♦ The join operation is of different tuples there are
    - ➢ **Natural join:**
      - ▪ It allows to combine certain selection and cartesion product into one relation.
      - ▪ It is denoted by join lXl symbol.
        Example:

Marriage ID	Name	Age
M1	Rohan	28
M2	Ritik	38
M3	Rahul	25

Husband

Marriage ID	Name	Age
M1	Disha	23
M2	Nisha	35
M4	Mona	22

Wife

Find out husband name and there is wife name

π
husband name, wife name (husband lXl wife)

Name	Name
Rohan	Disha
Ritik	Nisha
Rahul	Mona

- Outer join:
  - It is used to include the missing values of the relation.
  - There are 3 types of outer join
    1. Left outer join (-lXl):
       It is used to include the missing values left table
       It is denoted by (-lXl)

Marriage ID	Name	Age	Marriage ID	Name	Age
M1	Rohan	28	M1	Disha	23
M2	Ritik	38	M2	Nisha	35
M3	Rahul	25	M4	Mona	22

[The result of ⌐ (husband-lXl wife])

2. Right outer join (lXl-):
   It is used to include the missing value from right table.
   It is denoted by (lXl-) symbol;

Marriage ID	Name	Age	Marriage ID	Name	Age
M1	Rohan	28	M1	Disha	23
M2	Ritik	38	M2	Nisha	35
M3	Rahul	25	M4	Mona	22

[The result of ⌐ (husband lXl- wife)]

3. Full outer join(-lXl-)
   It is used to include the missing values from both the relation.
   It is denoted by (-lXl-) symbol.

Marriage ID	Name	Age	Marriage ID	Name	Age
M1	Rohan	28	M1	Disha	23
M2	Ritik	38	M2	Nisha	35
M3	Rahul	25	M4	Mona	22

[the result of ⌐ (husband -|X|-wife)]

- **Cartesion product (X):**
  It is also known as cross product denoted by X.
  It returns all fields X followed by all fields of y.
  Example:
  Suppose two relations X and Y are.

Emp no	E – Name
1	Soumya
2	San
4	Nitin

X:

PROJECT
A10
A11

Y:

xXy:

Marriage ID	Name	Age
1	Soumya	A10
1	Soumya	A11
2	San	A10
2	San	A11
4	Nitin	A10
4	Nitin	A11

- **Division (÷)**
  It is denoted by the symbol (÷).
  It is useful for a special kind of query that includes the phrase “for all”.
  Example:

Account:

B – name
Acc – no
Balance

Branch:

B – name
B – city
Assets

Customer:

C – name
C – city
C – state

Depositor:

C – name
Acc – no

All branches located in BAM

$R_1 = \pi$
B - name (σ b – city ="BAM"(Branch))

Find customer name and branch R_2

$R_1 = \pi$
C – name, B – name(Depositor |X| account)

Name for customer has an account at a branch so find customer appear in R_2 with every branch name in R_1.

R_1

B – Name
SBI
ANDRA

R_2

C - Name	E – Name
San	SBI
San	ANDRA
Sai	IOB
Ansu	INO
Subham	URBAN
Disha	URBAN

$R_1 \div R_2$

C - NAME
San

Functional dependency:

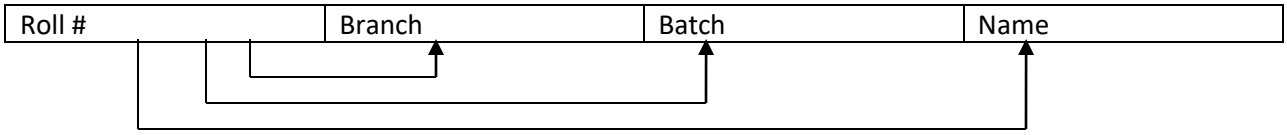
Consider a relation R that has two attribute A,B. attribute B of relation R is functionally depends on attributes A, if and only if the value of attribute A uniquely determines the value of B.

And if there were several tuples that had the same value of A then all these tuples will have an identical value of attribute B.

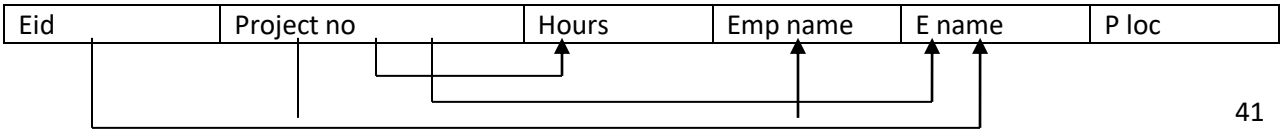
If t_1,t_2 are two tuples in relation R and if t_1 [A] = t_2[A] then we must have t_1[B]=t_2[B] then, R A→ R B.

That means A functionally derives B or B is functionally derived from A and which can be represented by A→ B.

Example: 1 student



2 emp project



EID → EMP – NAME

PROJECT →{p - name, p – loc}

{EID, PROJECT – NO} → HOUR

[**FUNCTION OF DEPENDENCY**]

Full functional dependency:

For a given relation R, A and B are attributes which is fully functionally dependent an attribute A only if it is not functionally dependent on subset of A. However A is composite in nature.

Example:

Student (Regd no, subcode, subname, faculty name, marks, r - no)

Here, combination of Regd no and subcode uniquely distinguishes each and every tuples in student relation. It means Regd no. subcode together defines exactly one value of marks. This can be represented as

Redg no subcode → marks

In this case, mark can be determined either by Regd No or subcode. It can be determine only using Regd no and subcode. Hence marks are fully functionally dependent on Regdno subcode.

- **Partial dependency:**
  - For a given relation R, A and B are attributes. B is partially dependent on A only if it is dependency on subset of attribute A.
  - In the above student relation, subname, faculty name, r – no, can be partially dependent on Regdno, subcode attribute because only subcode is sufficient to derive subname, faculty name, r- no.
- **Transitive dependency:**
  - In the given relation R, A, B and C are attributes.
  - The dependency is called as transitive if C depends on B and in turn B depends on A.
  - Example:
    In above student relation.
    R- no depends on faculty name and faculty name depends on subcode. So, R- no transitively dependes on subcode.
    It can be represented as
    If subcode →faculty name → R – no, then
    Subcode→ R- no

Armstrong's Axion:

Inference rules for functional dependencies:

Rule -1:

Reflexive rule.

Consider a relation R having attributes X and Y.

If x ⊇y, then x→y.

Rule – 2
Augmentation rule.
Suppose a relation R having attributes x, y and z
If x→ y, the xz→ yx.
Rule -3
Transitive rule.
Suppose a relation R having attributes x, y and z .
If {x→y, y→z}, then x→z
Rule – 4
Decomposition rule
Let a relation R having attributes x, y and z.
If x→yz, then x→ y and x→z
Rule – 5
Union rule
Let a relation R having attributes x, y and z.
If {x→ yz, x→ }, then x→yz
Rule -6
Pseudo transitive rule
Let a relation R having attribute x, y, z and w
If {x→y, wy→z} then wx→z
Rule -7
Composition rule
Let a relation R having attributes x, y, w and z.
If x→y and w→z, then xw→yz

Closure:

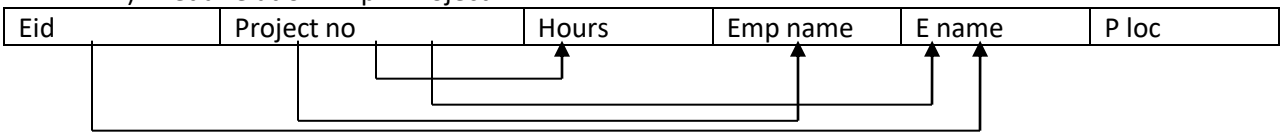
- Let a relation R having some functional dependencies denoted by F. then the closure of F is denoted by F+.
- F+ is a set of all FDS including F and can be logically derived from F by using Armstrong's Axioms.
- The closure is needed to find out one or more candidate keys of the relation.

Example:

1) Let R=(A, B, C, D, E)and set of F of FD are
   F= A→ BC
   CD→ E
   B→D
   E→A then, the closure set F+ of FD
   F+=A→B [BY UNION RULE]
   A→C
   CD→A [SINCE CD→E, E-A BY TRANSTIVE RULE]
   E→BC [since E→A, A→BC by transitive rule]
   BC→ E [since B→D and CD→E by pseudo transitive rule]

- The L.H.S terms of F and F+ of FDs are super key.
- The terms of super keys are cut-out which R.H.S of F and F+ of FDs are and remaining term is called candidate key.
  Super key = {A, CD, B, E, BC}
  R.H.S terms are = {A, B, C, D, E, BC}
  There for candidate key = {CD}

2) Let a relation Emp – Project.



F={Eid → Ename p- no→ pname, ploc}

p- no, Eid→ hours}

the closure sets are

$\{Eid\}^+ = \{Ename, Eid\}$

$\{P\text{ -no}\}^+ = \{P – no, Pname, Ploc\}$

$\{P – no, Eid\}^+ = \{Eid, Ename, P\text{- no}, Pname, Ploc\}$

Decomposition:

- A relation with redundancy can be refine or replacing it with smaller relation that contain some information but without redundancy. This process is known as decomposition.
- There are two properties of decomposition there are
  - Loss Less join Decompositions:
    - Loss less decomposition means there is no less of information from the relation which is decomposed.
    - A decomposition of a relation R into relations R_1, R_2, ….,R_n is called a loss less join decomposition (W.R.T FDs of F) if the relation R is always the natural join of the relations R_1, R_2, ….,R_n.
    - Let a relation R which is decomposed into R_1 and R_2 . if the decomposition is loss less join. Then one of the following two conditions must hold.
      - $R_1 \cap R_2 \rightarrow R_1\text{-}R_2$
      - $R_1 \cap R_2 \rightarrow R_2\text{-}R_1$
        That is common attributes in R_1 and R_2 must include a candidate key of either R_1 or R_2
        Example:
        1. R={A, B, C, D, E}
        R_1 ={A, D} R_2 ={A,B} R_3 ={B, E} R_4 = {C, D, E} R_5 = {A, E}
        FDS F= A→ C
        B→C
        C→D
        DE→ C
        CE→ A

[Find common between 2 relation and check. Whether it is L. H. S of F. If it is in L. H. S, then 2 relations are depend]

Here , the relation R is lossy (I,e not loss less) because R_4 is not dependent on any relation.

2. Student (Regd no, Subcode, Enroll, R- no, facullty)
   Suppose student relation is decomposed into
   Student (Regd no, Subcode, Enroll- date)
   Student2 (Regd no, R –no, faculty)
   The above relation is called loss less join decomposition because if perform join among these relation, then we get required information without any loss.

- Dependency Preservation:
  - Let f be the dependencies on a relation R which is decomposed in relations R_1, R_2, ….,R_n .
  - We separate dependencies given by F such that F_1, F_2, ….,F_n
  - Fn are dependencies that only invalve attributes form relations R_1, R_2, ….,R_n representation.
  - If the union of dependencies F1 implies all the dependencies in F, then the decomposition has preserved dependencies, otherwise not (i. e. $F^1 = F_1 U F_2 U …..F_n$)
    Example:
    i. R = (A, B, C, D) with FDS F = {A→ B, A→C, c→D}
       R_1 = (A, B, C) with FDS F_1 ={A→B, A→C}
       R_2 =(C, D) with FDS F_2 ={C→D}
       Here F= $F_1 U F_2$
       Hence, the decomposition is dependency preserving.
       Also the common attributes between R_1 and R_2 is ‘c’, which is L. H.S of F.
       Hence, the decomposition of R into R_1 and R_2 is loss less join.
    ii. R = (A, B, C, D) with FDS F = {A→ B, A→C,A→D}
       R_1 = (A, B, D) with FDS F_1 = {A→B, A→D}
       R_2 =(B, C) with FDS F_2 ={B→C}
       The decomposition is lossy because common attributes between R_1 and R_2 ‘B’ is not in L.H.S of FDS F.
       It is also not dependency preserving since A→C is not implies by.

Normalization:

- This process was first proposed by Boyce – codd in 1972.
- Normalization is a stepwise process that allows decomposing database tables in such a way that both data redundancy and update anomalies are minimized.
- The process of reducing data redundancy in a relational database is called normalization.
- The principle of normalization is “ the same data should not be stored in multiple places”.
- It is a refinement process of database design.
- In this process, no information is lost but the No. of tables increases as the rules are applied.

❖ **Need for normalization or pitfalls in relational database design:**

- no data value should be duplicated in rows
- each relation should be self – continued i. e if a row from a relation is deleted, important information should not be lost.
- Insert anomaly: In some cases, insertion of new data is difficult.
- Update anomaly: In some cases, updating of existing data is difficult.
- Delete anomaly: In some cases, deleting of existing data is difficult.
- There are pit falls of relational database design. Hence, we need to refine the design to make an efficient database.

- This refining technique is called as normalization.

❖ **Steps of Normalization:**

The process analyzing and decomposing complex relation into simple relation is called normalization.

Step – 1: Separate repeating and non – repeating attributes
Step – 2: Remove particular dependency
Step – 3: Remove transitive dependency
Step – 4: Remove multi-valued dependencies (Boys – codd normal form (BCNF))
Step – 5: Decompose tables such that further decomposition is not possible.
(Project joins normal form)(PJNF)
Step – 6: Domain key normal form. (DKNF)

❖ **Normal forms:**

- The process of analyzing and decomposing the complex relation into simple relation is called as normal form.
- To avoid the difficulties of relational database, normal form was introduced by Dr. E.F. codd and R. Boyce codd in 1972.
- There are 6 types of normal form. There are
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF) and boyce codd normal form.
  - Forth normal form(4NF)
  - Fifth normal form (5NF) or project join normal form(PJNF).
  - Sixth normal form or domain key normal form (DKNF)
    - First normal form(1NF)
      A relation is said to be 1NF, if it has atomic attributes. That means attributes which can't be subdivided.
      Example:

Name	Age	Phone	Sex

[un normalized form]

First name	Middle name	Last name	Age	Phone	Sex

[1st normal form]

    - Second normal form(2NF)
      A relation is said to be in 2NF, if it is in 1NF and satisfy any one o the following condition.
      - The primary key consists of only one attribute.
      - There exist no non- key attributes
      - Every non – key attributes present in the relation should functionally depends on primary key.
        Example: Student:

Roll	Name	Branch	Address

[un – normalized form]

Student:

Roll	First name	Middle name	Last name	Branch	Street	City	Pin

[1st NF]

Now this relation satisfies the first condition I,e. the primary key consist of only one attribute. Hence, this relation is in 2NF

- Third normal form (3NF)
  A relation is in 3NF if it is in 2NF and there exist no transitive FD of any non -key attributes on primary key.
  In a given relation R, x, y and z are attributes the dependency is called as transitive if Z depends on y and y depends on x.
  The attributes which is not taking port in formation of key is considered as non – key attribute
  Example:
  Work project (Work id, Dept, Project, Proj - Loc)
  Here, FDs are worked, Dept→ project
  Project→ proj – loc. Since it is decompose the relation into 2 relations. Work project (Work id, Dept, Project) and Project (Project, Proj - loc)
  These relations are in 3NF
- Boyce codd normal form (BCNF):
  BCNF is a strong 3NF, but every 3NF relation is not BCNF.
  A relation is said to be BCNF if and only if it is in 3NF and every determinants are candidate keys.
  [Determinant is the L.H.S term of FD]
  BCNF removes dependency among key attributes
  Example:
  Student (Roll, course, email, grade)
  FDS are Roll → email
  Email→ roll
  Roll, cause→ email and grade
  Course, email→ roll & grade
  Here above determinants are not candidate key hence it is not in BCNF but it is in 3NF.
  So, to make this relation in BCNF, decompose into 2 parts.
  Student (Roll, email) and mark (roll, course, grade)
  FDs Roll→email
  FDs Roll, course→grade.
  Now this relation is in 3NF and also in BCNF because both the determinate are candidate key.
- Forth normal form (4NF):
  It deals with multi valued dependency.
  A relation is said to be 4NF if it is in BCNF and every multi valued dependency x→ y holds and satisfy one of the following.
  - ♦ X is a super key
  - ♦ X is a subset or equal to R (or) XUY = T

  Example:
  Emp(ename, proj name,dept)
  Here, non trivial MVD Ename→proj name and Ename→ dept exist and Ename is not a super key.
  Emp1 (Ename, proj – name) Now, emp1 and emp2 are in 4NF because MVDS in emp1 and emp2 are trivial MVDS and emp2 (ename, dept)
- Fifth normal form or (PJNF):

5NF is also called as PJNF.
It deals with join dependency.
A relation is said to be in 5NF if it is in 4NF and non trivial join dependency ID
$R_1, R_2, ...,R_n$ in F, R is a super key of R.

- Sixth normal form or domain key:
  A table is said to be in 6NF if it is in 5NF and if enforcing domain constraints and also key constraints enforce all dependencies specified on the relation.

Multi valued Dependency (MVD):

If there exist $t_1, t_2 \in r$ such that $t_1[x]= t_2 [x]$ and if there exist $t_3, t_4 \in r,$ then that satisfy the below condition.

- $T_3[x]=t_4[x]=t_1[x]$
- $T_3[y]=t_1[y]$ and $t_4[y]=t_2[y]$
- $T_3[z]= t_2[z]$ and $t_4[z] = t_1[z]$

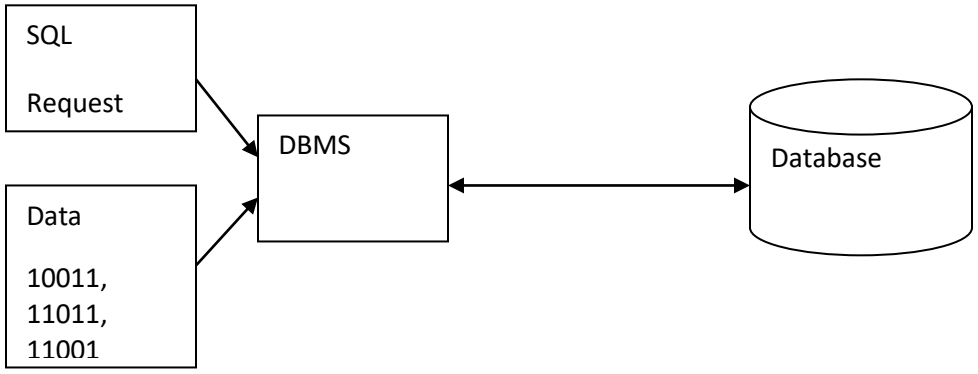
Whenever X→y holds, say X multi determine y if x->z holds in this relation, then say x→ y|z

Trivial multi valued dependency:

A MVD X→ Y in r is called trival MVD if $Y \subset X$ or XUY= r.

Structure Query Language (SQL):

- SQL is used to interact database to manage and retrieve data.
- The DBMS processes the SQL request, retrieves the requested data from the database and returns it.
- This process of requesting data from data and receiving back the result is called a database.
  Query and lence the name SQL



➔ Create: Create a table student with the following specification.

Column name	Data type	Constraint
Rollno	Number(10)	Primary key
Name	Varchar2(25)	Not null
Branch	Varchar2(15)	Not null
Sex	Varchar2(10)	Not null
Address	Varchar2(25)	Not null
Phone no	Number (15)	Not null

ANS: Create table student (Roll No number (10))
Constraint P_k primary key, name varchar2 (25),
Constraint not null, branch varchar2 (15),
Constraint not null, sex varchar2 (10),
Constraint not null, address varchar2 (25),

Constraint not null, phone no number (15),

➔ Create a table with the following specification

Column name	Data type	Constraint
Eid	Number (10)	Primary key
Name	Varchar2(25)	Not null
E mail	Varchar2(25)	Not null
Dept	Varchar2(10)	Not null
Manage id	Number (15)	Not null

ANS: create table employee (Eid numer(10) constraint P_k primary key, name varchar2 (25) Constraint not null, email varchar2 (25), dept varchar2 (10), managerid number (10), constraint F_k referencing employee (eid)

➔ Create emp table from employee table having attribute Eid, email.
ANS: create table emp as select Eid, E mail from employee

Alter:

1. Add age to the student table
   Ans: alter table student add (age number (10));
2. Modify the size of phone column to number (10) in student table.
   Ans: alter table student modify (Phno number (10));
3. Add the not null constraint on the dept column of employee table.
   Ans: alter table employee add (dept varchar2 (20) constraint nnul not null);
4. Add a unique constraint on phone attribute of student table.
   Ans: alter table student add (constraint uns unique (phone));
5. Drop not null constraint of branch column of student.
   Ans: alter table student drop (branch varchar2 (10) constraint nnull not null);
6. Drop phone attribute form student table.
   Ans: alter table student drop (Phno number (10));
7. Adding check constraint to C – id in customer table.
   Ans: alter table customer add (constraint chk, check (C- id between 100 and 999));

Drop:

This is used to drop a table from the database
Syntax:
Drop table tablename ;

Truncate:

It is used to delete all rows from a table.
Syntax:
Truncate table table name;

Data manipulation language:

Insert:

It is adds the rows into the database table.
Syntax: insert into whether values ('&city', '&tem', '&table');

Delete:

It is used to delete data form database table.

- Delete the customer table.
  Delete from customer.
- Delete all rows containing whether data from Mumbai city

Delete from whether where city= ‘Mumbai’;

Update:

It updates data in the database table.

Syntax: update table name set column name= new value.

- Update the weather table average temp 20.
  Update whether set temp = 20;
- Update the temp 22 of column a city
  Update whether set temp =20 where city = ‘culcutta’

Select:

It selects data from one or more database table.

Syntax:

Select column names, column name2 form table name.

Select retrieve city column from whether select city from whether

Where clause:

It works in consumption with other SQL clauses like select, insert and update to specify a search for the statements

- Retrieve all data in whether table whose temp more than 20 and city new delhi
  Select * from whether where temp>20 and city = ‘new delhi’;
- Retrieval all information from whether table except newyork city.
  Select * from whether where city <>(newyork);

Distinct clause:

It selects only distinct (unique data) from database.

Syntax: select distinct column from table.

- Display all city of whether table.
  Select distinct city from weather.

Order by clause:

It defines in what order to return a data set retrieve with SQL select statement.

Select * from whether order by city;

Select * from whether order by city desc;

Aggregate function:

These functions are used to sum, count, get the average, get the minimum, get the maximum value from a column or from a subset of column values.

- Count the rows in a whether table.
  Select count (*) from weather;
- Find the average temp for whether table
  Select avg (temp) from whether;
- Find the minimum marks from student table
  Select min (mars) from student;
- Find the maximum salary from employee table
  Select max (salary) from employee;
- Find the sum of commission from employee table
  Select sum (comm) from employee;
- Count the no of student stay on BAM
  Select count (*) from student where city ‘BAM’

Group by:

It is uses along with SQL aggregate function and specifies the groups were selected rows are places.

- ➔ Calculate the average temp for each of the city in whether table.

Select city, avg (temp) from whether group by city.

Having keyword:

It provides such condition for a group or aggregate.

There is difference between having and where clause.

Where

Where condition is tested against each and every row of data a while.

Having

Having condition is tested against the groups/aggregate specified in SQL group by clause.

➔ Find all city with average temp greater than 29 c
Select city, avg (temp) form whether group by city having Avg (temp)>29c

Join clause:

Weather

City	Avg. temp	Date

State

State	City

It selects data from two or more table type together by matching table column.

➔ Find all data from whether table and also state column to the result.
Select w. city, w.avg, temp, w.date, s.state from whether w, states where w.city = S.city.

➔ Select employee no, name, basic and depart no and department name form employee table and department table.
Select e.empno, e- name, e. basic, d. name from employee, dept d where e.deptno = d.deptno;

Between operator:

The between operator includes both and values specified.

➔ List all acc no with balance in the range 10,000 to 50,000
Select acc-no from customer where balance bet 10,000 to 50,000;

In operator:

The in operator is used to check if a value belongs to a set of values or not.

➔ List all customer who have account in Berhampur or Chatrapur
Select C- id from customer where branch in ('Berhampur', 'Chatarapur');

Like Operator:

Like operator is used to check for similarity of string.

➔ List all account where branch begins with 'D' or '0' as 2nd letter.
Select * from customer where branch like 'D0%';

Grant:

It is used to Grant security privileges to specific leaders. Normally it is used by the owner of the table to give other used access to the data.

Syntax:

Grant selects/ insert/ update/ delete all privilege a tablename to username.

Revoke:

It is used to revoke privilege previously granted with grant statement.

Syntax:

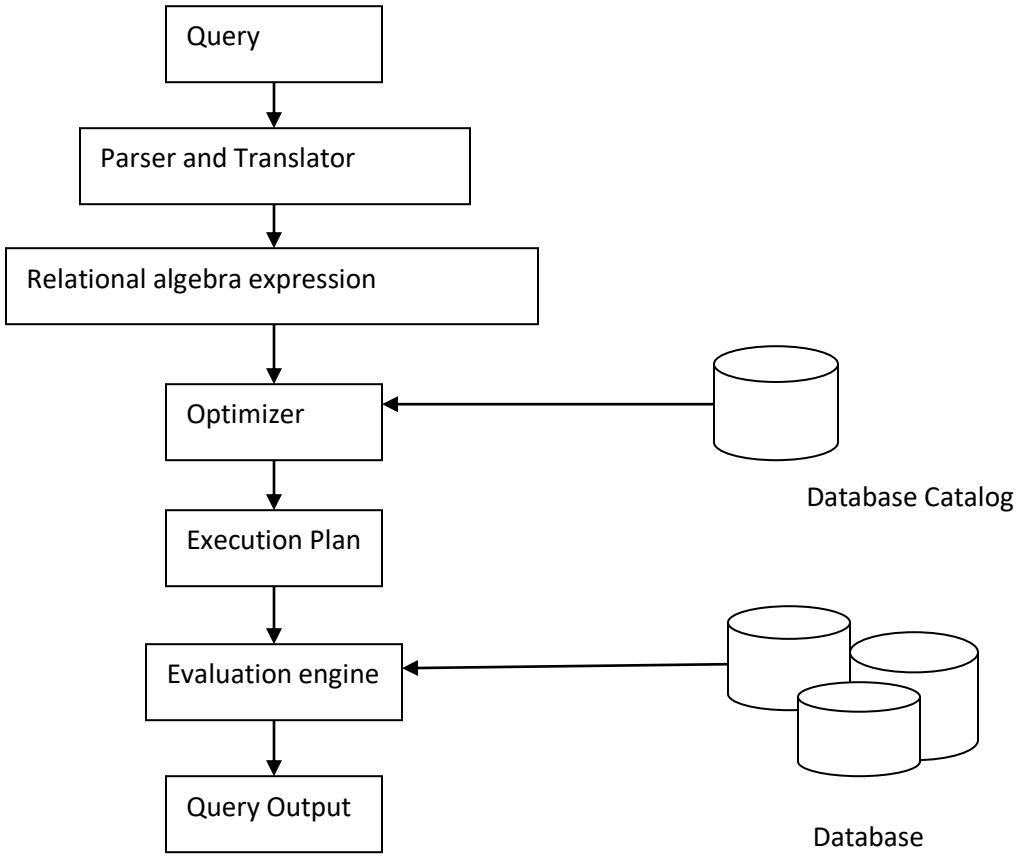
Revoke select/ insert/ update/ delete all privilege on tablename from user ame.

Query by example (QBE):

- QBE was developed by IBM in 1970 to help the database users to access and retrieve information from the database.
- It is a powerful facility that gives the end user the capability to access the information without any programming knowledge, not even SQL.

- QBE works as follows:
  - When we create a query using QBE interface, DBMS (e.g MS-Access) constructs the equivalent SQL statements.
  - This SQL statement is used to retrieve the information from the database table (S).

Query Processing:



- It is a stepwise process of transforming a query written in high level language (like SQL) into a correct and efficient execution strategy expressed in a low level language and to execute the strategy to retrieve the required data.
- Steps involved in query processing and query optimization.

  Step – 1

  Transform the query into a standard form I, e. a query expressed in QBE is translated into SQL and subsequently into a relation algebraic expression.

  Step – 2

  During transformation process, the parser and translator checks the syntax and verifies whether relations and attributes used in query are defined in the database.

  Step – 3

  Query optimization is performed by transforming the query into equivalent expressions that more efficient to execute.

  Step – 4

  The transformed query is used to create a no. of strategies called access plan/execution plan.

Step – 5

These access plans are used to evaluating the transformed query by evolution engine. The cost of each access plan is determined and the plan with least cost is executed.

Step – 6

Query output will gives the result.

Query optimization:

- ❖ It is the process of selecting the most efficient query evolution plan from among the many strategies usually possible for processing a given query.
- ❖ As there are many equivalent transformations of some high – level query, the aim of query optimization is to choose one that minimizes resource usage.
- ❖ Generally, reduce total execution time of query.

Translating SQL Queries into Relational algebra:

1. Consider two relations:

Emp(Eid, Ename, Desig, Did)

Dept(Did, Dname, Dloc)

Query

Find all the employee who work in 'CSE' department.

SQL:

Select * form EMPE, DEPTD where E.Did=D.Did and D.Dname='CSE';

RA1: Temp← π Did ($\sigma_{Dname = 'CSE'}(Dept)$)

π Eid, Ename, design, Did(Emp E.did = Temp.Did|X| TEMP)

RA2: π Eid, Ename, Desig, Did(⌐dname='CSE'(EMP|X|DEPT)E.Did= D.Did);

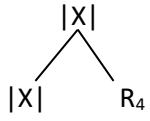
Query Tree:

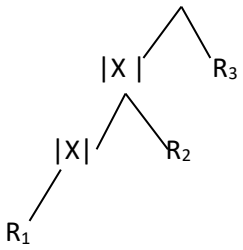
Roof: The desired result of the query.

Leaf node: Shakes relation.

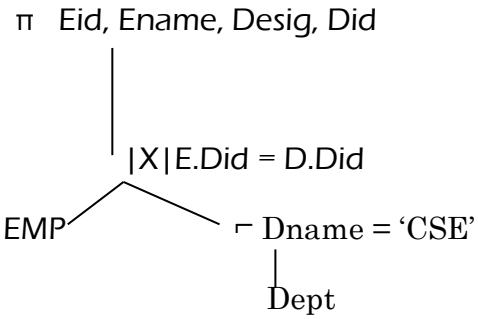
Non – leaf node: Intermediate relation created form relational algebra operation.

2. $R_1|X|R_2|X|R_3|X|R_4$

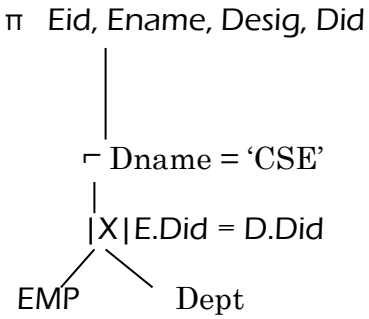




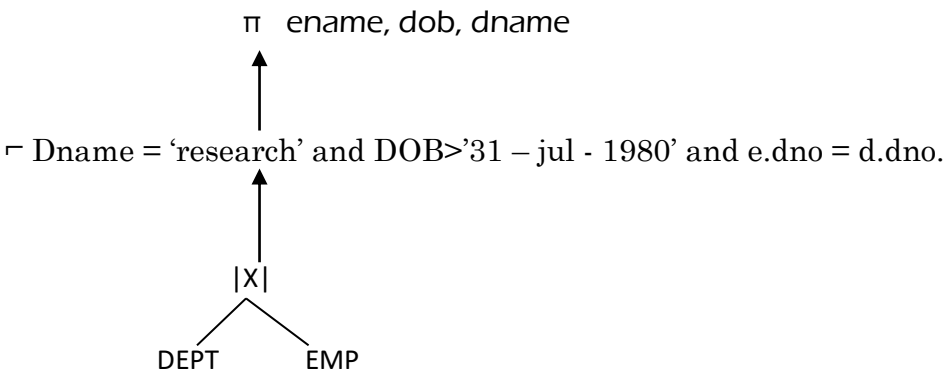
Query tree for RA_1



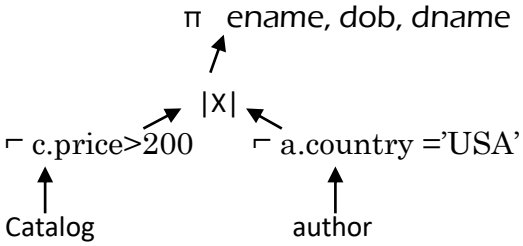
Query tree for RA_2



3. Find employee name who bean after 31 – Jul – 1980 and works on research department.
   Select e.name, e.dob, d.dname, emp, depted where dname = 'Research' and e.dno = d.dno and DOB> '31 – jul - 1980';



4. Select * from catalog c, author a where c.aid= a.aid, and (c. price>200 and a. country = 'USA');



****

- **Database Development Life Cycle (DDLC):**
  - It is a part of (embedded inside) the s/w development life cycle.
  - Different phases of DDLC are:
    - Requirement analysis.

- Database design
- Evolution and selection
- Logical database design.
- Physical database design.
- Implementation.
- Data loading.
- Testing and performance tuning.
- Operation.
- Maintenance.

1. **Requirements Analysis:**

- The 1st step is to find out
  - What is required?
  - What kind of a database is needed for the organization?
  - What is the volume of data?
  - How much data to be starred in the master file and soon?
- To get this information the database analyst should spend a lot of time.
- The analyst should also gather detailed and accurate information on the no. of users.
  - No of users simultaneously accessing the system.
  - Performance expectations of the users.
  - The rate at which the database will grow and soon.
- The main goal of this phase is
  - Study the existing system.
  - Define problems and constraints of the database environment.
  - Define design objectives.
  - Define standards and guidelines.

2. **Database Design:**

- In this phase, the database designer study the documents prepared by the analyst and will go about developing a system that satisfies the requirements.
- First a conceptual design of the database is created.
- In this stage, data modeling is used to create an abstract database structure that represents the real world scenario.
- The conceptual design is h/w and s/w independent.

3. **Evolution and Selection:**

- Once the data model is created, tested and verified, the next step is to evaluate the different DBMS and select one that is suited the needs of the organization.
- The main factors influence the selection of the DBMS are:
  - Cost of the system.
  - Features and tools.
  - Customer support and training.
  - Portability.
  - H/W requirement.

4. **Logical Database Design:**
- Logical design is S/W dependent.
- In this state, the conceptual design is translated into internal model for the selected DBMS.

5. **Physical Database Design:**
- It is the process of selecting the data storage and data access characteristics of the database.
- Physical design translates the logical design into h/w dependent one.

6. **Implementation:**
- It requires the creation of special storage related constructs to use the end user table.
- These contracts usually include storage graph, table – space, data files, tables and soon.

7. **Data Loading:**
- After creating the database, the data must be loaded into the database.
- It data stored in different format, then the data needs to be converted and then me grated to the new database.

8. **Testing and Performance testing:**
- Testing and performance tuning occurs in parallel with application program.
- The DBA and application programmers work hand – in – law during this phase.

9. **Operation:**
- During this phase, the database is accepted by the users and application programs, new data is added, the existing data is modified and some absolute data is deleted.
- The database delivers its usefulness as a critical tool in management decision making and help in smooth and efficient functioning of the organization.

10. **Maintenance:**

The main tasks in this phase are.
- Database backup and recovery.
- Performance tuning.
- Database access management.
- Database audits.
- H/W maintenance.
- DBMS S/W up gradation.

Transaction

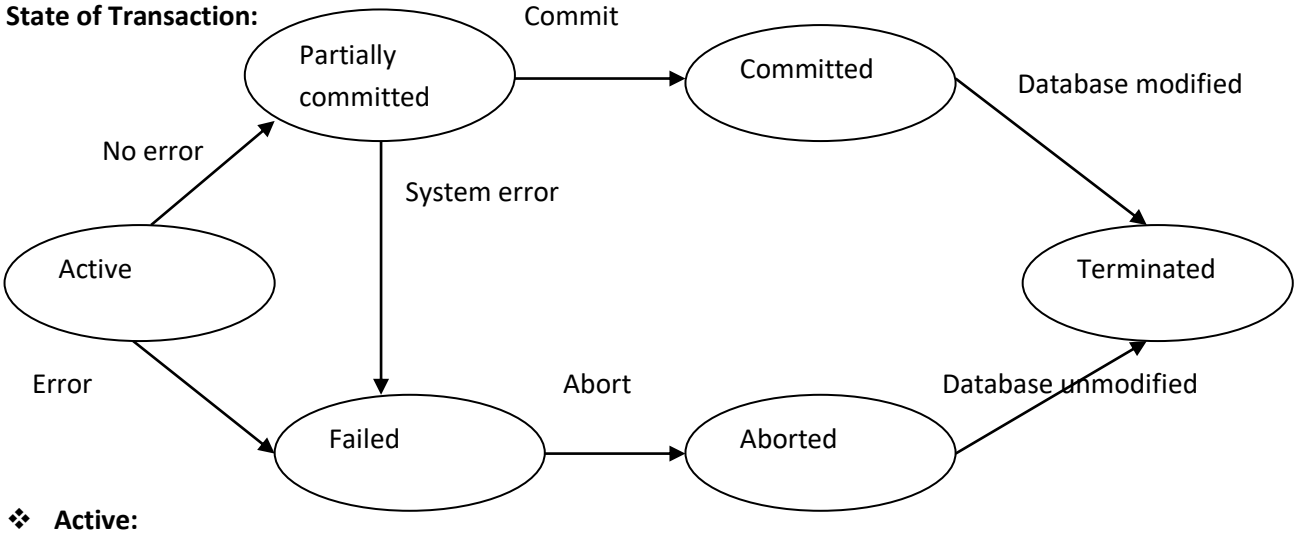
- Collections of operations that form a single logical unit of work are called transaction.
- It access data using two operations
  - **Read (X):**
    If transfers the data item (x) from the database to a local buffer belonging to the transaction.
  - **Write (X):**
    It transfers the data item (x) from the local buffer of the transactions that executed the write back to the database.

Properties of Transaction:

Transactions have four basic properties which are ACID properties.

- **Atomicity:**
  - It is an atomic unit of processing.
  - It is either performed completely or not performed at all.
  - The user must know in which state he is present.
- **Consistency:**
  If a database was in consistent state before the initiation of a transaction, then at the end of transactions the database will also be a consistent state.
- **Isolation:**
  It indicates that the execution of a transaction should not be interfered by any other transaction being executed concurrently.
- **Durability:**
  - The changes applied to the database by a committed transaction must persist (permanent) in the database.
  - These changes must not be lost because of any failure.

State of Transaction:



- **Active:**
  It is the initial state, the transaction says in this state white it is executing.
- **Partially committed:**
  After the final statement has been executed.
- **Failed:**
  When normal execution can no longer proceed.
- **Committed:**
  After successful completion of the transaction.
- **Aborted:**
  After the transaction has been rolled back and the database has been restored to its prior to the start of the transaction.

- **Terminated:**

  Once transaction leaves of system is said to be terminated.

  When the transaction enters to the aborted state the system has two operations

  - **Restart the transaction:**

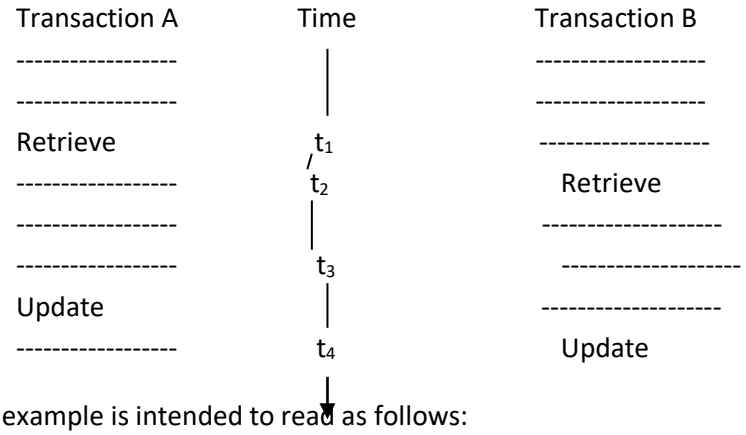
    If some hardware or software failure. A restart transaction is considered to be a new transaction.

  - **Kill the transaction:**

    It usually does so because of some logical error that can be corrected only by rewriting the application program

- **Concurrency:**
  - When multiple transactions of a database executed at same time is known as concurrency.
  - It performs various database updates simultaneously.
  - Concurrency control is the process of managing simultaneous operations on the database without having the interface with one another.
  - The main reasons of using concurrency are:
    - Improved throughput and resource utilization.
    - Reducing waiting time.
    - Concurrent execution of transaction may lead to certain problem if not handled in controlled manner.
    - There problems are

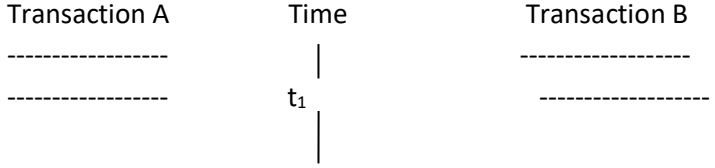
      Lost – update problem:



The above example is intended to read as follows:

Transaction A retrieves some tuples P at time t_1, transaction B retrieve same tuple P at time t_2, transaction A updates the tuple at time t_3 and the transaction B updates the same tuple at time t_4. Transaction A's update is lost at time t_3 because transaction B over writes it without even looking at if.

The uncommitted dependency problem:



Retrieve		Update P
----------	t_2	----------
----------	|	----------
----------	t_3	rollback
	↓	

Transaction a sees an uncommitted update at time t_2 that update is then undone at time t_3. Transaction A is therefore operating on a false assumption and produce incorrect output.

Inconsistent Analysis Problem:

Transaction A	Time	Transaction B
Retrieve AC_1 sum = 40	t_1	
	|	
Retrieve AC_2 sum = 90	t_2	
	\	
	t_3	Retrieve AC_3
	|	
	t_4	Update AC_3, 30→20
	|	
	t_5	Retrieve AC_1
	|	
	t_6	Update AC_1, 40→50
	|	
Retrieve AC_3	t_7	
	|	
	t_8	Commit
	↓	

Transaction A and B operating an account tuples. Transaction A is summing account balance, transaction B transferring amount to from AC_3 to AC_1. The result produce by A=110 is

Serializability:

When several transactions are executing concurrently, then the order of execution of various instructions is referred as a schedule.

Type of Schedule:

Different types of schedules are

- Serial Schedule
- Non – serial schedule

- ❖ Conflict schedule
- ❖ View schedule

I. Serial Schedule:

Schedule A

T_1	t_2
Read (A)	
A= A-100	
Write (A)	
Read (B)	
B= B+100	
Write (B)	
	Read (A)
	T= A*0.1
	A= A-t
	Write (A)
	B= B+t
	Write (B)

Schedule B

t_1	t_2
	Read (A)
	t= A*0.1
	A =A-t
	Write (A)
	Read (B)
	B= B+t Write (B)
Read (A)	
A=A-100	
Write (A)	
Read (B)	
B= B+100	
Write (B)	

Suppose there are two schedules A and B called serial schedule. If the operations of each transaction are executed consecutively without any interleaved operations from the other transaction.

In a serial schedule, entire transactions are performed in serial order. T_1 and then T_2 and T_2 and T_1.

II. Non – serial schedules:

A schedule in which the operations from a set of concurrent transaction are interrelated is called a non – serial schedule.

T_1	T_2
Read (A)	
A= A-100	
	Read (A)
	T= A*0.1

A= A- t

Write (A)

Read (B)

Write (A)

Read (B)

B= B+100

Write (B)

B= B+ temp

Write (B)

III. **Conflict schedules/ conflict serializability:**

- If graph contains no cycle, then the schedule s is conflict serializable schedules.
- Let us consider a schedule in which there are two consecutive instructions T_i and T_j of transactions T_i and T_j respectively.
- If T_i and T_j refer to different data items, then we can swap T_i and T_j without affecting the results of any instructions in the schedule.
- If T_i and T_j refer to some data item Q then the order of the two steps may matter.
- There are four possible cases of read () and write () operations.
  - T_i = read (Q), T_j = read (Q)
    The order of T_i and T_j does not matter since the same value Q is read by T_i and T_j.
  - T_i = read(Q), T_j =write(Q)
    - If T_i comes before T_j , then T_i does not read value Q written by T_j in instruction T_i.
    - If Tj comes before T_j , then T_i read value Q written by T_j . thus, the order of T_i and T_j matters.
  - T_i = write(Q), T_j = read (Q)
    The order of T_i and T_j matters for reasons similar to those of the previous case – 2
  - T_i = write (Q), T_j =write(Q)
    Since both instructions are write operations the order of these instruction does not affect either T_i or T_j .
    Hence, the T_i or T_j conflict if they are operation by different transactions. On the same data item and at least one of these instructions is a write operation.

T_1	T_2
Read (A)	

T_1	T_2
Read (A)	

Write (A)			Write (A)	
	Read (A)		Read (B)	
Read (B)			Write (B)	
	Write (A)			Read (A)
Write (B)				Write (A)
	Read (B)			Read (B)
	Write (B)			Write (B)

- ♦ Swap Read (B) of T_1 with read (A) of T_2.
- ♦ Swap write (B) of t_1 with write (A) of T_2.
- ♦ Swap write (B) of with read (A) of T_1.

- If a schedule S can be transformed into a schedule S by a series of swap o non – conflicting instruction, we say that S and S^1 are conflict equivalent.
- A schedule S is said to be conflict seralizable if it is conflict equivalent to some of the serial schedule.

IV. **View schedules/ View serializability:**

- ❖ Consider two schedules s_1 and S_2 where some set of instructions participate in both schedule.
- ❖ The schedules S_1 and S_2 are said to be view equivalent if the following conditions are satisfied.
  - For each data item X, if transaction T_i executes Read (A) in schedule S_1, the transaction T_i must in schedule S_2 also read the initial value X.
  - For each data item X, if transaction T_i executes read (A) in schedule S_1 and the value was produced by transaction T_j, then t_3 must in schedule S_2 also read value of X that was produced by T_j.
  - For each data item X, the transaction that performs final write (X) operation in schedule S_1 must perform the final write(X) in schedule S_2..
  - A schedule S is view serializbility if it is view equivalent to serial schedule.

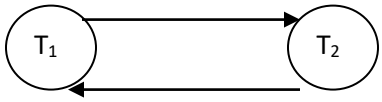
- **Blind Write:**

T_1	T_2	T_3
Read (Q)		
	Write (Q)	
Write (Q)		Write (Q)

In transaction T_2 and T_3 perform write (Q) operation without having performed a read (Q) operations such write operation is called blind write.

- **Testing serializability:**
  - In order to test the serializability the simple and efficient method is to construct a directed graph, known as precedence graph from S.
    G = (V, E) where V is a set of verities and E is a set of edges.
  - The set of vertices consists of all transaction participating in the schedule.
  - The set of edges consist of all edges $T_3 \rightarrow T_1$ for which one of the following 3 conditions holds.
    1. Example:

T_1	T_2
Read (A)	
A : A- 50	
	Read (A)
	T: =A*0.1
	A: =A – t
	Write (A)
	Read (B)
Write (A)	
Read (B)	
B: =B +50	
Write (B)	
	B: =B++
	Write (B)

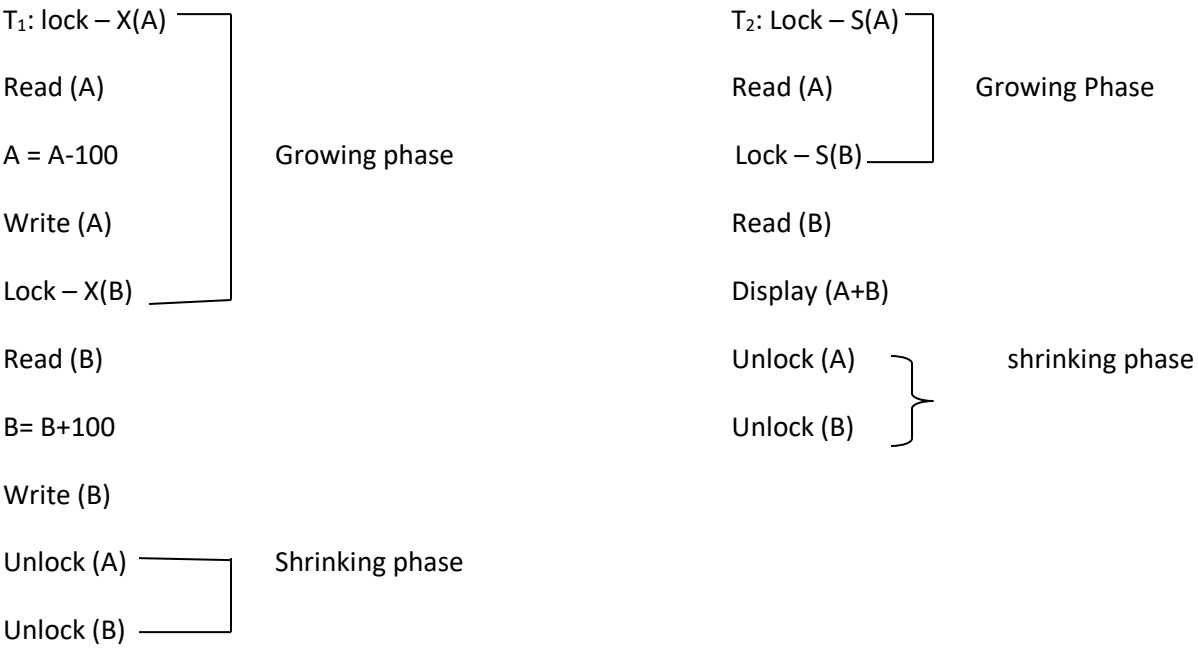


- **Mode of locking:**
  - Locking is a procedure to control concurrent access to data.
  - When one transaction is accessing the database a lock may be deny access to other transactions to prevent incorrect results.
  - It ensures serializability.
  - A transaction must obtain a read or write lock on a data item before it can perform a read/ write operation.
  - There are mainly two modes in which a data item may be locked.
    1. **Shared lock:**
       - If a transaction, T_i has obtained a shared made lock on item Q, then T_i can read but cannot write Q.
       - Usually select operation takes this lock on resources
    2. **Exclusive lock:**
       - If a transaction T_i has obtained an exclusive made, lock an item Q, then T_i can both read and write Q.
       - Usually insert/update/delete operation takes X lock.

- **Two phase locking protocol:**
  - If ensure serializability of multiple transaction.

- It requires each transaction issue lock and unlocks requests in two phases.
  1. **Growing Phase:**
     In this phase a transaction can obtain lock but cannot release any lock.
  2. **Shrinking Phase:**
     In this phase a transactions can only relive locks but not obtain any new locks.
     - Initially a transaction is in the growing phase, the transaction acquires locks on data items as needed. Once a transaction release a lock it entries the shrinking phase and if cannot issue any more lock request.
     - The point in the schedule where the transaction has obtained its final lock (end of growing phase) is called lock point of transaction.

Example:



- **Time stamp ordering protocol:**
  It is concurrency control protocol in which the fundamental goal is that older transaction gate priority in the event of a conflict.
  - In this method if a transaction attempts to read or write a item then it is allowed only if the last update on the data item was carried out by an older transaction otherwise the transaction request the read or write is restarted added and given a new times tamp.
  - There are two method of implementing this schema
    1. Use the value of system clock as timestamp.
    2. Use a logical counter i. e. incremented one after a new timestamp has been assign.
  - There are two type of timestamp:
    1. R – timestamp (Read):

It contains the timestamp of the last transaction the read the item.

2. W – time stamp (Write):
   It contains the timestamp of the last transaction the update the item.

- For a transaction T the timestamp ordering protocol works as follows.
  - Transaction T request to read the data item 'X' that already been updated by younger transaction. This means that earlier transaction is trying to read a data item that has been updated by a latter transaction. In this situation, T aborted and restarted with a new timestamp.
  - In all other cases the transaction is allows the produced with the read operation and R – timestamp of the data items is the updated with the timestamp of T.
  - Transaction T request to write (update) the item (X) that has already been read by a younger transect. This means that the younger transaction is already using the current value and it would be an error to update if new in this case T is aborted and restarted a new time stamp.
  - Transaction asked to write the data item(X) that has already been written by a younger transaction this means T is attempting to write and old value of item. In this time T is aborted and restarted with timestamp.
  - In all other cases T is allowed to precede W time stamp of data item is updated with time stamp of T.
- **Thomos's write rule:**
  - The modification of timestamp ordering protocol is called as "Thomas's Write operation".
  - It modifies the checks for a write operation by transaction T as follows
    - ♦ When T request to write the data item 'X' whose value has already been read by younger transaction, T is aborted, rollback and restarted with new stamp.
    - ♦ It request to write the data item 'X' whose values has already been written by a younger transaction this means that T is writing a old value to a data item. In this case the write operation is ignored and T is allowed to continue as if the write where perform. This principle is called ignored absolute write rule.
    - ♦ In all other cases T is allowed to proceed to w – timestamps data item is updated with the timestamp T.
- Types of Database failure:
  There are various types of failures that may occur in a system
  - **Transaction failure:**

  There are two types of errors may cause of transaction to fail.
    - ♦ **Logical error:**
      The transaction cannot be executed because bad input of data not found, overflow etc.
    - ♦ **System error:**

The system has entered an undesirable state such as deadlock as a result the transaction cannot be executed.

- **System Crass:**
  - There is a hard ware problem or error in chart database s/w or error in operation system chart causes the loss of the content of volatile storage and transaction execution to half.
  - This problem does not corrupt the non – volatile storage contents are known as fail – stop assumption.
- **Disk Failure:**
  - A disk block loses its content as a result of either a head crash or failure during a data transfer operation.
  - Copies of the data on other disks for backup are used to recover from the failure.
- **Database Recovery:**
  - It is a technique that restores the database to the state that exists before the failure.
  - The most widely used schema for the database recovery is log- based recovery.
- **Log – based Recovery:**
  - The structure used for recording the database modification is called the log.
  - The log is a sequence of log records and maintains a record of all the updated activities in the database.
  - A log record is maintained for each write (X) operation on the database and consists of 4 fields.
    - **Transaction identifier:** It is the unique identifier of the transaction that preformed the write operation.
    - **Data - item identifier:** It is the unique identifier of the data item written.
    - **Old Values:** It is the value of the data item prior to write.
    - **New value:** It is the value of the data item after the write.
  - The log consists of various types of log records as
    <T_i start>: Transaction T_i has started.
    <t_i,X, V_1, V_2>: transaction T_i has performed a write operation on data item X.
    X has values V_1 before write and will have value V_2 after write.
    <T_i , commit>: transaction T_i has committed.
    <T_i, abort>: transaction T_i has aborted.
  - The log is written before any updates are made of the database, then this is called write ahead log strategy.
  - Recovery From crash:
    There are three phases of recovery:
    1. **Do:** Transactions of current database transaction if from the current state to a new state, this is called do operation.

2. **Undo:** It reverses the actions of the transaction and restore the database exist before start of transaction. It is called into play.
3. **Redo:** It reduces the action of the transaction and restores the database to the state if would be at the end of the transaction.

- **Update strategies:**

There are two techniques used to ensure atomicity which are based on log.

1. **Deferred update:**
   - All write within transaction are recorded in log.
   - During lifetime of transaction, this writes invisible to other transactions.
   - In this technique, the execution of transaction T_i proceeds as follows.
     - T_i starts execution, a log record <T_i, start> is written to log.
     - When T_i executes a write (X) operation, a new log record <T_i,X, V_2>is written to log where v_2 is new value.
     - When T_i is partially committed a record <T_i commit> is written to log. All the updates are reflected in actual database.
     - The deferred write technique requires recording only the new value of the data.
     - Thus, in log record of each update, we omit the old value of x.

Example:

T_1: read (A)	T_2: Read (c)
A = A -50	C= C-100
Write (A)	Write (c)
Read (B)	
B= B+ 50	
Write (B)	

Log	Database	Log	Database
<T_1, start> <T_1, A, 950> <T_1, B, 2050> <T_1, Commit>	 A = 950 B = 2050	 <T_2, Start> <T_2, C, 600> <T_2, Commit>	 C = 600

2. Immediate Update:

- In this techniques, all update made by the transaction are directly reflected in the database.
- If the system crash or failure occurs, the system must restore the old value of each data items updated by the transaction.
- Thus the transaction must maintain a log a record $<T_i, X, V_1, V_2>$ for each write operation on X where V_1 us the old value and V_2 is the value of X.
- In this techniques, the transaction T_i executes as follows
  - As T_i starts execution the system write a log record $<T_i$ start> into the log.
  - When T_i executes write (x) operation, the value of X is updated in the actual database, but the systems writes a log record$<T_i$, commit> into the log.
  - The transaction T_i needs to be redone if the log contains with the record $<T_i$ start> and $<T_i$, commit>

    Example:

Log	Database
$<T_0$, start> $<T_0$, A, 1000, 950> $<T_0$, B, 2000, 2050> $<T_0$, commit> $<T_1$, start> $<T_1$, C, 700, 600> $<T_1$, commit>	 A = 950 B = 2050 C = 600

$<T_0$, start>

$<T_0$, A, 1000, 950>

$<T_0$, B, 2000, 2050>

$<T_0$, commit>

$<T_1$, start>

$<T_1$, C, 700, 600>

$<T_1$, commit>

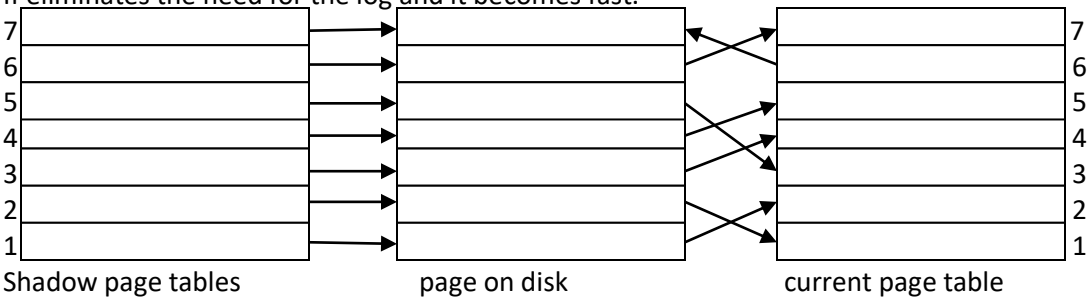
❖ **Check point:**

- When a system failure occurs we must consults the log to determine those transactions that need to be redone and those that need to be undone.
- Rather than reprocessing the entire log, which is time – consuming and much of it necessary use the check point.

- The systems periodically perform a checkpoint which requires the following sequence of actions to take place.
  - Output all log records onto stable storage currently residing in main memory.
  - Output all modified buffer blocks to the disk.
  - Output a log record <check point> onto stable storage.
  - Don't need to perform redo on a transaction have been commuted prior to a checkpoint.
  - After a failure search backward in the log for all transactions that were executing/ executed before and redo or undo all transaction.

❖ **Shadow paging:**

- In this technique a new page is allocated for the modified data and old page remains as a shadow page (Copy of the data).
- This is easily achieved by maintaining two tables of page address, one table called current page table and other is called shadow page table.
- When the transaction starts, both tables are identical.
- The shadow page is updated with write operation.
- If a failure occurs, before the transaction commits, the shadow page table is used to recover the database before the transaction started.
- When a transaction commits, the page in the current page table becomes the pages in the shadow page table.
- If eliminates the need for the log and it becomes fast.



- **Advantage:**
  - Overhead of log record output is eliminated.
  - Recovery from crashes is much faster.
- **Disadvantage:**
  - Commit overhead:
    The commit of single transaction requires multiple blocks to be output the current page table, the actual data and disk addresses of current page table.
  - Data fragmentation:
    It could by change of location when database page is updated.
  - Garbage collection:
    Find all of the garbage pages and add them to the list of free pages. This process is called garbage collection.
  - Garbage collection impasse additional overhead and complexity.
  - Difficult to be adapted for concurrent transaction.

Dead lock:

A system attends at deadlock state, when in a set of transaction; every transaction is waiting for another transaction to finish its operations.

Deadlock is a situation where one transaction is waiting for another transaction to release the resource it needs the vice – versa.

Example:

In fig, both the train is using the same railway track to cross each other; here one train has to wait forever for the situation is called as deadlock.

There are two methodic for dealing with deadlock problems.

1. Deadlock prevention protocol.
2. Deadlock detection and recovery process.

Deadlock prevention protocol:

- It ensures that the system will never enter into a deadlock state.
- Deadlock can be prevented if circular waits are eliminated. This can be done either by defining an order and who may wait for whom or by eliminating all waiting.
- Two different deadlock prevention schemes are used.
  - **Wait and die:**
    - This is a non- preemptive technique when a transaction T_1 requires a data item currently held by T_2. T_1 is allowed to wait only if it has a timestamp smaller. Then T_2 else T_1 is called back or dies.
    - Example: if T_1, T_2 and T_3 have timestamps 5, 10 and 15 min respectively.
      - If T_1 request data held by T_2, then T_1 will wait.
      - If T_3 request data held by T_2, then T_3 will dies.

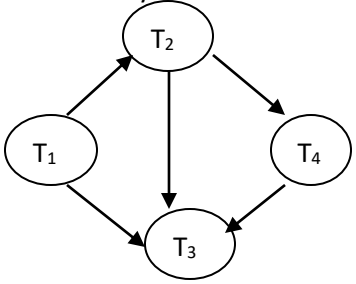
  - **Wound – wait:**

    This is a preemptive technique when a transaction T_1 requests data item currently held by T_2, T_1 is allowed to wait only if it has a timestamp larger then T_2, else T_2 is rolled back.

    Example: if T_1, T_2 and T_3 have timestamp 5, 10 and 15 min
    - If T_1 request data held by T_2, T_2 will be rolled back.
    - If T_3 request data held by T_2, then will wait.

Deadlocks determine:

- Deadlock can be described interns of a directed graph called wait for graph, constants of G = (V,E) where V is a set of vertices and E is a set of edges.
- Each element in the set E of edges is an ordered points $T_1 \rightarrow T_2$
- If $T_1 \rightarrow T_2$ is in E, then there is a directed edge from $T_1$to T_2, implying that T_1 is waiting for T_2 to release a data item that it needs.
- The system is in a deadlock state if the wait for graph contains a cycle.
- Each transaction involved in the cycle is called deadlock.



- This graph contains a cycle, so a system in deadlock state and transaction T_2, T_3 and T_4 are all deadlock.
- **Recovery form Deadlock:**
  - When a deadlock exists, the system must recover from this.
  - This common solution is to rollback one or more transactions to break the deadlock.
    - **Selection of a victim:**
      Determine those transactions that will in the minimum cost should rollback.
    - **Rollback:**
      Once decide that a particular transaction must be rolled back, we must determine how for this transaction should be rolled back.
      - The simple solution is total rollback that means abort the transaction and then restart it.
      - Break the deadlock is a more effective to rollback the transaction.
    - **Starvation:**
      It happens if same transaction is always chosen as victim. The solution is to include the no. of rollback in cast factor.

Object – oriented database (OODB):

- OODB are extensible which identifies new data types and the operations to be performed on them.
- It provides encapsulation, which the data representation and the methods implementation are hidden from external entities.
- It provides inheritance properties in which an object inherits properties of other objects.
- Object database should be used when there is complex data and/or relationship.
- It includes many to many object relationships.
- Object database are applicable for
  - CASE applications, CAD, CAN
  - Multimedia applications.
  - E – commerce
  - Object projects that change overtime
- **Advantages over RDBMS:**
  - Reduced paging.
  - Easier navigation.
  - Better concurrency control a hierarchy of objects may be.
  - Data model is based on the real world.
  - Less code required when applications are object oriented.
- **Disadvantages compared to RDBMS:**
  - Lower efficiency when data is simple and relationship are simple.
  - Relational tables are simpler.
  - More user tools exist for RDBMS.
  - Late binding may slow access speed.
- **Advantages of OODB:**
  - It allows for storages of complex data structures that cannot be easily stored using conventional database technology.
  - OODB support all the persistence required for object – oriented applications.
  - Reusability.
  - Complex data types such as documents, graphics, images etc.
  - Distributed database can support distribution of data occurs networks more easily.

- **Parallel database:**
  - In parallel database systems, multiple CPUs work in parallel to improve performance through parallel implementation of various operations such as loading data, building indexes and evaluating queries.
  - Parallel processing divides a large task into many smaller tasks and executes the smaller tasks concurrently on several nodes that results larger tasks completes more quickly.
  - It improves processing I/O speeds by using multiple CPUs and disks working in parallel.
  - In parallel processing, many operations are performance simultaneously.
- **Advantages:**
  - Increased through put.
  - Improved response time.
  - Increased availability of system.
  - Greater flexibility.
  - Possible to serve large no. of users.
- **Disadvantages:**
  - More startup cost.
  - Inference problem.
  - Skew problem.

Distributed database:

- It consists of a collection of sites, connected together through the some kind of communication network.
- Each site is a database system site in its own right and a user at any site can access anywhere in the network if data were stored at local site.
- A distributed database is fragmented into smaller data sets based on usage.
- DDBMS can handle both local and global transaction.
- Distributed database are of 2 types
  - **Homogeneous Distributed Database:**
    In this, all sites have identical DBMS S/W are aware of one another and agree to co – operate in processing user's requests.
  - **Heterogeneous Distributed Database:**
    - In this, different sites may use different schemas and different DBMS S/W.
    - The sites may not be aware of one another and they may provide only limited facilities for co – operator in transaction processing.
  - **Advantages of distributed database**:
    - New sites can be added to the system any time with no or little efforts.
    - It allows each site to store and maintain its own database causing immediate and efficient access to data.
    - It allows accessing the data stored at remote sites.
    - If a user needs to access the data from multiple sites then the desired query can be subdivided into sub – queries and the parts evaluated in parallel.
  - **Disadvantages of DDBMS:**
    - Complex S/W is required for a distributed database environment.
    - The various sites must exchanges message and perform additional calculations to ensure proper co – ordination among the site.

- If data are not distributed properly according to theirs usage response to requests for data can be extremely slow.
- **Data warehouse:**
  - It is the process of gathering information from different parts of a business process in a centralized database.
  - It is an integral part of any organization.
  - It is implemented in order to support decision making in an organization.
  - It is a data structure that is optimized for distributed.
  - It collects and stores integrated sets of historical data from multiple operational systems and feeds them to one or more data marts.
- **Properties of data ware house:**
  Data warehouse has the specific characteristic
  - Subject - oriented:
    Information is present according to specific subjects.
  - Integrated:
    A single source of information for and about understanding multiple areas of interest. It provides one- stop shaping and contains information about a variety of subjects.
  - Non – volatile:
    Stable information that doesn't change each time an operational process is executed.
  - Time – variant:
    Containing a history of the subject as well as current information.
  - Process oriented:
    Data warehousing is a process for delivery of information.
  - Accessible:
    It provide readily accessible for delivery of information.
- **OLAP (Online analytical processing):**
  - It is a method by which multidimensional analysis occurs.
  - OLAP is the technology that enables client application to efficiently access this data.
  - It provides a lot of benefits to analytical users.
    - ♦ An initiative multidimensional data model makes it easy to select, navigate and explore the data.
    - ♦ An analytical query language provides power to explore complex business data relationships.
    - ♦ Pre – calculation of frequently queried data enables very fast response time to adhac queries.
- **Data mart:**
  - It is a data structure that is optimized for access.
  - It is designed to facilitate end user analysis of data.
  - It support a single, analytical application used by a distinct set of workers.
  - It is a special form of data warehouse containing a topic oriented subset of data appropriate to specific business function.
- **Merits:**
  - It facility data reporting.
  - It makes data design simpler.
  - It ensures security also.
  - It focuses on the presentation instead of organization of data.

- **OLTP (Online Transaction Processing):**
  - Support large no. of concurrent users who are actively adding and modifying data.
  - Have complex structure.
  - Provide the technology infrastructure to support the day to day operations of an organization.
  - Contain large amounts of data, including extensive data used to verify transactions.
  - Are tuned to be responsive to transaction activity.

❖ **Data mining:**

- It is the technique of extraction of hidden predictive information from large database.
- It is a process of analyzing large volume of data and discovering useful knowledge and patterns from data.
- The knowledge discovered from the database can be represented by a set of rules.
- The main goals of data mining are.
  - Portion:
    It allows predicting certain information based on values of certain attributes in the database.
  - Classification:
    Data mining partition items into different classes, the classification can be done by finding rules that partition the given data into different groups.

RAID : There are various combinations of these approaches giving different trade offs of protection against data loss, capacity, and speed. RAID levels 0, 1, and 5 are the most commonly found, and cover most requirements.

■ RAID 0 (striped disks) distributes data across several disks in a way that gives improved speed and full capacity, but all data on all disks will be lost if any one disk fails.

■ RAID 1 (mirrored disks) could be described as a backup solution, using two (possibly more) disks that each store the same data so that data is not lost as long as one disk survives. Total capacity of the array is just the capacity of a single disk. The failure of one drive, in the event of a hardware or software malfunction, does not increase the chance of a failure nor decrease the reliability of the remaining drives (second, third, etc).

■ RAID 2 It uses memory style redundancy by using Hamming codes, which contain parity bits for distinct overlapping subsets of components.

■ RAID 3 It uses a single parity disk relying on the disk controller to figure out which disk has failed.

RAID 4 It uses block-level data striping.

RAID 5 (striped disks with parity) combines three or more disks in a way that protects data against loss of any one disk; the storage capacity of the array is reduced by one disk.

■ RAID 6 (less common) can recover from the loss of two disks through P + Q
redundancy scheme using Reed- Adoman codes