KiiT

KIIT POLYTECHNIC

LECTURE NOTES

ON

DATABASE MANAGEMENT SYSTEM

Compiled by

Swagatika Dalai

Lecturer, Department of Computer Science and Engineering,
KIIT Polytechnic, Bhubaneswar

CONTENTS

Chapter-1

BASIC CONCPETS OF DBMS

DATA : Data is a raw fact or figures of any object through which an object cannot be identified.
For example: Emp_No, Name, Salary of an employee individually known as one data.

INFORMATION: Organized Collection of related data is known as information through which an object can clearly identified OR the processed data is called information.
For example: Emp_No, Name, Salary of an employee combined together, is known as information.

DATA FILE: Organized Collection of related information is known as data file.
For example: Sales dept, purchase dept, Administrative dept of a particular company maintains their data file.

DATA BASE: Organized collection of related data files or information is called database.
OR
A database is an integrated collection of logically related records and files. OR A database can be defined as a collection of coherent, meaningful data.

For example: Qr_No, Road name, Area name, Dist name, State name, pin code Combined together creates a postal address. The multiple addresses kept together in one place, such as an address book is a database and the postal address in the book is the data that fills the data base.

DBMS: It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

DATABASE SYSTEM: The database and DBMS software together is called as Database system.

NOTE: Though data is a raw fact through which an object cannot be identified, but it's the bare minimum requirement to build a database.

Applications of DBMS:

Databases are widely used. Some of them are as follows.
- ❖ **Banking**-> For customer information, accounts, loans and banking transactions.
- ❖ **Airlines**-> For reservation and scheduled information.
- ❖ **Universities**-> For student, course and grade information.
- ❖ **Credit card transactions**-> For purposes on credit cards and generation of monthly statements.

❖ **Telecommunications**-> For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards and storing information about the communication network.

❖ **Finance**-> For storing information about holdings, sales and purchases of financial instruments such as stocks and bonds.

❖ **Manufacturing**-> For management of supply chain and for tracking production of items in factories, inventory of items in warehouses and orders for items.

❖ **Human resources**-> For information about employees, salaries, payroll taxes& benefits and for generation of payments.

ADVANTAGE OF DBMS/ PURPOSE OF DATABASE SYSTEMS:-

1. **Reduction of redundancy & inconsistency:** Unlike file processing system in database approach data can be stored at a single place or Centralized control of data by the Database administrator. It avoids unnecessary duplication of data which saves space and does not permit inconsistency.

2. **Shared data:-** A database allows the sharing of data under its control by any number of application programs or users.

3. **Data independence**: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to protect application code from such details.

4. **Integrity can be improved:** Data integrity refers to validity and consistency. Data integrity means data should be accurate & consistent. This is done by providing some checks & constraints. These are the consistency rules that the database is not permitted to violate. For Example, the age of an employee can be between 18 to 58 years only. While entering the data for the age of an employee, the database should check this.

5. **Supports Concurrent access**: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Example: Two people reading a balance and updating it at the same time and gives accurate result.

6. **Improved Backup & Recovery:** DBMS provides facilities for recovering the hardware & software failures. A backup & recovery subsystem is responsible for this. In case a program fails, it restores the database to a state in which it was before the execution of program.

7. **Improved Security:** DBMS can enforce access controls that govern what data is visible to different classes of users. For Example, salary of the employees should not be visible anyone other than the department dealing in this.

Disadvantages of DBMS

Although there are many advantages of DBMS, the DBMS may also have some minor disadvantages. These are:

1. **Cost of Hardware & Software**: A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also very costly.

2. **Cost of Data Conversion:** When a computer file-based system is replaced with a database system, the data stored into data file must be converted to database file. It is very difficult and costly method to convert data of data files into database.

3. **Cost of Staff Training:** Most DBMSs are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to be paid a lot of amount for the training of staff to run the DBMS.

4. **Appointing Technical Staff:** The trained technical persons such as database administrator, application programmers, data entry operators etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the" system cost increases".

5. **D**atabas**e Damage:** In most of the organizations, all data is integrated into a single database. If database is damaged due to electric failure or database is corrupted on the storage media, then your valuable data may be lost forever.

DBA

Database Administrator (DBA): This is the chief administrator, who oversees and manages the database system (including the data and software). Duties include authorizing users to access the database, coordinating/monitoring its use, acquiring hardware/software for upgrades, etc. In large organizations, the DBA might have a support staff.

ROLE OF A DBA
Defining the Schema:
The DBA defines the schema which contains the structure of the data in the application. The DBA determines what data needs to be present in the system how this data has to be represented and organized.
Defining Security & Integrity Checks:
The DBA finds about the access restrictions to be defined and defines security checks accordingly. Data Integrity checks are also defined by the DBA.
Defining Backup / Recovery Procedures:
The DBA also defines procedures for backup and recovery. Defining backup procedures includes specifying what data is to backed up, the periodicity of taking backups and also the medium and storage place for the backup data.
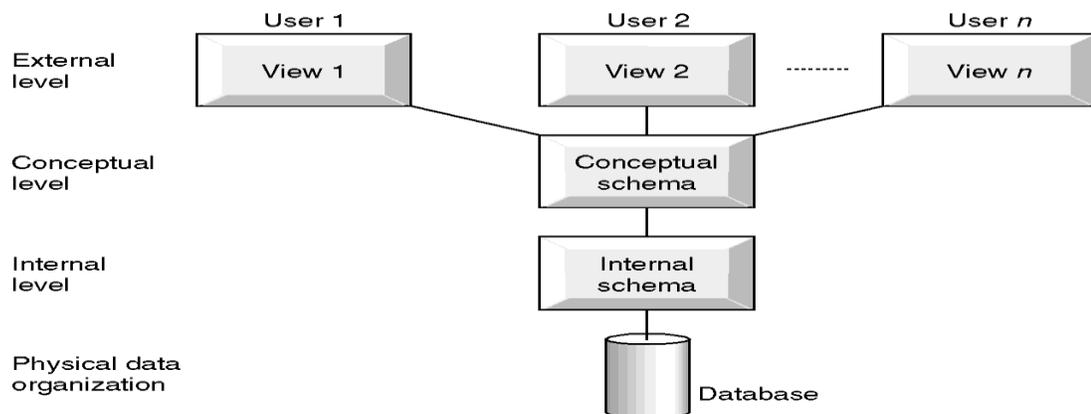**Monitoring Performance:**
The DBA has to continuously monitor the performance of the queries and take measures to optimize all the queries in the application.

THREE LEVEL ARCHITECTURE OF DBMS / LOGICAL DBMS ARCHITECTURE / LEVEL OF ABSTRACTION
- A database system is a collection of interrelated data and set of programs that allows users to access and modify these data.

- The major purpose database system is to provide users with an abstract view of data (i.e. the system hides certain details of data how data are stored and maintained).
- The logical architecture describes how data in the database is perceived by users. It is not concerned with how data is handled and processed by DBMS, but only with how it looks.
- The system is called ANSI/SPARC MODEL and system is divided in to three levels of abstraction: the internal or physical level, conceptual or logical level, external or view level.
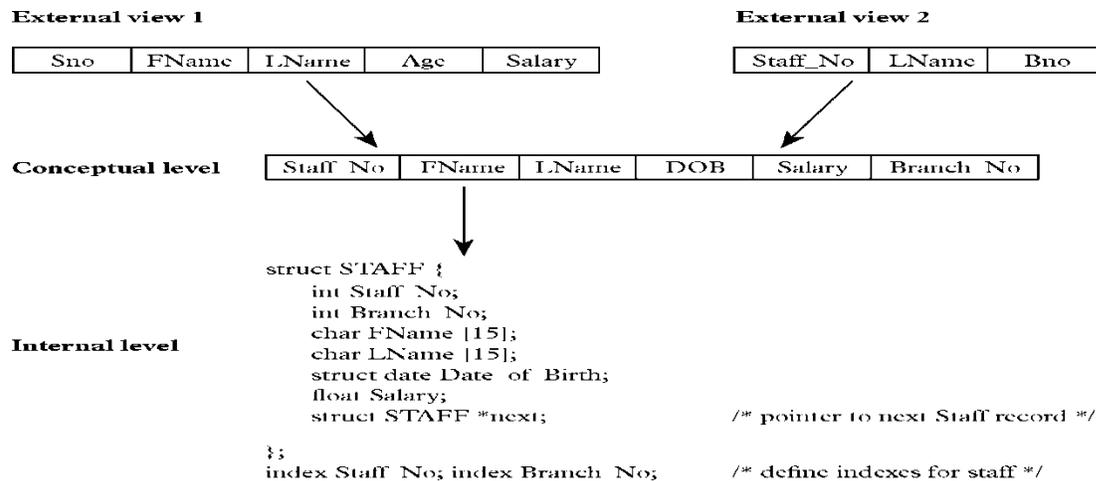

1. Physical level / Internal level:
- This level of abstraction is concerned with physical storage of information.
- It provides the internal view of the actual physical storage of data by telling how the data is stored.
- This level defines various stored data types.
- Here the complex low level data structures are described in details.

2. Conceptual/ Logical level:
- The next higher level of abstraction is conceptual level which describes what data are stored and relationship among them.
- Here the entire data base describes in terms of a small number of relatively simple structures.
- This level of abstraction is used by the database administrator, who must decide what information is kept in the database.
- Only one logical view is present.

3. External/view level:
- This is the highest level of abstraction which describes only part of entire data base.(i.e. Describes that part of database that is relevant to a particular user.)
- Many users of database system will not be concerned in all information. The system may provide different views for different users.
- This level is closest to the user.
- The view level of abstraction exists to simplify their interaction with the system.

Differences between Three Levels of ANSI-SPARC Architecture:

Schemas:

- The overall design of the database is called database schema or database scheme.
- In other words it is the logical structure of the database.
- It is analogous to type information (i.e. data type) of a variable in a program.
- Database system has several schemas, partitioned according to level of abstraction. They are:

1. Physical schema: The physical schema describes the database design at the physical level, which is the lowest level of abstraction describing how the data are actually stored.

2. Logical schema: The logical schema describes the database design at the logical level, which describes what data are stored in the database and what relationship exists

3. subschemas: The several schemas present at the view level to describe different views of the database.

Instances:

- The collection of information stored in a database at a particular point in time is called an instance of the database.
- It is analogous to the value of a variable.

DATA DICTIONARY

- The relational database system needs to maintain data about relation. This information is called data dictionary.
- In other word, a data dictionary is a set of metadata (i.e. data about data) which contains the definition and representation of data elements.

- The data dictionary is created when database is created.
- Then after, whenever the database is in operation data dictionary is updated automatically by DBMS.
- The data dictionary is a reference for all database users.

The contents of dictionary are:
- Names of relations, time of its creation and when it was last accessed.
- The names and data types of its attributes of each relation.
- Domains of the attributes.
- Where the data is stored.
- Names of views defined on the database and definitions of those views.
- Constraints applied to tables.
- Names of authorized user and the rights and privileges they have granted.

DATABASE USER

- There are different types of database system users differentiated by the way they expect to interact with the system.

  Naive Users:-
- These are end users who interact with the system by invoking permanent application program that have been written previously. They need not be aware about the application program they only use it.
- Ex:- ATM users

  Application Programmer:-
- They are the computer professional who develop the application program. The application programs could be written in a general purpose programming language such as PASCAL, COBOL, C, C++.

  Sophisticated User:-
- These users interact with the system without writing the program. They form the request by writing queries in database query language.
- Analysts who submit queries to explore data in the database fall in this category.

  Specialized User:-
- Users who are responsible to write specialized database applications that do not fit into the conventional data processing system. For ex. Computer Aided Design (CAD), Artificial Intelligence System etc.

Chapter-2
DATA MODELS

DATA INDEPENDENCE: The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called data independence.

<div align="center">OR</div>

Data and programs are independent of each other, so change is one has no or minimum effect on the other. Data and its structure are stored in the database where as application programs manipulating this data are stored separately, the change in one does not necessarily affect the other.

These are of 2 types:

1.Logical data independence: The ability to modify the conceptual schema without changing the external schema or application programs to be rewritten.

Example: The addition or removal of new entities attributes or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

2. Physical data independence: The ability to modify the internal schema without having to change the conceptual schemas.

Modifications at this level are usually to improve performance.
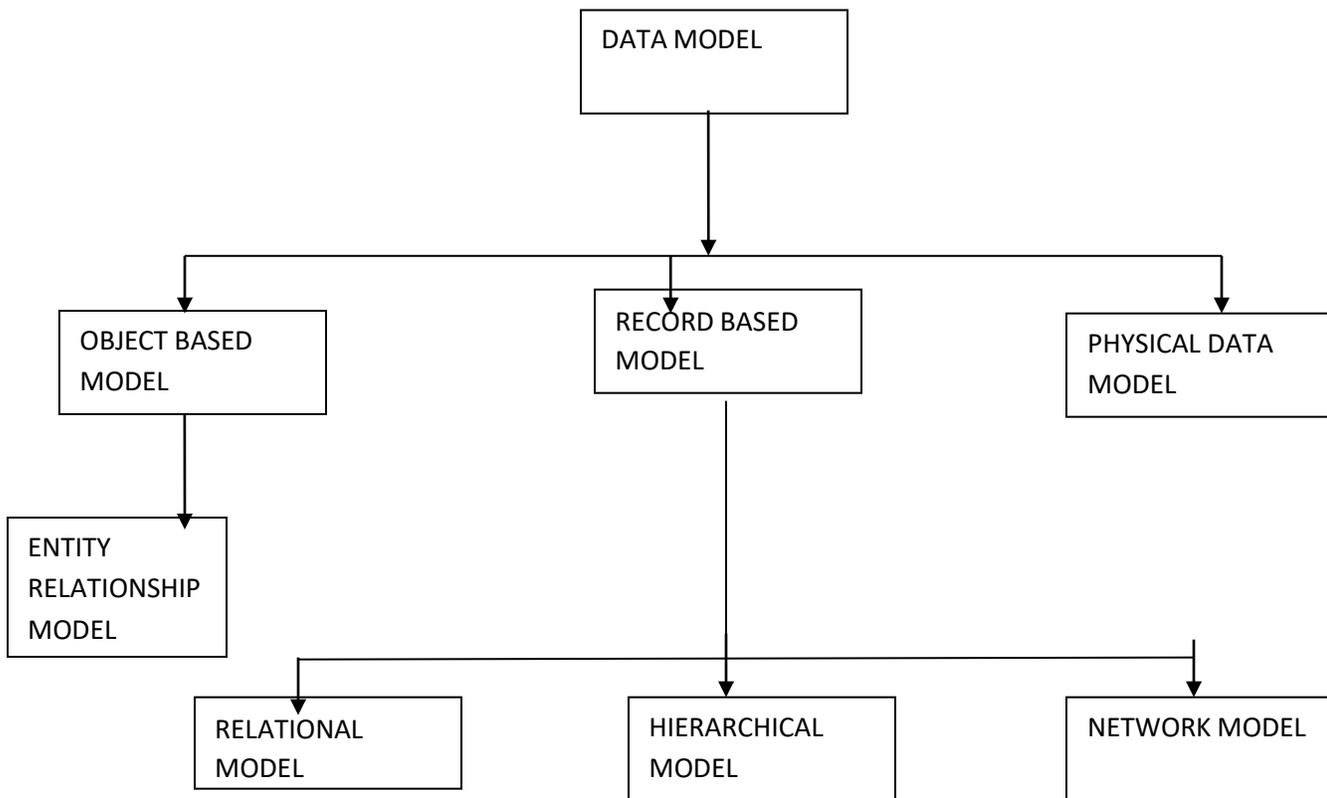
Changes in the physical schema may include:

- Using new storage device
- Using different data structures
- Switching from one access method to another, modifying indexes etc.

Data model

Data models are the collection of conceptual tools which describes data, their relationship, their schematic and consistency constraints.

In other word we can say that a database model defines the logical data structure & the data relationships.

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.

```
                    ┌─────────────┐
                    │ DATA MODEL  │
                    └─────────────┘
                           │
        ┌──────────────────┼──────────────────┐
        ▼                  ▼                  ▼
┌──────────────┐   ┌──────────────┐   ┌──────────────┐
│ OBJECT BASED │   │ RECORD BASED │   │ PHYSICAL DATA│
│ MODEL        │   │ MODEL        │   │ MODEL        │
└──────────────┘   └──────────────┘   └──────────────┘
        │                  │
        ▼                  │
┌──────────────┐           │
│ ENTITY       │           │
│ RELATIONSHIP │           │
│ MODEL        │           │
└──────────────┘           │
        ┌──────────────────┼──────────────────┐
        ▼                  ▼                  ▼
┌──────────────┐   ┌──────────────┐   ┌──────────────┐
│ RELATIONAL   │   │ HIERARCHICAL │   │ NETWORK MODEL│
│ MODEL        │   │ MODEL        │   │              │
└──────────────┘   └──────────────┘   └──────────────┘
```

OBJECT BASED MODEL:

This data model uses objects as the key data representation components.

For ex: Entity-Relationship model
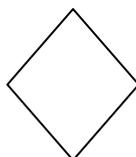
ENTITY-RELATIONSHIP MODEL

- It is a collection of real world objects called entities and their relationships.
- It is mainly represented in graphical form using E-R diagrams.
- This model is very useful in database design.
- E-R diagram which consists of following components.

☐ : - Represents Entity

⬭ :-Represents Attributes

◇ : Represents Relationship among entity set

_____ :-Represents link between entity set and entity set to relationship set.

Entity:

An entity is an object of concern to represent the things in real world and is distinguishable from other by certain property.e.g. car, table,book etc.

*An entity need not be physical entity, it can also be represents a concept in real world.e.g. project, loan etc.

*It represents a class of things, not any instance.

e.g. (student) entity has instances of "Ramesh and Mohan".

Entity Set/Entity Type:

A collection of a similar kind of entities is called an entity set or entity type.
It is of two types.
1. Weak entity set: Weak entity set does not contain any key attribute (i.e. primary key).

It is denoted by

2. Strong entity set: The entity type containing a key attribute are called strong entity or regular entity set.

It is denoted by

e.g.- The 'STUDENT' entity has a key attitude ROLL NO which uniquely identifies it.

Attribute:

An attribute is a property used to describe the specific feature of entity. So to describe an entity entirely, a set of attributes is used.

Types Of Attribute:
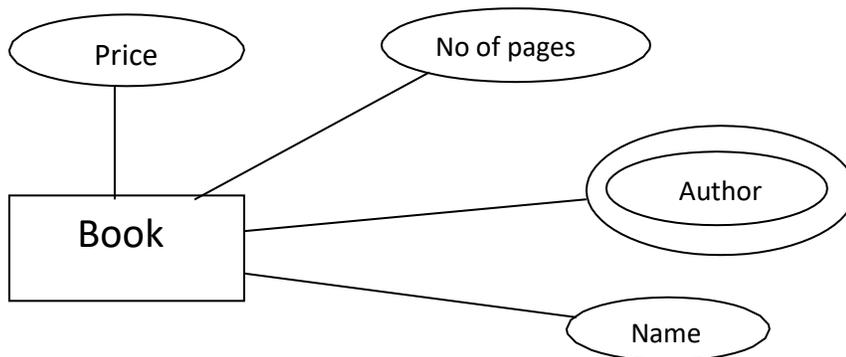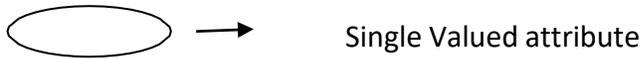
Attribute are of following type:
1. Single valued and multi valued
2. Simple and composite
3. Store and derived
4. Null value

Single valued and multi valued attribute:

The attributes having a single value for a particular entity is called single valued.
Ex: age of a person is a single-values attribute.

Attributes that have more than one value for a particular entity is called a multivalued attribute .e.g. book entity have attributes name, author, no of pages. Here author is the multivalued attribute as a single book has more than one author & others are single valued.
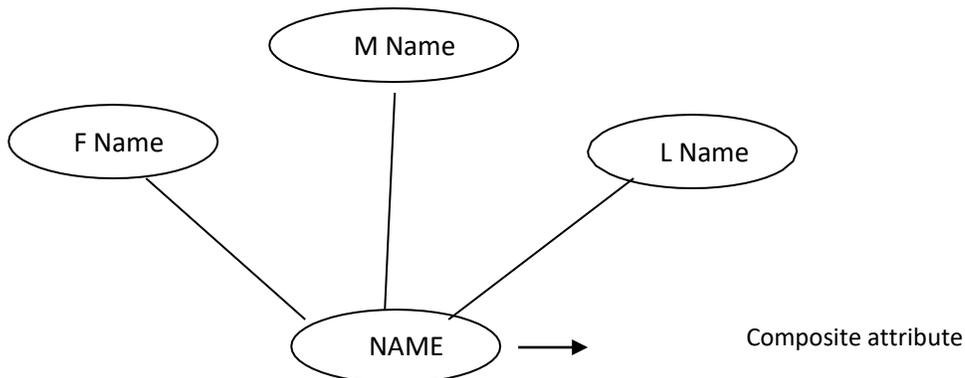
Single Valued attribute

Multivalued attribute

Price

No of pages

Book

Author

Name

Simple and composite Attribute
The attribute that cannot be further sub-divided into smaller parts and represents the basic meaning is called simple attribute.
e.g.'age'.'Roll no' etc.

Simple attribute

*Composite attributes are attributes that can be further divided into smaller units and each individual unit contains a specific meaning. .e.g. The 'NAME 'attribute of an employee entity can be subdivided into 'First name', middle name', 'last name'.

M Name

F Name

L Name

NAME

Composite attribute

STORED AND DERIVED ATTEIBUTE

Attributes that are directly stored in the database are called stored attribute.

e.g. 'Date of birth' of a person.

Stored attribute

An attribute whose value is derived from the stored attribute is known as derived attribute.

Example: age, and it's value is derived from the stored attribute Date of Birth.

Derived attribute

NULL VALUED ATTRIBUTE

*The attribute which has no value.

RELATIONSHIP

A relationship can be defined as a connection or set of association, or a rule for communication among entities.

e.g. in college data base, the association between student and course entity. i.e." student opts course "is an example of relationship.

RELATIONSHIP CARDINALITY/MAPPING CONSTRAINT
*Cardinality specifies the no of instances of an entity associated with another entity, participating in a relationship.
*Based on the cardinality binary relationship can be further classified into the following categories.

    a>one----to---one
    b>one---to---many
    c>many---to---one
    d>many---to---many

one----to---one
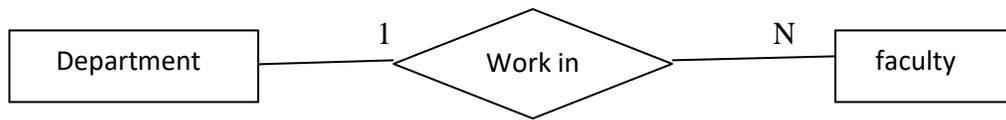Only one instance of entity 'A' associated with at most one instant of entity B.
e.g.: Relationship between college and principal.

College — Has — Principal

One college can have at most one principal and one principal can be assigned to only one college.

one---to---many

e.g. Relation between department and faculty.

```
┌──────────────┐  1  ╱────────────╲  N  ┌──────────────┐
│  Department  │─────│   Work in   │─────│   faculty    │
└──────────────┘     ╲────────────╱     └──────────────┘
```

One department can appoint any no. of faculty but a faculty is assigned to one department.

e.g. Relationship between HUB & COMPUTER.

```
                                        ┌──────────────┐
                                        │   COMPUTER   │
                                        └──────────────┘
┌──────────┐  1  ╱──────────╲
│   HUB    │─────│  Connect  │
└──────────┘     ╲──────────╱
```
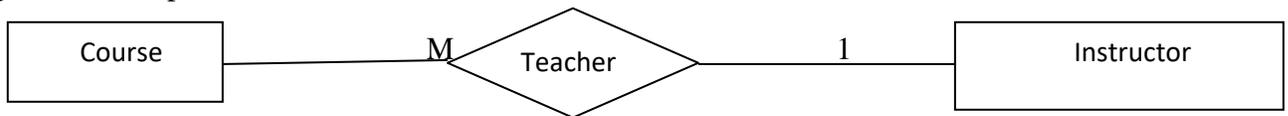
One HUB can connect to any no of COMPUTER but a computer can connect to any one HUB.

many---to---one

Any one of instance of entity A are associated with only one instance of entity B.

E.g. Relationship between course and instructor.

```
┌──────────────┐  M  ╱────────────╲  1  ┌──────────────┐
│    Course    │─────│   Teacher   │─────│  Instructor  │
└──────────────┘     ╲────────────╱     └──────────────┘
```

An instructor can teach various courses but a course can be taught only by one instructor.

many---to---many

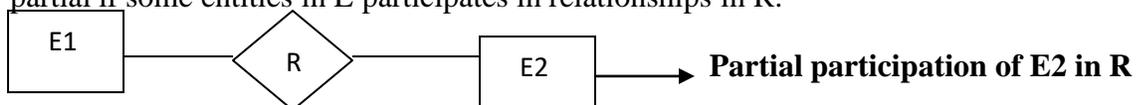Instances of entities A & B are associated with any number of instances from each other.

e.g. Relationship between book & author.

```
┌──────────────┐  M  ╱────────────╲  N  ┌──────────────┐
│    BOOK      │─────│   Writes    │─────│    Author    │
└──────────────┘     ╲────────────╱     └──────────────┘
```
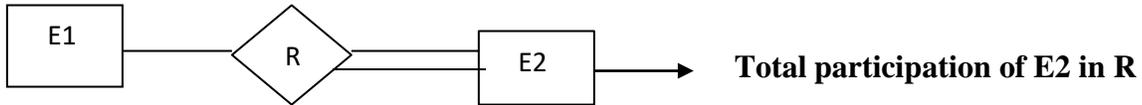
One author can write many books and one book can be written by more than one authors.

One author can write many books and one book can be written by more than one authors.

Partial participation: The participation of an entity set E in a relationship set R is said to be partial if some entities in E participates in relationships in R.

```
┌──────┐        ╱───╲        ┌──────┐
│  E1  │────────│ R │────────│  E2  │───────▶  Partial participation of E2 in R
└──────┘        ╲───╱        └──────┘
```
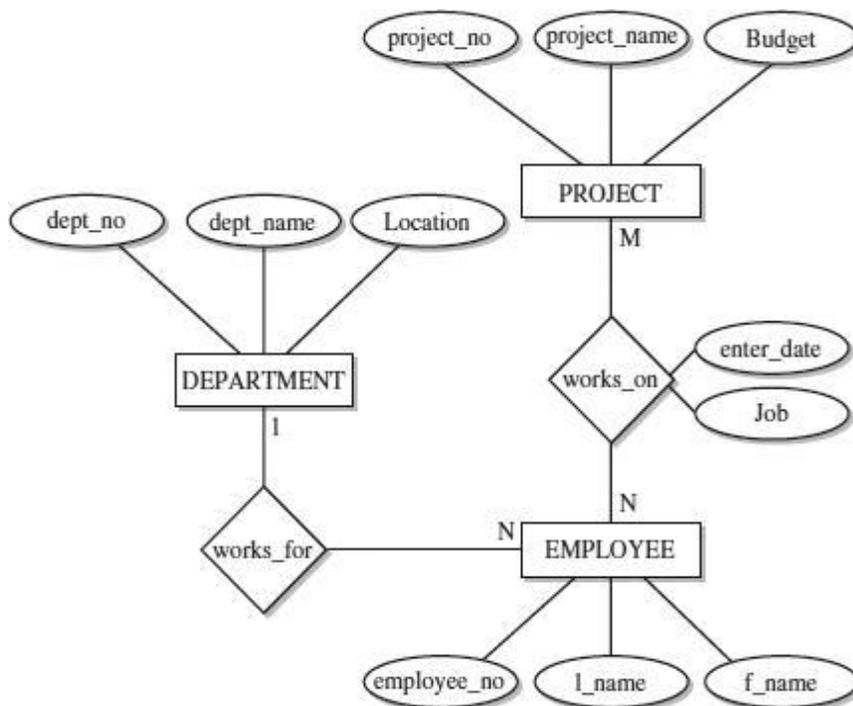
Total participation: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.
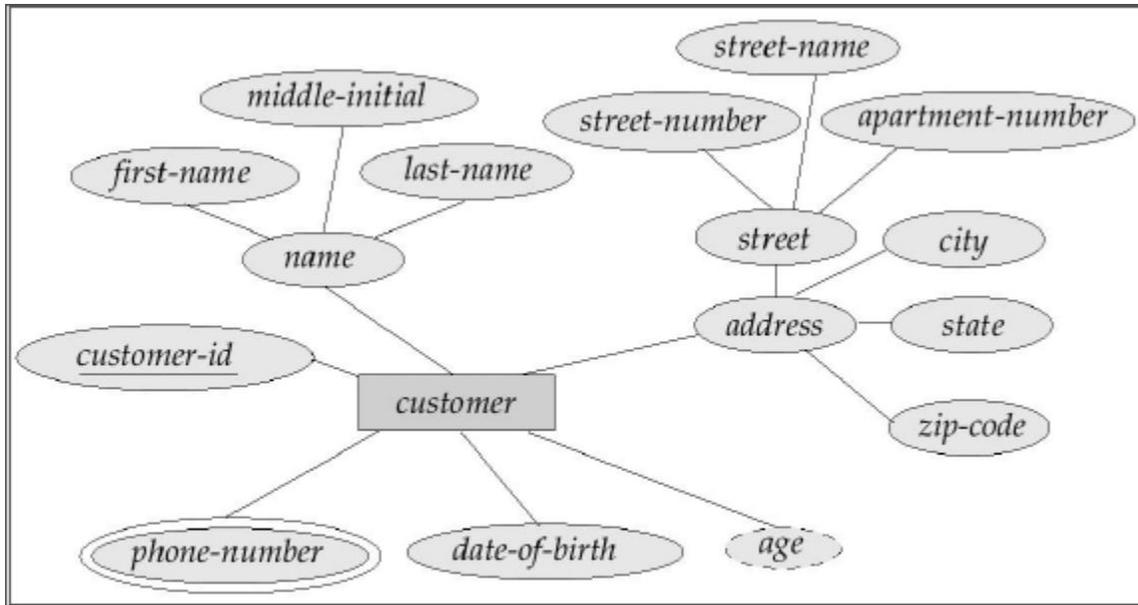

Total participation of E2 in R

E---R DIAGRAM (Entity-Relationship Diagram)

The overall logical structure of database can be expressed using E-R model graphically with the help of an E-R diagram.

Example:

RECORD BASED MODEL

A record based data model is used to specify the overall logical structure of the database.

Each record type defines a fixed no. of fields having a fixed length.
This model describes data at the conceptual level or view level and provides higher level of description of the implementation.
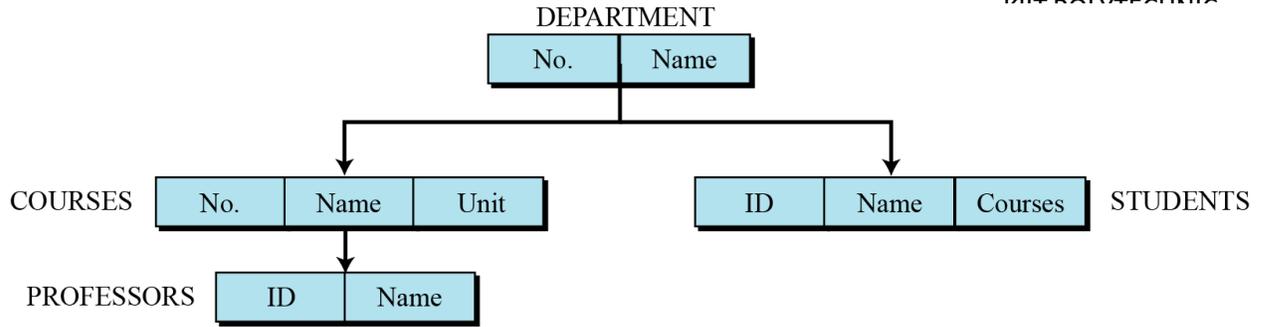This data model uses records as key data representation components.

There are many record based data model
  1) Relational Model
  2) Hierarchical Model
  3) Network Model

HIERARCHICAL MODEL
- The hierarchical data model organizes data in a tree structure .
- Each entity has only one parent but can have several children. At the top of the hierarchy, there is one entity, which is called the root.
- Data and relationships among them is represented by records and pointers.
- This model efficiently describes many real-world relationships like index of a book, recipes etc.

No.	Name

COURSES

No.	Name	Unit

ID	Name	Courses

STUDENTS

PROFESSORS

ID	Name

ADVANTAGES
- Adding and deleting records is easy.
- Fast data retrieval through higher level records.
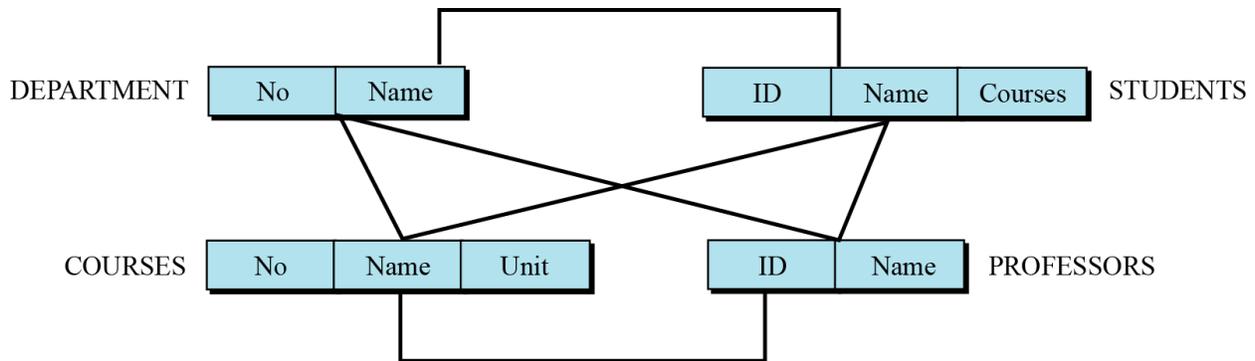- Efficient for 1 to many relationship fixed over time.

DISADVANTAGES
Management difficulties caused by parent segment deletion in that all child segments must also be deleted.

Pointers requires large amount of computer storage.

NETWORK MODEL

In the network model, the entities are organized in a graph, in which some entities can be accessed through several paths.

DEPARTMENT

No	Name

ID	Name	Courses

STUDENTS

COURSES

No	Name	Unit

ID	Name

PROFESSORS

The data are represented by collection of records and relationship among the data represented by links.

The network model is an improvement of hierarchical model; here multiple parent child relationships are used.

In network model the relationship as well as the navigation through the data base is predefined at the data base creation time.

ADVANTAGES
Data independence.

It can handle more relationships than hierarchical e.g. a child can have multiple parents.

DISADVANTAGES
Detailed structural knowledge is required.
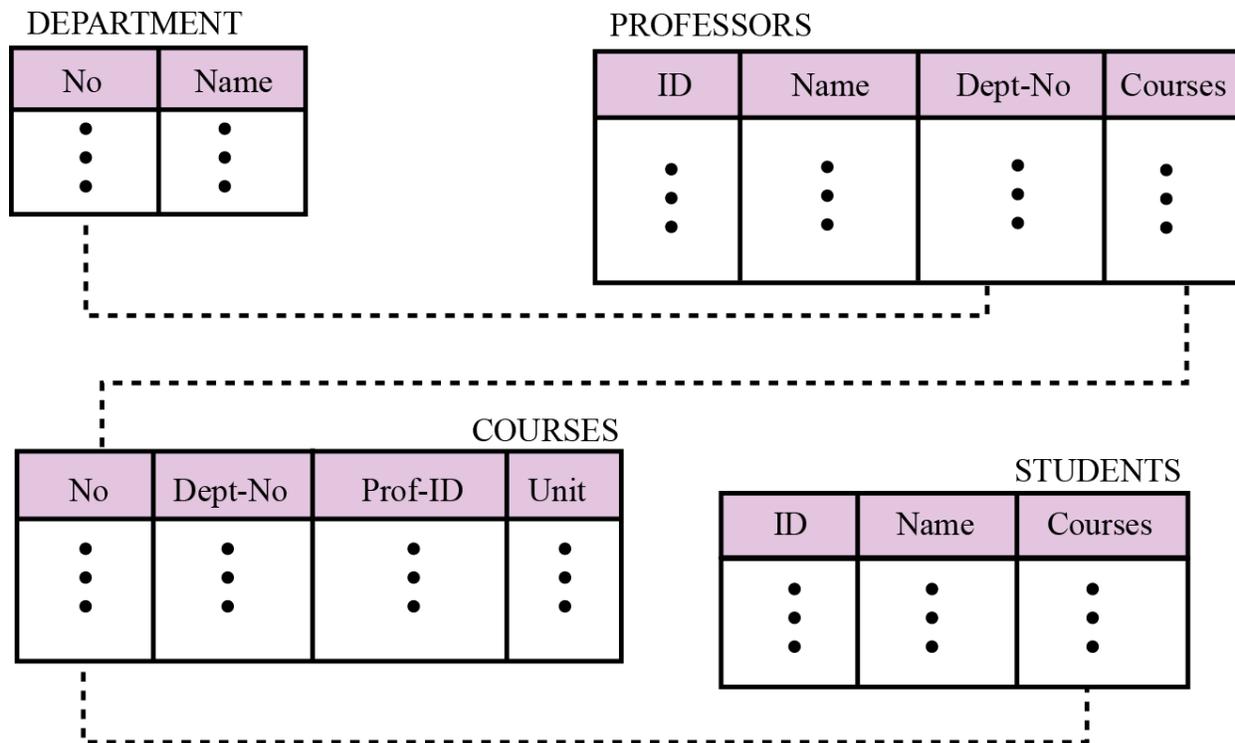Lack of structural independence.

RELATIONAL MODEL

In the relational model, data is organized in two-dimensional tables called relations. The tables or relations are related to each other.
Constraints are stored in a meta-data table.
This is a very simple model and most widely used data base model.
The relational model is based on the relational algebra.

DEPARTMENT

No	Name
⋮	⋮

PROFESSORS

ID	Name	Dept-No	Courses
⋮	⋮	⋮	⋮

COURSES

No	Dept-No	Prof-ID	Unit
⋮	⋮	⋮	⋮

STUDENTS

ID	Name	Courses
⋮	⋮	⋮

Relation: Each table in a database is known as relation.

Attributes: Each column in a relation is called an attribute. The attributes are the column headings in the table .

Domain: The set of all possible value of an attribute is known as domain of the attribute.

Name.: Each relation in a relational database should have
    a name that is unique among other relations.

Tuples. Each row in a relation is called a tuple.

ADVANTAGES
Avoids data duplication.
Avoids inconsistent records.
Easier to change data and data format.
Easier to maintain security.
Data can be added and removed easily.

DISASVANTAGES
New relations can require considerable processing.
Sequential access is slow.
Method of storage on disk impacts processing time.

KEY

A key is an attribute or a set of attribute in a relation that uniquely identifies a tuple in a relation.

Types of keys:

1. **Super Key:** A super key is an attribute or any set of attributes that uniquely identifies a row in a relation.
2. **Composite Key:** A composite key is a key that contains more than one attribute.
3. **Candidate Key**: A candidate key is an attribute or set of attribute that uniquely identifies a row in a relation. In other words minimal super key is called candidate key.
4. **Primary Key**: A primary key is a candidate key which can uniquely identify a record or tuple. Each table can have only one primary key. The primary key should be selected in the manner such that it is unique and not null. .
5. **Alternate Key**: The alternate keys of any table are those candidate keys which are not selected as the primary key.
     AK(Alternate Key)=CK(Candidate Key)-PK(Primary Key)
6. **Overlapping Key**: The keys having common attributes are called overlapping key.
7. **Foreign Key**: Foreign key is a referential key which must be a primary key of another table. This way we can link two tables for retrieving the data jointly. we only insert those values which are present in the base table. If we delete the base table automatically the the foreign key and primary key relationship is broken.

Chapter-3
RELATIONAL DATABASE

Relational Terminology

RELATION
Roughly a table in a relational database is called a relation.
A relation consist of
>   1. Relational scheme
>   2. Relational instances.

Attribute
The name of each column in a database is used to interpret its meaning is called attribute.

Domain
Each attribute is defined over a set of values known as its domain.

Tuple
A tuple is a row in a relation\table.

Degree/Arity
The no. of attributes in the relation scheme is called its degree or arity.

Cardinality
No. of tuples present in a relation scheme is called cardinality of a relation.

Union Compatible
Two relation P[P] and Q[Q] are said to be union compatible, if both P & Q are of same degree and the domains of the corresponding attributes are equal.

i.e. if $P=\{P_1,P_2 ....P_n\}$ & $Q= \{Q_1,Q_2........Q_n\}$ then $DOM(Pi) = DOM(Qi)$ for $i=\{1,2,3,4.......n\}$,

where $DOM(Pi)$ represents the domain of attribute Pi.

RELATIONAL ALGEBRA
Relational algebra is a collection of operation used to manipulate the data in relational model
NOTE: Result of each relation operation is also a relation.
→The operation can be classified in two categories.
1. Basic set operation
  a) Union
  b) Intersection
  c) Set difference
  d) Cartesian product
2. Relational operation
 a) Selection
 b) Projection
c) Join
 c) Division.

BASIC SET OPERATION
→These are binary operations (i.e. each is applied to two relations)
→These two relations should be union compatible expect in case of Cartesian product.

Example

R1

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

$R3 = R1 \cap R2$ is

A	B
A1	B1
A2	B2
A4	B4

UNION

→If R1 & R_2 are two union compatible relations then the union of R_1 & R_2 (R_3= R_1 U R_2) is the relation containing tuples that are either in R_1 & R_2 or in both of them.
→The duplicated tuples such that

R3 = {t | R1 ∈ t ∨ R2 ∈ t} and max ($\mid R1 \mid$, $\mid R2 \mid$) ≤ $\mid R3 \mid$ ≤ ($\mid R1 \mid$ + $\mid R2 \mid$)

→The cardinality of the resultant relation depends on duplication of tuple in R_1 & R_2.
   If the tuples in R_1 & R_2 are disjoints, then $\mid R3 \mid = \mid R1 \mid + \mid R2 \mid$

NOTE
   • Union is a commutative operation i.e. R_1 U R_2 = R_2 U R_1
   • Union is an associative operation i.e. R_1 U (R_2 U R_3) = (R_1 U R_2) U R_3

INTERSECTION

   • If R1 & R2 are two union compatible relation the R=R1 …R2 is the relation that includes all tuples that are in both the relations.

   • In other words R3 will have tuples such that
      R3 = {t | R1 ∈ t A R2 ∈ t}.and 0≤ $\mid R3 \mid$ ≤ min($\mid R1 \mid$, $\mid R2 \mid$)

   Intersection is really unnecessary .it can be simplified R1..R2=R1(R1-R2)

Example

R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R3 = R1 ∪ R2 is

Q

A	B
A1	B1
A2	B2
A3	B3
A4	B4
A7	B7

Note: 1) Intersection is a commutative operation, i.e.,
R1 ∩ R2 = R2 ∩ R1.
2) Intersection is an associative operation, i.e.,
R1 ∩ (R2 ∩ R3) = (R1 ∩ R2) ∩ R3

SET DIFFERENCE
If R1 & R2 are two union compatible relations, The results of R=R1-R2 is the relation that includes only those tuples that are in R1 but not in R2 .

In other words we can say it removes that common tuples from the first relation.

R3 will have tuples such that

R3 = {t | R1 ∈ t A t∉R2}. and 0≤│R│≤│R1│

Note: -1) Difference operation is not commutative,
i.e., R1 – R2 ≠ R2 – R1
2) Difference operation is not associative,
i. e.,R1 — (R2 – R3) ≠ (R1 – R2) – R3

R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R1-R2 =

A	B
A3	B3

R2-R1=

A	B
A7	B7

CARTESIAN PRODUCT

- The Cartesian product of 2 relation is concatenations of tuples belonging to two relations.
- A new resultant relation scheme is created consisting of all possible combinations of tuples that in the both relations.
- R3 = R1 $\times$ R2 where a tuple is given by:
  R3 = {t1 || t2 | R1 ∋ t1 ∧ R2 ∋ t2}.
- R3 is obtained by concatenation each tuple in relation R1 with each tuple in relation R2 here || represents the concatenation operation.
  Degree (R3)=(Degree(R1)+Degree(R2)
- The cardinality of resultant relation is given by |R|=|P|*|Q|

R1

C
C1
C2

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

$R3 = R1 \times R2$ is

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C1
A2	B2	C2
A3	B3	C1
A3	B3	C2
A4	B4	C1
A4	B4	C2

Relational Operations

Select Operation

- The select operation is used to select some specific records from the database based on some criteria.

- This is a unary operation mathematically denoted as σ

Syntax:

σ **<Selection condition> (Relation)**

The Boolean expression is specified in <Select condition> is made of a number of clauses of the form:

<attribute name><comparison operator><constant value> or

<attribute name><comparison operator><attribute name>

Comparison operators in the set { $\leq, \geq, \neq, =, <, <$} apply to the attributes whose domains **are** *ordered value* **like integer.**

Example :

Consider the relation PERSON. If you want to display details of persons having age less than or equal to 30 than the select operation will be used as follows:

σ **AGE <=30 (PERSON)**

The resultant relation will be as follows:

PERSON_ID	NAME	AGE	ADDRESS
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.

Note:

1) Select operation is commutative; i.e.,

σ <condition1> (σ <condition2> (R)) = σ <condition2> (σ <condition1> (R))

Hence, Sequence of select can be applied in any order

2) More than one condition can be applied using Boolean operators AND & OR etc.

PROJECT Operation

The project operation is used to select the records with specified attributes while discarding the others based on some specific criteria.

This is denoted as Π.

Π **List of attribute for project** (**Relation**)

Example :
Consider the relation PERSON. If you want to display only the names of persons then the project operation will be used as follows:

Π Name (PERSON)
The resultant relation will be as follows:

NAME
Sanjay Prasad
Sharad Gupta
Vibhu Datt

Selection+ Projection

For example:
Find those students who live in Bhopal from student relation.

Πsname(σsaddress=" Bhopal" (student));

JOIN:-

The join operator joins two or more relations to form another relation.
The join operator joints two relations on the basis of some comparison operator in meaningful way.

Syntax:- R1 ⋈ R2
   x Θy
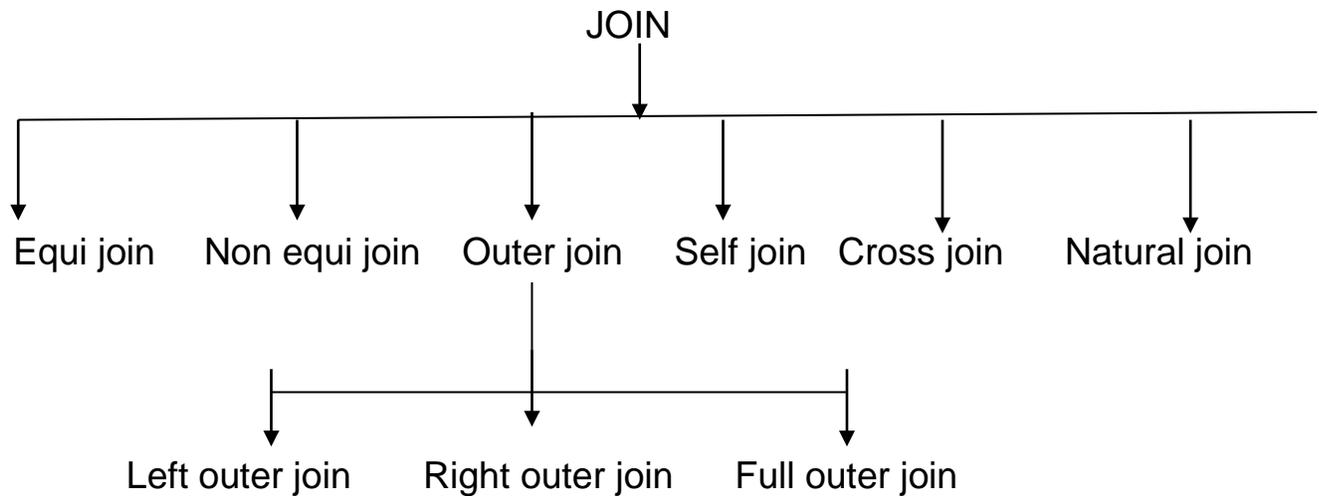Where R1, R2→ Two relations
    X → Attributes of R1
    Y → Attributes of R2
    ⋈ →join operator
    Θ → Comparison operator

1. Equi join:-

- When there is a common attribute in two relations can be joined by equating the common column value.
- When the comparison operator is an equal operator then the join is called equi join.

STUDENT

Name	Roll No	Class
Sita	1	3rd CS
Rama	4	5th CS
Gita	9	3rd ETC
Swayam	10	5th CS
Rabi	14	3rd Elect

HOSTEL

Name	Hostel	Room No
Sita	L-H	1
Rabi	B-H	4
Gita	L-H	2
Ganga	L-H	2

If we join the relation student with relation hostel then the resulting will as follows.

Student ⋈   # Hostel

Student.Name = Hostel.Name

Name	Roll no	Class	Hostel	Room no
Sita	1	3rd CS	L-H	1
Gita	9	3rd ETC	L-H	2
Rabi	14	3rd Elect	B-H	4

2.Non-Equi join:-

       A non-equi join is the join where comparison operator is other than equal operator.
The non equi join takes place where there is no common attribute is present in both relations.

Student

Roll no	Name	Mark
1	XXX	55
2	YYY	35
3	ZZZ	69
4	PPP	25

Grade

Lower	Upper	Division
60	100	1st
50	60	2nd
30	50	3rd
0	30	fail

If we want to retrieve the relation , what is the grade of each student then we have to join the two relation as follows

Student ⋈   Grade

         (Marks between lower and upper)

Roll no	Name	Marks	Division
1	XXX	55	2nd
2	YYY	35	3rd
3	ZZZ	69	1st
4	PPP	25	Fail

3. Outer Join:-

Sometime we do not want to lose any information in the join operation and where this can be possible is called outer join. The use of Outer Join is that it even joins those tuples that do not have matching values in common columns are also included in the result table. Outer join places null values in columns where there is not a match between them.

Outer join is of three types.
A) Left outer join,
B) Right outer join,
c) Full outer join

Student

Name	Roll no	Class
Sita	1	3rd CS
Rama	4	5th CS
Gita	9	3rd ETC
Swayam	22	5th CS
Rabi	10	3rd Elect

Hostel

Name	Hostel	Room no
Sita	LH	1
Gita	LH	2
Rabi	BH	4
Ganga	LH	2

A) Left outer join:-

- In this type of outer join the resulting relation consists of all tuples of the left relations and the tuples of right relation which satisfy the condition.
- If we join the relation student with hostel through left outer join then the result will be as follows:

Student $\bowtie$ Hostel

Student.Name = Hostel.Name

Name	Roll no	Class	Hostel	Room no
Sita	1	3rd CS	LH	1
Gita	9	5th CS	LH	2
Rama	4	3rd ETC	NULL	NULL
Swayam	22	5th CS	NULL	NULL

| Rabi | 10 | 3rd Elect | BH | 4 |

B) Right outer join:-

In this type of outer join the resulting relation consists of all tuples of the right relations and the tuples of left relation which satisfy the condition.

If we join the relation student with hostel through left outer join then the result will be as follows:

Student ⋈   Hostel

      Student.Name = Hostel.Name

Name	Roll no	Class	Hostel	Room no
Sita	1	3rd CS	LH	1
Gita	9	3rd ETC	LH	2
Rabi	10	3rd ELECT	BH	4
Ganga	NULL	NULL	LH	2

C) Full outer join:-

Here all the tuples of both relations are present in the resulting relation. If we join the relation student with hostel through full outer join then the result will be as follows:

Student ⋈   Hostel

      Student.Name = Hostel.Name

Name	Roll no	Class	Hostel	Room no
Sita	1	3rd CS	LH	1
Rama	4	5th CS	NULL	NULL
Gita	9	3rd ETC	LH	2
Swayam	22	5th CS	NULL	NULL
Rabi	10	3rd Elect	BH	4
Ganga	NULL	NULL	LH	2

4. Cross Join -:

This type of join is rarely used as it does not have a join condition, so every row of table 1 is joined to every row of table 2. For example, if both tables contain 100 rows the result will be 10,000 rows.

The size of a Cartesian product is the number of the rows in first table multiplied by the number of rows in the second table. This is sometimes known as a Cartesian product and can be specified in either one of the following ways:

SELECT * FROM table1 CROSS JOIN table2
SELECT * FROM table1, table2

5. **Natural Join**: It is the same like Equi join except one of the duplicate columns is eliminated in the result table. The natural join is the most commonly used form of join operation.

Courses		
CID	**Course**	**Dept**
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD	
Dept	**Head**
CS	Alex
ME	Maya
EE	Mira

Courses ⋈ HoD			
Dept	**CID**	**Course**	**Head**
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Mira

6.**Self-Join**: It is a join operation where a table is joined with itself. Consider the following sample partial data of EMP table:

EMPNO   ENAME   MGRID   …..

1   Nirmal   4

2   Kailash   4

3   Veena   1

4   Boss   NULL

…..   …..   …

The DIVISION operation:

To perform the division operation R1 ÷ R2, R2 should be a proper subset of R1. In the following example R1 contains attributes SNo and PNo and R2 contains only attribute PNo so R2 is a proper subset of R1. If we perform R1 R2 than the resultant relation will contain those values of SNo from R1 that are related to all values of PNo present in R2.

If $R(x) = R1(z) \div R2(y)$

The relation R(x) is having all the tuples t(x) in R1(z) that appears in R1 in combination with every tuple from R2(y).

Where Z= X U Y

R1   R2

SNo	PNo
S1	P1
S1	P3
S1	P2
S2	P1
S1	P4
S2	P2
S4	P1
S4	P4

PNo
P1
P4

The result of R1 ÷ R2 will be

SNo
S1
S4

Note:Degree of relation: Degree (R ÷ S)=Degree of R – Degree of S.

Chapter-4
NORMALIZATION IN RELATIONAL SYSTEM

Database anomalies.

- The real world is model in database through number of relations, since the real world changes over time, the tuples changes throughout.
- Those tuples may be added, deleted or updated and this frequent updation leads to some anomalies (problem). These anomalies arise due to the bad design.

The anomalies are:-

Updated anomalies -

The multiple copies of some data may leads to updates anomalies or inconsistency.
When an update is made and only some of the multiple copies are updated.

Insertion anomalies:-

Sometimes the insertion is depends upon another if the dependent relation values are inserted of inserting to the relation on which it is dependent, then it leads to bad database design.

Deletion anomalies:-

If the relations are dependent with each other then deletion of tuples in one relation should be updated and stored in other relation otherwise it leads to deletion anomalies.

Functional dependency

A functional dependency is a particular relationship between two attributes.

Given a relation R, a set of attributes X in R is said to functionally determine another attribute Y, also in R, (written $X \rightarrow Y$) if and only if each X value is associated with at most one Y value.

Let R be a relation and x & y are two non empty set of attribute in R then a relation instance has a functional; dependency $x \rightarrow y$, if the following conditions holds for every pair of following conditions holds for every pair of tuples of t1 &t2 is R.

$$\text{If } t_1[x]=t_2[x] \text{ then } t_1[y]=t_2[y]$$

$x \rightarrow y$ denote: - x functionally determine y or y functionally dependent on x

- Functional dependency does not imply mathematical dependence that the value of the attributes may be computed from value of another attribute.
- Functional dependent of y on x means that there can be one value of Y for each value of x.

- An attribute may be functionally dependent on two or more attributes rather than a single attribute or vice versa.
- Functional dependencies are consequences of interrelationship among attributes of an entity represented by a relation or due to relationship between the entities i.e. also represented by a relation.

EXAMPLE:-

Emp-id→Ename

Emp-id→Dept, addr

Lets x= {roll no}

y= {name, branch}

x→y

NOTE

The values of all attributes are functionally dependent on primary key of a relation.

INFERENCE RULE OF FDS

For a govern relationship there are many FDS.

let F->set of FDS that are given for a relation R.

There are many other FDS that can be derived from the given FDS and is called as infered.FDS is denoted by F^+.

The set of all dependencies that include F as well as dependencies that can be inferred from F is called closure of F is denoted as F^+.

EXAMPLE:-

F= {Emp -id →ename, addr, dept-no}

Let x= Emp –id, y= dept-no

Dept-no-> {dept-name, mgr no}

x→y

y→z

Then

x→z inferred dependency

F^+={emp-id →dept –name}

to inferred the FDS from F, there are certain rules which are describes below.

RULES(AMSTRONGS AXIOMS)

1)REFLEXIVE

IF (X⊃Y) then automatically x derives y or x→y

2)AUGMENTATION

IF X→Y then xz →yz

3) TRANSITIVE

If(x→y)&(y→z) Then (x→z)

If x determine y & y determine z then automatically x determines z.

4)DECOMPOSITION
IF (X→YZ) then (x→y) & (x→z)
if x determines y & z combinedly then automatically x determines y &x determines z

5)UNION

If (x→y) (x→z) Then(x→yz)

If x determines y &z independently then x can be determines y & z combinedly.

6) PSEUDOTRANSITIVE

If x→y & yz→p then xz→p

Let X=Rollno, y=Name , z= Addr. , p= ph.no

Rollno-→name, name,addr.→Ph.no

Then address, rollno→Ph.no

NOTE

The first three rule (reflexive, augmentation, transitive)are fundamental axioms and are sufficient to generate all possible FDS.

Example

Let R(A,B,C,D,E,F) & the sets of FDS are given below :

$F\rightarrow[\{A\rightarrow BC\},\{B\rightarrow E\},\{CD\rightarrow EF\}]$

Ans:

(1) $A\rightarrow BC$

 $\Rightarrow A\rightarrow B$ & $A\rightarrow C$ *(Decomposition)*

(2) $A\rightarrow B$ & $B\rightarrow E$

 $\Rightarrow A\rightarrow E$ *(Transitive)*

(3) $CD\rightarrow EF$

 $\Rightarrow E$ & $CD\rightarrow F$ *(Decomposition)*

(4) $A\rightarrow BC$ & $A\rightarrow E$

 $\Rightarrow A\rightarrow BCE$ *(Union law)*

(5) $A\rightarrow C$ & $A\rightarrow E$

 $\Rightarrow A\rightarrow CE$ *(Union law)*

$F+=[\{A\rightarrow BC\},\{B\rightarrow E\},\{CD\rightarrow EF\},\{A\rightarrow B\},\{A\rightarrow C\},\{A\rightarrow E\},\{CD\rightarrow E\},\{CD\rightarrow F\},\{A\rightarrow BCE\},\{A\rightarrow CE\}]$

EXAMPLE

 Let R = (A,B,C,D) & F={A$\rightarrow$B,A$\rightarrow$C,BC$\rightarrow$D}. Find F+.

Answer

(1) A$\rightarrow$B & A$\rightarrow$C

 A$\rightarrow$BC(Union)

(2) A$\rightarrow$BC & BC$\rightarrow$D

 A$\rightarrow$D (Transitive rule)

(3) A$\rightarrow$B & A$\rightarrow$D

 A$\rightarrow$BD (Union)

(4) A$\rightarrow$AC & A$\rightarrow$D

 A$\rightarrow$CD (Union)

(5)   A→BC & A→D

A→BCD (union)

F+=[{A→B},{A→C},{BC→D},{A→BC},{A→BC},{A→D},{A→BD},{A→CD},{A→BCD}]

PARTIAL DEPENDANCY:

- A dependency in which one or more non-key attributes are functionally dependant on part but not all of the primary key.

OR

- If primary key is a composite key, some of the attributes will depend on the primary key then those attribute are called partially depends upon the primary key.

EXAMPLE

Stud{s-Id, Name, DOB, Addr, Course, Date of completion}

The FDS are:-

S-id:- Name,DoB,ADDR,Course

S-id:- date of completion

The primary key of the relation is the composite key i.e **s-id,course**, therefore the non-key attributes name, dob, address are functional dependant on only s-id, but not on course. Hence, these are partial dependant.

NORMALIZATION

- ❖ **Normalization** is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like insertion. deletion and modification.
- ❖ The poor database design gives rise to many anomalies. This anomalies can be avoided by a process called Decomposition process or **"Normalization".**

- ❖ The guidelines for constructing relation scheme having desirable property and avoiding anomalies are called normal forms.

- ❖ Normalization is a process of designing the scheme in an appropriate normal from that can allow to retrieve information easily.

Advantages of normalization
- Improve database design.
- Ensures minimum redundancy of data.
- Removes anomalies for database activity.
- Eliminate the problems of flow data querying and using memory space usage.

There are many normal forms based on there of functional dependency:-
1. First normal form (1NF)
2. Second normal form (2NF)
3. Third normal form (3NF)
4. Boyee codd normal form (BCNF)
5. Fourth normal form (4NF)
6. Fifth normal form (5NF)

FIRST NORMAL FORM

A relation schema is in first normal form if the values in the relation instances are   atomic for every attribute in relation.

<div align="center">OR</div>

A table will be in first normal form when each cell in the table contains precisely one value that means multivalued attribute is not allowed.

EX: -

We might have to store data item, colors, price and tax. As every item can have more than one color so, the arrangement might look like this table.

item	colors	price	tax
T-shirt	red, blue	12.00	0.60
polo	red, yellow	12.00	0.60
sweatshirt	blue, black	25.00	1.25

The Table is not in first normal form because Multiple items present in color field.
To make the table in 1NF we should have the data like this.

item	colors	price	tax
T-shirt	red,	12.00	0.60
T-shirt	blue	12.00	0.60
polo	red,	12.00	0.60
polo	yellow	12.00	0.60
sweatshirt	blue,	25.00	1.25
sweatshirt	black	25.00	1.25

SECOND NORMAL FORM (2NF)

A relation schema is in second normal form

If it is in first normal form.

If non-key attribute is not functionally dependant on part of the key attribute (i.e. **no partial dependency**) i.e. if the non key attribute is fully functional dependent of key attribute.

Example :

item	colors	price	tax
T-shirt	red,	12.00	0.60
T-shirt	blue	12.00	0.60
polo	red,	12.00	0.60
polo	yellow	12.00	0.60
sweatshirt	blue,	25.00	1.25
sweatshirt	black	25.00	1.25

Now the table is in 1NF because each field contain single attribute. Here we have taken {item, color} as primary key. However the table is not in 2NF because the non key attribute price and tax depend only on item, but not on color which is a part of the primary key. This violates the rule for 2NF as the rule says no non-key attribute is functionally dependant on part of the key attribute.

To make the table complies with 2NF we can break it in two tables like this:

item	color
T-shirt	red,
T-shirt	blue
polo	red,
polo	yellow
sweatshirt	blue,
sweatshirt	black

Item	price	tax
T-shirt	12.00	0.60
sweatshirt	25.00	1.25
polo	12.00	0.60

Now the table is in 2NF.

3NF (THIRD NORMAL FORM)

A relation R is said to be 3NF.
- If it is in 2NF.
- All non-key attribute are functionally depends on the key attribute(primary key)in other words we can say on non-key attribute of relation R is **not transitively depend** on the key attribute.

Item	price	tax
T-shirt	12.00	0.60
sweatshirt	25.00	1.25
polo	12.00	0.60

In this table **item** is the Primary key. The non key attribute **tax** depends on **price** not on **item.**
Here item→price and price → tax. So item →tax. That means transitive dependeny exists here. Therefore the table is not in 3NF.
To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency.

Item	price
T-shirt	12.00
sweatshirt	25.00
polo	12.00

price	tax
12.00	0.60
25.00	1.25

Now the tables are in 3NF.

BOYCE CODD NORMAL FORM (BCNF)
- It is slightly stronger version of 3NF.

A table is in BCNF if and only if, it is in 3NF and for every functional X->Y, X should be the

super key of the table..
- The BCNF is a special case of 3NF and BCNF can only be violated if a 3NF complaint table contain more than one candidate key.

CHARACTERISTICS
- 3NF (special case)
- Every determinant is a candidate key.

Example: Suppose there is a company wherein employees work in **more than one department**.

They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:

emp_id -> emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
Stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	Stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:

emp_id -> emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:

For first table: emp_id

For second table: emp_dept

For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

Chapter-5
STRUCTURED QUERY LANGUAGE

QUERY LANGUAGE:-
- ❖ A query language is a language in which a user requests information from the database. These languages are usually on a level higher than that of a standard programming language.
- ❖ Query languages can be categorized as either procedural or non-procedural. In a procedural language the user instructs the system to perform a sequence of operations on the database to compute the desired result. In a non-procedural language the user describes the desired information without giving a specific procedure for obtaining that information.

SQL:
- ➢ Structured Query Language (SQL) is a standard query language.
- ➢ It is commonly used with all relational databases for data definition and manipulation.
- ➢ SQL is called a non-procedural language as it just specifies what is to be dome rather than how it is to be done.
- ➢ since SQL is a higher-level query language, it is closer to a language like English.
- ➢ Therefore, it is very user friendly.
- ➢ The American National Standard Institute (ANSI) has designed standard versions of SQL.

Some of the important features of SQL are:
- ➢ It is a non procedural language.
- ➢ It is an English-like language.
- ➢ It can process a single record as well as sets of records at a time.
- ➢ It is different from a third generation language (C& COBOL). All SQL statements define what is to be done rather than how it is to be done.
- ➢ SQL is a data sub-language consisting of three built-in languages
- ➢ Data definition language (DDL), Data manipulation language (DML) and Data Control language (DCL).
- ➢ It insulates the user from the underlying structure and algorithm.
- ➢ SQL has facilities for defining database views, security, integrity constraints, transaction controls, etc.

Data definition language (DDL)

- ➢ The Data definition language (DDL) defines a set of commands used in the creation and modification of schema objects such as tables, indexes, views etc.
- ➢ These commands provide the ability to create, alter and drop these objects.
- ➢ These commands are related to the management and administrations of the databases.
- ➢ Before and after each DDL statement, the current transactions are implicitly committed, that is changes made by these commands are permanently stored in the databases.

CREATE TABLE Command

This command creates a table in the data base.

Syntax:

create table tablename (colname datatype(size), colname datatype (size),…………, colname datatype (size));

Example:-

Create table student(rollno varchar2(20), name varchar2(30), address varchar2(50),semester varchar2(10));

To view the structure of a table:-

Syntax:-

Sql> desc table name;

ALTER TABLE Command:

This command is used for modification of existing structure of the table in the following situation:

➢ When a new column is to be added to the table structure.
➢ When the existing column definition has to be changed, i.e., changing the width of the data type or the data type itself.
➢ When integrity constraints have to be included or dropped.
➢ When a constraint has to be enabled or disabled.

Modifying the structure of a table:-

To add a new column:-

Syntax:-

Sql> alter table table name add (column1 (data type (size)), (column2 (data type (size))) ;

Example:-

Alter table student add (dob date);

To drop a column from a table:-

Syntax:-

Sql> alter table table name drop column name;

Example:-

Alter table student drop dob;

To modify columns from a table:-

Syntax:-

Sql> alter table table name modify (column name (new data type (new size)));

Example:-

Alter table student modify(name varchar2(50));

DROP TABLE Command:

When an existing object is not required for further use, it is always better to eliminate it from the database. To delete the existing object from the database the following command is used.

Syntax:

DROP TABLE<table name>;

Example :

DROP TABLE emp;

Example:-
Create table student(rollno varchar2(20), name varchar2(30), address varchar2(50),semester
varchar2(10));
To view the structure of a table:-
Syntax:-
Sql> desc table name;

DATA MANIPULATION LANGUAGE (DML)

Data manipulation language (DML) defines a set of commands that are used to query and modify
data within existing schema objects.
In this case commit is not implicit that is changes are not permanent till explicitly committed.
DML statements consist of SELECT, INSERT, UPDATE and DELETE statements.
SELECT Statement
This statement is used for retrieving information from the databases. It can be coupled with many
clauses.
Syntax:
SELECT <column_list> FROM <table_name>;

Example:
 SELECT Rollno, name FROM Student;

INSERT INTO command:
Values can be inserted for all columns or for the selected columns
Values can be given through sub query.
If data is not available for all the columns, then the column list must be included following the
table name.
While inserting a single row of data into a table, the insert operation first creates a new row
(empty in the database table) and then loads the value passed into the column specified.

Syntax:-
Insert into table name (col1, col2, col3,………., coln) values (expr1, expr2, expr3,........ ,
exprn);

Example:
Insert into student ('f1201207005','radhika','bbsr','3rd cse');
Example : Insert the employee numbers, an increment amount of Rs.500/- and the increment
date-today (which is being entered through function SYSDATE) of all the managers into a table
INCR (increment due table) from the employee file.
INSERT INTO INCR
SELECT EMPNO, 500, SYSDATE FROM EMP
WHERE JOB = 'MANAGER';
UPDATE Command:
The update command is used to change or modify data value of a table.
Syntax :
UPDATE <table name> Set <column name> = <value>Where <condition>;
Example:
UPDATE student SET addr="etc" Where rollno=10;

DELETE Command

This command is responsible for deleting certain values of a relation those satisfies the conditions.

To remove all the rows from a table the syntax is as follows
Syntax:-
Sql> delete from table name;
Example:-
delete from student;
To remove a set of rows from a table the syntax is as follows
Syntax:-
Sql> delete from table name where <condition>;
Example:-
delete from student where name='radhika';

Example : Delete the records of employees who have got no increment.
DELETE FROM EMP
WHERE EMPNO NOT IN (SELECT EMPNO FROM INCR);

DATA CONTROL LANGUAGE (DCL)

The data control basically refers to commands that allow system and data privileges to be passed to various users. These commands are normally available to database administrator. Let us look into some data control language commands:
Create a new user:
CREATE USER < user name > IDENTIFIED BY < Password>
Example:
CREATE USER MCA12 IDENTIFIED BY W123

1.Grant: This command grant users various privileges to table .It is used to provide database access permission to users. It is used to allow specified user to perform specified task . Privileges can be the combination of select, update, delete, alter. It is of two types
      (1) System level permission (2) Object level permission.
System level permission
 Syntax
      GRANT CREATE SESSION TO USER;
Example:
         GRANT CREATE SESSION TO CSE;

Object level permission

Syntax
      GRANT <previllege_list> ON <table_name> to user;
Example:
      GRANT select On student to CSE;

2.Revoke: It is used to cancel the permission granted.
Syntax :

REVOKE <previllege_list> ON <table_name> FROM user

Example:
REVOKE ALL ON EMP FROM CSE;
(All permissions will be cancelled)
We can also revoke only some of the permissions.

Drop: A user-id can be deleted by using drop command.
Example:DROP USER CSE;

Chapter-6
TRANSACTION PROCESSING CONCEPTS

TRANSACTION

- Transaction is a programming unit of work whose execution may change the database.
- It consists of a set of operation including reading, writing & modifying the database objects.
- Thus a transaction is a program until whose execution preserves the consistency of the database is in a consistent state before a transaction is executed, and then the database is also in a consistent state after execution.
- A **read operation** brings a database objects into main memory from disk where the database is residing & then the value is copied to a program variable defined within the transaction.
- A **write operation** identifies a copy of the data base object in the main memory, updates its value & then writes this objects to the disk.
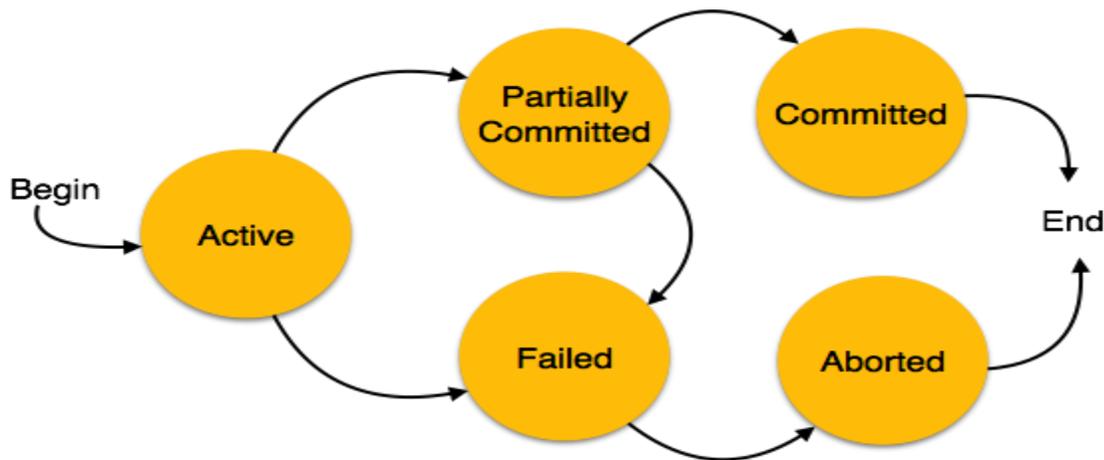
Example:

Ti is a transaction that transfers Rs 50 from account A to account B. This transaction can be defined as:

T1 Schedule
Read (A,a)
a: =a-50;
Write (A,a)
Read(B,b)
b: = b+50;
Write (B,b);

States of Transactions:

A transaction in a database can be in one of the following states −

- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.

- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.

- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −

  i)Re-start the transaction

  ii)Kill the transaction

- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

PROPERTIES OF TRANSACTION (ACID properties)

There are 4 properties of transactions.

1. ATOMICITY :-A transaction is said to be atomic when all the operation of the transaction are either executed at a time or none of them is executed.

2. CONSISTENCY :
- Each transaction must preserve the consistency of data base.
- That is state before the transaction beings its execution & the database remains in a consistent state after the execution of transaction. .

3. ISOLATION :
- A transaction executes in isolation without any interference from any other transaction. It should not make visible to other transaction until it is successfully completed.
- Transaction should be visible till they are committed.

4. DURABILITY:
- Once a transaction is successfully completed its effects should never be lost even if the system crashes before all changes are reflected in the disk.Then effect should never be lost even in subsequent system failure.

SCHEDULE

- A **schedule** is a process of grouping the transactions into one and executing them in a predefined order.
- A schedule is required in a database because when some transactions execute in parallel, they may affect the result of the transaction.
- Hence a schedule is created to execute the transactions.

Types of schedule:

1. Serial schedule: when the actions of different transactions are not interleaved (i.e. transactions are executed from start to finish)then the schedule is called as serial schedule.
It always ensures a consistent state.

2. Non-Serial schedule: A schedule in which the operations from a set of concurrent transactions are interleaved is called as a non-serial schedule.
A non-serial schedule is acceptable if the final state produced by a non-serial schedule is equivalent to some state produced by a serial schedule.

3. Equivalent schedule: Two schedules are said to be equivalent schedules if they produce the same result on any database state.
Equivalent schedules always produce identical results.

4. Serializable schedule: A non-serial schedule that is equivalent to some serial execution of transactions is known as serializable schedule.

5. Complete schedule: A schedule that contains either an abort or commit for each transaction whose actions are listed in it is called as a complete schedule. A complete schedule must contain all the actions of every transactions that appears in it.

RECOVERABILITY

There are 2 main types of damage happened to the data base.

Type 1:When there is catastrophic error to the database. Ex:-when the hard disk crashes.
- The only way to recreate the data is if we have a copy of it stored as backup.
- During back up the entire database and the database transaction logs are copied to an alternative storage medium.

Type 2: there are several reasons for a transactions to fail which may lead to errors in database. This is called as non-catastrophic errors.

There are 2 mechanisms to do such error recovery.
- Deferred update method
- Immediate update method

Deferred update method
- In this method, the physical data on a database is not changed until the entire transactions is completed successfully.
- The successful completion of the transaction is called COMMIT point.
- During commit all update are stored in a file called as database log file.
- After committing the contents of the log are copied to the database.
- If the transactions fails before reaching the commit point there is no need to UNDO any data in database.
- Therefore, this method is called **NO UNDO/REDO METHOD.**

Immediate update method
- The dbms which use this method changes can be made on the database while a transaction is still being processed.
- However, before making any change in database the change is recorded in the log file.
- Only fter recording the change in log file the database is modified at each step.
- In this acse, if the tranasactions fails at some intermediate point then all updates perform before the fail point must be rolled back i.e. we need to UNDO this change.
- Next, to complete the transactions we must REDO the entire transactions.
- Therefore this method is called **UNDO/REDO method**.

Chapter-7
CONCURRENCY CONTROL CONCEPTS

If only one transaction to be executed at a time in serial order ,then performance will be quite poor.

- Then to improve the performance we can execute the transaction concurrently .But due to concurrent execution the lost update problem ,incorrect summary anomaly and dirty reads occurs.
- To recover from these problems we have to control the concurrency.
- If concurrency is controlled properly then the transaction through put can be maximized while avoiding the corruption of the data base.
- The concurrency can be controlled by the technique called **locking**.

LOCKING:

- A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it.
- Locking is a mechanism to ensure data integrity while allowing maximum concurrent access to data. It is used to implement concurrency control when multiple users access table to manipulate its data at the same time.

The simplest form of lock can be a binary lock having only 2 possible state only -
1. LOCK
2. UNLOCK

Depending upon the rules we have found, we can classify the locks into two types.

Shared Lock(S): A transaction may acquire shared lock on a data item in order to read its content. The lock is shared in the sense that any other transaction can acquire the shared lock on that same data item for reading purpose.

Exclusive Lock(X): A transaction may acquire exclusive lock on a data item in order to both read/write into it. The lock is excusive in the sense that no other transaction can acquire any kind of lock (either shared or exclusive) on that same data item.

The relationship between Shared and Exclusive Lock can be represented by the following table which is known as **LockMatrix.**

	Shared	Exclusive
Shared	TRUE	FALSE
Exclusive	FALSE	FALSE

Two Phase Locking Protocol:-

The Two Phase Locking Protocol defines the rules of how to acquire the locks on a data item and how to release the locks.
 The Two Phase Locking Protocol assumes that a transaction can only be in one of two phases.

Phase 1: The lock acquisition phase/Growing phase:

In this phase the transaction can only acquire locks, but can not release any locks till all the locks acquired by the transaction are obtained.

The transaction enters into growing phase as soon as it acquires first lock.

Once all locks have been acquired the transaction is in its **locked point**.

Phase 2: Lock Release Phase/Shrinking Phase:

After a lock point has been reached the transaction enters into shrinking phase.

The existing locks can be released in any order but no new lock can be acquired after a lock has been released.

The transaction enters into shrinking phase as soon as it release the first lock.
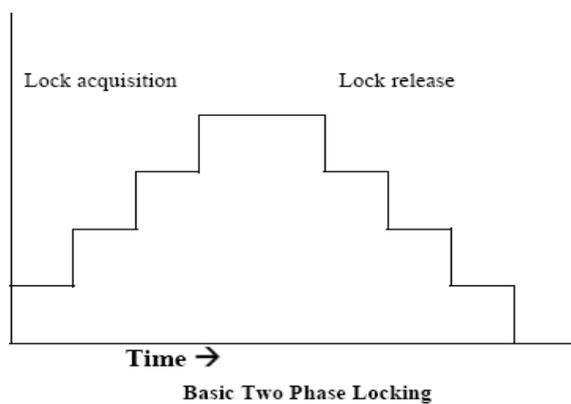
There are two types of 2PL:
>   (1) The Basic 2PL
>   (2) Strict 2PL

The Basic 2PL:

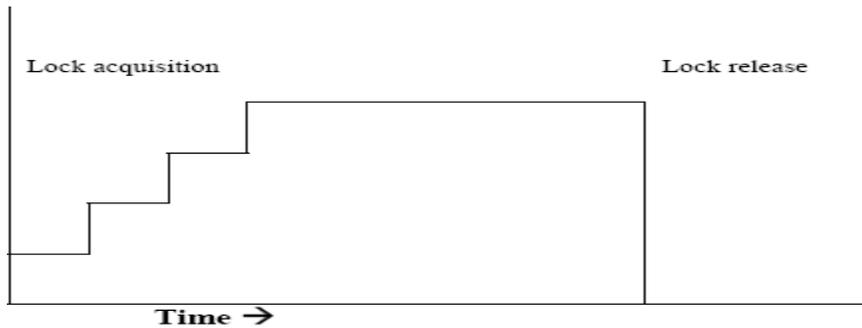The basic 2PL allows release of lock at any time after all the locks have been acquired.

Disadvantage: Once a lock has been released on a data item it can be modified by another transaction before the first transaction comits or aborts.

Basic Two Phase Locking

Strict 2PL:

To avoid such a situation we use strict 2PL. Transaction T does not release any of its exclusive lock (X) until that transaction comits or aborts.

In this way no other transaction can access the item that is written by T unless the transaction T comits.

Strict Two Phase Locking

But due to locking we can face two problems
   1. DEAD LOCK
   2. LIVE LOCK

DEAD LOCK:

A deadlock situation occurs when two transaction wait indefinitely for each other to unlock data.

Deadlock occurs when 2 transaction T1 &T2 exist in the following mode.

T1= Access data item X and Y

T2= Access data item Y and X

If T1 has not unlocked data item Y, T2 cannot begin. If T2 has not unlocked data item X ,T1 can not continue. Consequently T1 and T2 wait indefinitely, each waiting for the other to unlock the required data item. Such situation is called as deadlock.

NECESSARY CONDITION
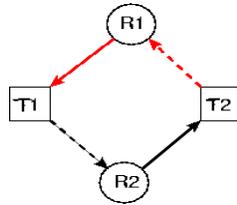There are four necessary condition for dead lock to occur
1.Mutual Exclusion
2.Non Primitive Locking
3.Partial Allocation
4.Circular wait
1.**Mutual Exclusion:** A resource can be locked in exclusive mode by only one transaction at a time.
2.**NON -Primitive Locking:** A data item can only be unlocked by the transaction that lock it No other transaction can unlock it.
3.**PARTIAL ALLOCATION:** A transaction can acquire locks on data base in a piecemeal fashion.

4.**CIRCULAR WAITING:** Transaction lock part of the data resources needed and then wait indefinitely to lock the resources currently locked by other transaction.

APPROACHES FOR DEALING WITH DEAD LOCK:
1.Prevention
2.Avoidance
3.Detection and recovery

DEADLOCK PREVENTION

In order to prevent the dead lock one has to ensure that at least one of above condition does not occurs.

The better prevention algorithm have been evolved to prevent a deadlock having the basic logic: not to allow the circular wait to occur.

Their are 2 schemes to prevent deadlock

1. Wait die scheme

2. Wound wait scheme

Wait die scheme:

This schemes is based on the non primitive technique .

It is based on a simple rule.
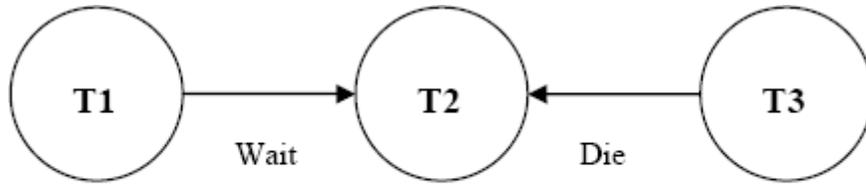
If Ti request a data base resource that is held by Tj .

Then if Ti has a smaller time stamp than that of Tj, it is allowed to wait

    else Ti aborts

A **timestamp** is defined as the system generated sequence no. that is unique for each transaction so a smaller time stamp means an older transaction.

For example:

Assume that 3 transaction T1,T2,T3, is generated in that sequence .if T1request for a data item which is currently held by transaction T2, it is allowed to wait as it has a smaller time stamp as that of T1, however T3 request for a data item which is current held by the T2 then T3 is rolled back(die or abort).

Wait-die Scheme of Deadlock prevention

2. WOUND WAIT

It is **b**ased **on** the primitive technique .
It is based on the simple rule:
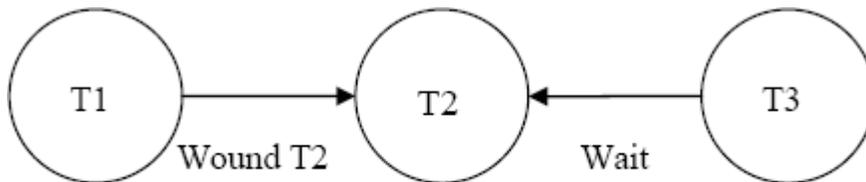If Ti request a database resources that is held by Tj
Then if Ti has a larger timestamp than that of Tj, It is allowed to wait
        else Tj is wounded of by Ti
FOR EXAMPLE:

Assume that 3 transaction T1, T2, T3 are generated in that sequence. . If T1 request for a data item which is currently held by transaction T2 , then T2 is rolled back and data item is allowed to T1. however if T3 request for a data item which is currently held by T2 then T3 is allowed to wait.

Wound-wait Scheme of Deadlock prevention

Here is the table representation of resource allocation for each algorithm

	Wait/Die	Wound/Wait
Older process needs a resource held by younger process	**Older process** waits	**Younger process** dies
Younger process needs a resource held by older process	**Younger process** dies	**Younger process** waits

DEAD LOCK AVOIDANCE:
Transactions are not required to request resources a priori.
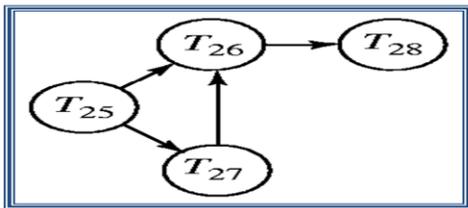The transaction must obtain all the locks it need before it can be executed**.**
Transactions are not allowed to proceed unless a requested resources is available.
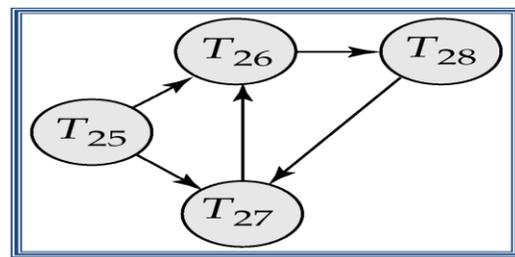In case of conflict, transactions may be allowed to wait for a fixed time interval.
This technique avoids rollback of conflicting transaction by requiring that locks be obtained in a sequence.

DEAD LOCK DETECTION:

- Deadlocks can be described as a wait-for graph, which consists of a pair $G = (V,E)$,

    - V is a set of vertices (all the transactions in the system)

    - E is a set of edges; each element is an ordered pair $T_i \rightarrow T_j$.

- If $T_i \rightarrow T_j$ is in E, then there is a directed edge from T_i to T_j

    - implying that T_i is waiting for T_j to release a data item.

- When T_i requests a data item held by T_j, then $T_i \rightarrow T_j$ is inserted in the wait-for graph.

    - This edge is removed only when T_j is no longer holding a data item needed by T_i.

- The system is in a deadlock state if and only if the wait-for graph has a cycle.

- The system invokes a deadlock-detection algorithm periodically to look for cycles.

Wait-for graph without a cycle

Wait-for graph with a cycle

DEADLOCK RECOVERY:

1. Through preemption
2. Roll back
 keep check point periodically
when a deadlock is detected , see which resource is needed.
Take away the resource from the process currently having it.
 Later on , you can restart this process from a check pointed state where it may need to re acquire the resource. from deadlock.
If more then one process takes action , the deadlock detection algorithm can repeatedly trigger.

<u>LIVELOCK</u>

- A live lock is similar to a deadlock, except a condition that occurs when two or more process continuously change their state in response to change in other process.
- The result is that none of the process will complete its transaction.
- Live lock is a special case of resource starvation the general definition only state that a specific process is not progressing.
- A real world example of live lock is it occurs when two people meet in a narrow corridor and each tries to polite by moving a side to let the other pass but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time.
- If more than one process takes action, the deadlock detection algorithm can repeatedly trigger .
- This can be avoided by ensuring that only one process takes action.

<u>Serializability</u>

Serializability is a property of a transaction schedule (history). It relates to the *isolation* property of a database transaction.

- Basic Assumption – Each transaction preserves database consistency.
- Thus serial execution of a set of transactions preserves database consistency.
- A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:
    - conflict serializability
    - view serializability
- We ignore operations other than **read** and **write** instructions, and we assume that transactions may perform arbitrary computations on data in local buffers in between reads and writes. Our simplified schedules consist of only **read** and **write** instructions.

Chapter-8
SECURITY AND INTEGRITY

DATA ENCRYPTION

- ➢ Sometimes some users want to bypass the system by physically removing the part of the database or by tapping into a communication line.
- ➢ The Most effective counter measure against such threats is data encryption; it means storing and transmitting of data in an encrypted form.
- ➢ The original data is called plaintext the plain text is encrypted by subjecting it to an encryption algorithm, whose I/P's are the plain text and encryption key.
- ➢ The O/P's from this algorithm is the encrypted form of the plain text and is called as the cipher text.
- ➢ The details of the encryption algorithm can be made public but encryption key is kept secret.
- ➢ The cipher text is stored in the database and transmitted down the communication line.
- ➢ The data encryption can be made by two means :-
   1. Substitution:-
In This type of encryption the plain text is substituted by another key, it gives the cipher text.
   2. Rearrange
In this type of encryption, the plain text characters are simply rearranged into some different sequence and then it combines with the encryption key and gives the cipher text.
Neither of the approaches are particularly secure in itself but the algorithm that combines are to provide.

AUTHORIZATION:-

Authorization is the administrative policies of organisation ,expressed as a set of values that be used to determine which user has what type of access on which portion of the database .

The person who is in charge of specifying the authorization is usually called the authorizer. The authorizer is not the DBA but the person who owns the data.

The authorization usually maintained in the table called as access matrix . The access matrix contains rows called subject is columns called as objects .

The entity in the matrix at the portion corresponds to the intersection of a row & column indicates type of access that the subject has with respect to the object.

Objects:-

An object in the database environment could be a unit of data that needs to be protected. Thus a object can be data that needs to be protected. Thus a object can be data field, a record , a file or view.

Subjects:-

A subject can be an active element who operates on objects to a class of users or an application program.
If an user belong to more than one class of users, then access rights for a given access made by user that is being use by that user of access.

Access Type:-

The access allowed to the user or program could be for data manipulation or control. The manipulation operations are read, insert, update. The control operations are add, drop, alter.

VIEWS:-

❖ The relations that really exist in the database are referred to as base relation.
❖ Sometimes for security purpose it is undesirable to have all the users to see the entire relation. Any relation that is not part of the physical database i.e. virtual relation is called as views/subschemas.
❖ The view can be created from the base relation according to the requirement the views can be created by 'SQL' Query.

Types of View

There are two types of view,
- Simple View

- Complex View

Simple View	Complex View
Created from one table	Created from one or more table
Does not contain functions	Contain functions

Does not contain groups of data	Contains groups of data

SYNTAX:-

CREATE VIEW viewname AS (query expression);
OR
CREATE VIEW viewname AS (select expression);
Example: create view view1 as select empno, empname from emp;
To see the view: select *from viewname;

> ➢ A view is a relation and can be used in query expressions. Views are generally not stored the data in the base relation may change .
> ➢ The create view state is stored in the system catalogue, when a query refers to any view the virtual relation defined by the view.

SECURITY AND INTEGRITY :-

- ❖ Security & Integrity are frequently heard together in database, but the two concepts are actual quite distinct.

- ❖ Security refers to the protection of data unauthorised disclosure, alteration or destruction security involves ensuring that users all to do things while others are trying to do.

- ❖ Integrity refers to the accuracy or validity of data.

- ❖ Integrity means ensuring that the things are trying to do are correct.

- ❖ Modern DBMS support either or both of 2 approaches to data security . They are as follows:-

    1) Discretionary Model

    2) Mandatory Control

- ❖ Most of Systems supports the discretionary control and some systems supports the mandatory control.

Discretionary Control:-

- ❖ Here are given user will have different access regular on privileges on different object, different users will have different rights on the same object.
- ❖ This Scheme is very flexible.

SYNTAX:-

CREATE SECURITY RULE rule name GRANT

Privilege – list ON expression To user – list [ON ATTEMPTED VIOLATION ACTION];

EXAMPLE:-

CREATE SECURITY RULE SR3 GRANT RETRIVE, DELETE ON S WHERE CITY ='BLS' TO X,Y,Z.
[ON ATTEMPTED VIOLATION REJECT];

The rule will register under system catalogue under the name SR3.

The privilege RETRIVE & DELETE are granted to the users by means of to clause where the city of table is Balasore.

Mandatory Control:-

❖ Mandatory control are applicable to database in which the data is rather static and rigid classification structure.
❖ This control is applied to the military or government environment.
❖ Here the data object has a classification level and user has a clearance level.
❖ User 'I' can see to object 'J' if the clearance level of 'I' is greater than or equal to clearance level of 'J'.
❖ User 'I' can modify object 'J' of the clearance level of 'I' is equal to the classification of 'J'.

INTEGRITY CONSTRAINT:-

Integrity constraint gives the protection to the data by following means:-
➢ It enforces rules on the data in a table where a row is inserted, updated or deleted from that table.
➢ Prevent deletion of a table on the content of its,   if there are dependencies from other tables.

The constraint types are :-
➢ NOT NULL
➢ UNIQUE
➢ PRIMARY KEY
➢ FOREIGN KEY
➢ CHECK

❖ The constraint can be imposed at the time of table creation or after the table has been created.
❖ The constraint can be defined at the column or table level.
❖ All the constraint are stored in the data dictionary.
❖ The Syntax when constraint is defined at the column level is a s follows:-

CREATE TABLE table name
Column data type
Column data type [CONSTRAINT CONSTRAINT-NO CONSTRAINT-TYPE];

❖ The Syntax when constraint is defined at the table level is a s follows:-

CREATE TABLE table name
Column dataype

[CONSTRAINT CONSTRAINT-NAME] CONSTRAINT-TYPE(COL);

NOT NULL:-

The NOT NULL constraint ensures the column contains no NULL values. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

SYNTAX:-
CREATE TABLE table name (Column name) data type
EXAMPLE:-
CREATE TABLE std (roll number (10) not null, name branch varchar2 (20, dob date);

UNIQUE:-

A unique key constraint requires that value in a column be unique i.e. no rows of table can have duplicate values.

SYNTAX:-

CREATE TABLE table name (Column name 1 data type, Column name 2 data type unique, Column name 3 data type);

EXAMPLE:-

CREATE TABLE std (roll number (10),name varchar2(10) unique, dob date not null, mark number (8));

PRIMARY KEY:-

A Primary key constraint creates a primary key for the table by which each row can be identify uniquely.

➢ Only one primary key can be created for each table.
➢ The primary key is the combination of unique & not null.

SYNTAX:-

CREATE TABLE table name (Column name 1 data type, Column name 2 data type primary key, Column name 3 data type);

EXAMPLE:-

CREATE TABLE std (roll number (10), CONSTRAINT PK-roll primary key, name varchar2 (10) unique, branch varchar2 (20), dob date not null);

FOREIGN KEY:-
  ❖ The foreign key or referential integrity constraint designates a column as a foreign key and establishes a relationship between primary key in the same table or a different table.

  ❖ The table where the foreign key is present is called the depend or child table where the primary key which is referred is present is called the referenced or parent table.

SYNTAX:-

CREATE TABLE table name 1 (Column name 1 data type primary key, Column name 2 data type reference table name 2, (Column));

EXAMPLE:-

CREATE TABLE std (roll number (10), CONSTRAINT PK-roll primary key, name varchar2 (10) unique, branch varchar2 (10), CONSTRAINT fR-branch);

CHECK:-
Check defines the condition that each row must satisfy. This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

SYNTAX:-
CREATE TABLE table name 1 (Column name 1 data type check (condition), Column name 2 data type);
EXAMPLE:-
CREATE TABLE sal (salary number (10) check (salary, name varchar2(10));
TRANSACTION FILE & MASTER FILE:-

  ❖ Master file is the main file, there is no other master file.
  ❖ Transaction file is derived file from the master file are a no. of transaction files.
  ❖ In Master file we can't do any updation or manipulation.
  ❖ All changes are done on transaction file.
  ❖ All the updation are done on the transaction file and are stored in the master file at a time on a particular day.
  ❖ Master file is the root and the transaction files are nodes.

SECURITY CONSTRAINTS:-

Security in a database involves both policies and mechanisms to protect the data and ensure that it is not accessed, altered or deleted without proper authorization.

There are four levels of defense or security constraints are generally recognized for database security: human factors, physical security, administrative control, and DBMS and Operating System Mechanisms .

1. Human Factors:-.
- ➢ An organization usually performs some type of clearance procedure for personnel who are going to be dealing with sensitive information, including that contained in a database.
- ➢ This clearance procedure can be a very informal one, in the form of the reliability and trust that an employee has earned in the eyes of management.

- ➢ The authorizer is responsible for granting proper database access authorization to the user community.

2. Physical Security:-
- ➢ Physical security mechanisms include appropriate locks and keys to computing facility and terminals.
- ➢ Security and physical storage devices (magnetic disk packs etc.) within the organization and when being transmitted from one location to another must be maintained..
- ➢ Authorized terminals from which database access is allowed to have to be physically secure, otherwise unauthorized persons may be able to access the data.
- ➢ User identification and passwords have to be kept confidential.

3. Administrative Controls:-
Administrative controls are the security and access control policies that determine what information will be accessible to what class of users and the type of access that will be allowed to this class.

4. DBMS and Operating System Mechanisms:
- ➢ The proper mechanisms for the identification and verification of users.
- ➢ Each user is assigned an account number and a password. The operating system ensures that access to the system is denied unless the number and password are valid.
- ➢ In addition to the DBMS could also require a number and password before allowing the user to perform any database operations.

References

1. Book Reference: A. Silberschatz, H.F. Korth Database System Concepts
2. Book Reference: Database Management System (DBMS)A Practical Approach Book by Rajiv Chopra
3. Book Reference: Database Management System by P.K Yadav
4. Book Reference: An Introduction to Database Systems Bipin C.Desai
5. https://nptel.ac.in
6. https://www.javatpoint.com
7. https://www.geeksforgeeks.org