Introduction of DBMS :- Section-A

A Database Management System is a collection of interrelated data and a set of programs to access those data.

- DBMS is used to organize the data in the form of a table, schema, view and report etc.

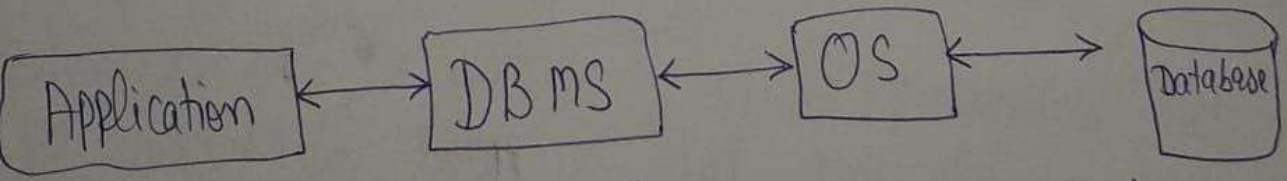
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

- Database management system is the combination of two words

Database + Management System = DBMS

- A Database is a collection of related information stored, so that it is available to many users for different purpose.

- Database Management System is a collection of programs that enables user to create and maintain the database.



- DBMS can also be defined as an interface between the Application program and the operating system to access and manipulate that database.

- Characteristics :-

- It uses a digital repository established on a server to store and manage the information.
- It can provides a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contain ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirement of the user.

- Advantages and Disadvantages :-

- Advantages :

(1) Control database redundancy → (repetition) :-

- It control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.

(2) Data sharing :-

In DBMS the authorized users of an organisation can share the data among multiple users.

(3) Easily Maintainance :-

It can be easily maintainable due to the centralized nature of the database system

(4) Reduce time :-

- It reduces development time and maintanance need.

(5) Backup :-

- It provides backup and recovery subsystems which create automatic backup of data from hardware and software failure and restores the data if required.

(6) Multiple user interface :-

- It provides different types of user interface like graphical user interfaces, application program interfaces.

⇒ Disadvantages :-

(1) Cost of hardware and Software :

- It requires a high speed of data processor and large memory size to run dbms software.

(2) Size :-

It occupies a large space of disks and large memory

to run them efficiently.

(3) Complexity :-

Database system creates additional complexity and requirements

(4) Higher impact of failure :-

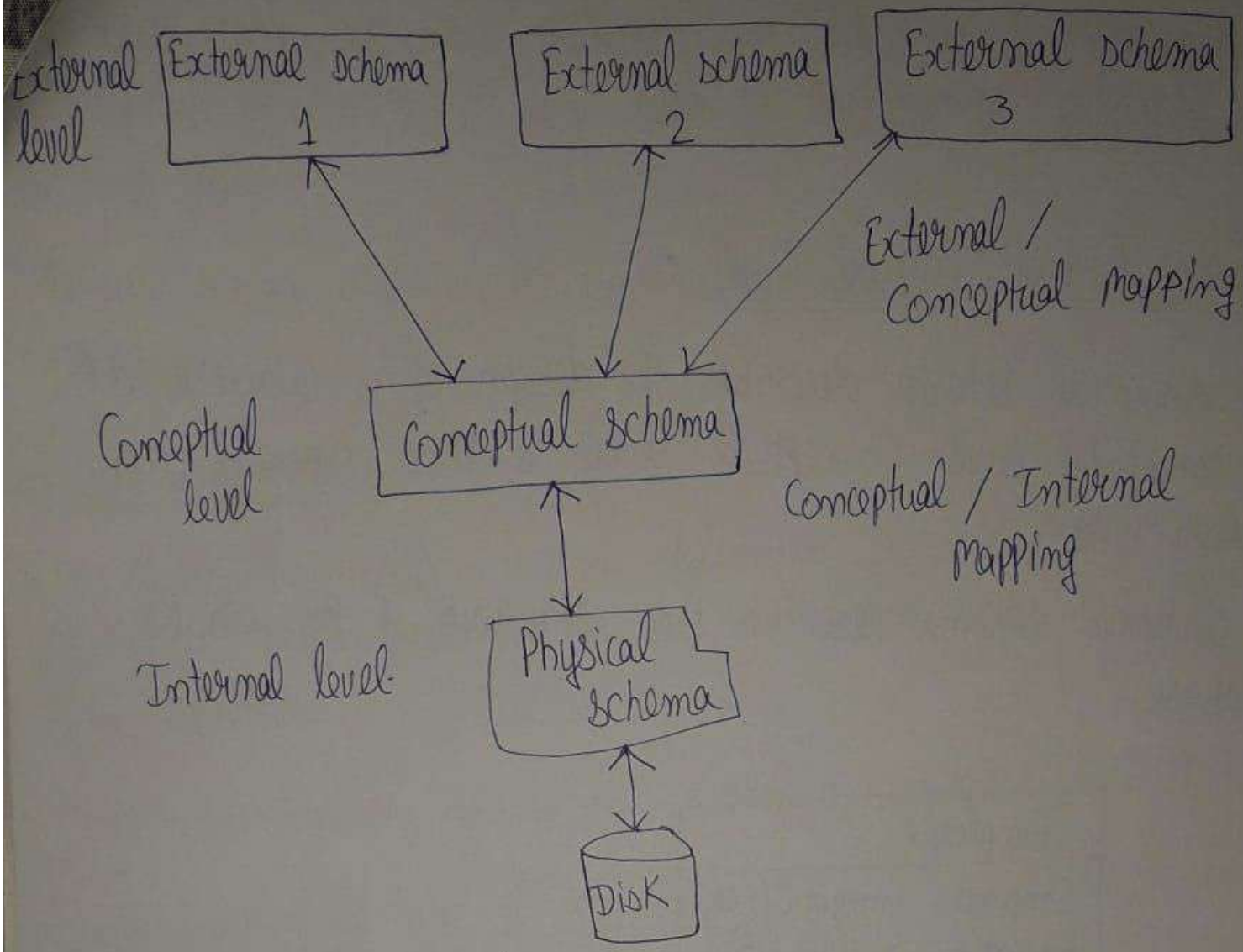
- Failure is highly impacted the database because in most of the organisation, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

⇒ Three Schema Architecture of DBMS :-

- The overall design of the database is called the database schema.
- The three schema Architecture is also called ANSI / SPARC (American National Standard Institute, Standards Planning and requirements committed architecture or three-level architecture.
- The three schema Architecture is also used to separate the user applications and physical database.

Three - schema Architecture

Diagram



1. Internal level / Internal view

- The internal level has an internal schema which describes the physical storage structure of the database.

Internal View

STORED_EMPLOYEE record Length 60
Empno : 4 decimal offset 0 unique
Ename : String length 15 offset 4
Salary : 8,2 decimal offset 19
Dept no : 4 Decimal offset 27
Post : String length 15 offset 31

- The internal schema is also known as a physical schema
- It uses the physical data model. It is used to define the how the data will be stored in a block.

- The physical level is used to describe complex low-level data structure in detail.

2. Conceptual level / Logical level :-

- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level
- The conceptual schema describes the structure of the whole database

Employee
Emp no : integer (4) Key
Ename : String (15)
Salary : String (8)
Dept no: integer (4)
Post : String (15)

- In the conceptual level, internal details such as an implementa of the data structure are hidden
- Programmer and database administrator work at this level.

3. External level / View level :-

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.

Empno	Ename	Salary	Dept no	Post

Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.

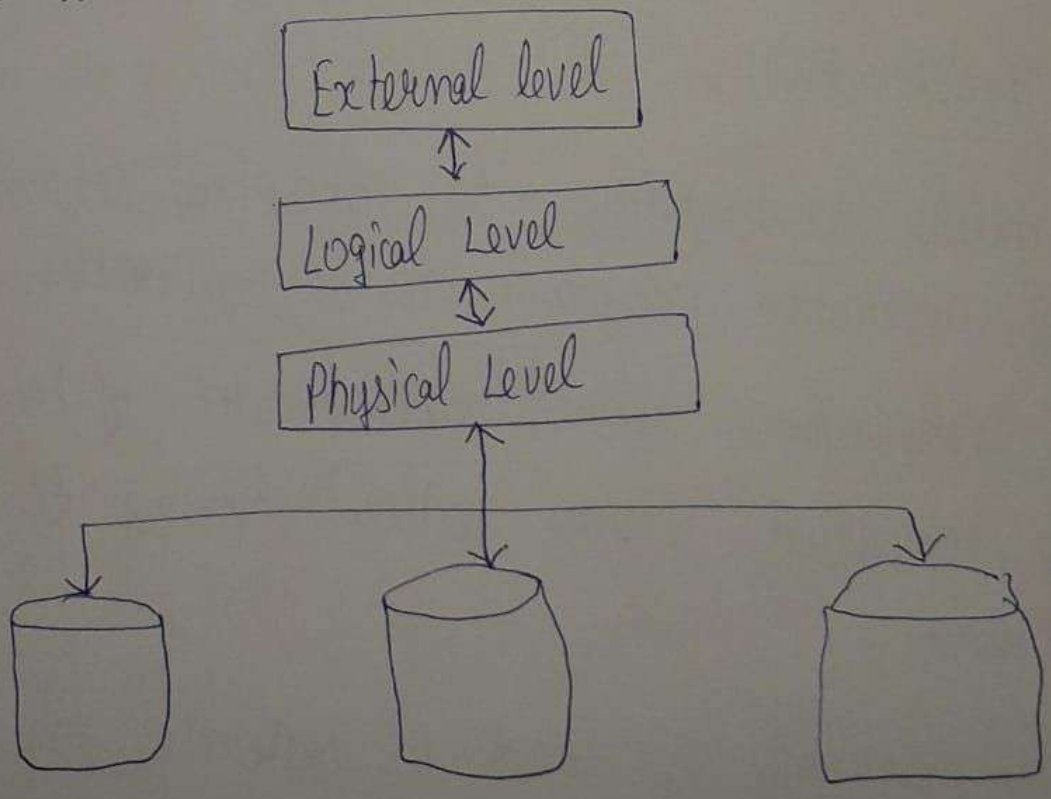
- The view schema describes the End user interaction with database systems.

⇒ 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.
- Data independence is one of the main advantages of DBMs.

Data Independence are of two types :-

- Physical data Independence
- Logial data ——



Physical data Independence :-

Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.

- If we do any changes in the storage size of the database system server then the conceptual structure of the database will not be affected.
- It is the ability to modify the physical schema without causing application programs to be rewritten.
- Physical data Independence is used to separate conceptual levels from the internal levels.
- It occurs at the logical Interface levels.

Logical data Independence :-

- It is the ability to modify the conceptual schema without causing application program to be rewritten.
- Logical data Independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual.

If we do any change in conceptual view of the data, the user view of the data would not be affected.

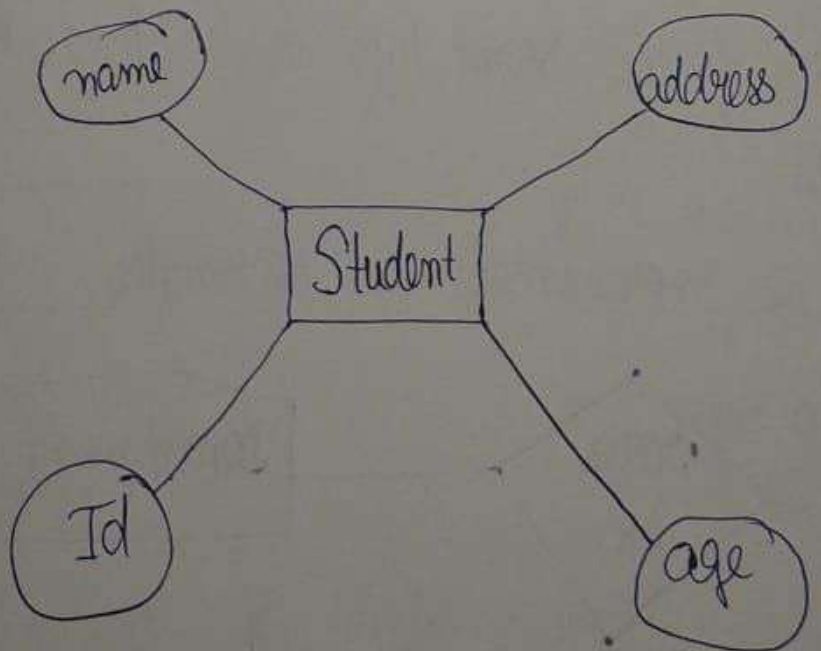
- Logical data independence occures at the user interface level.

⇒ ER Model :-

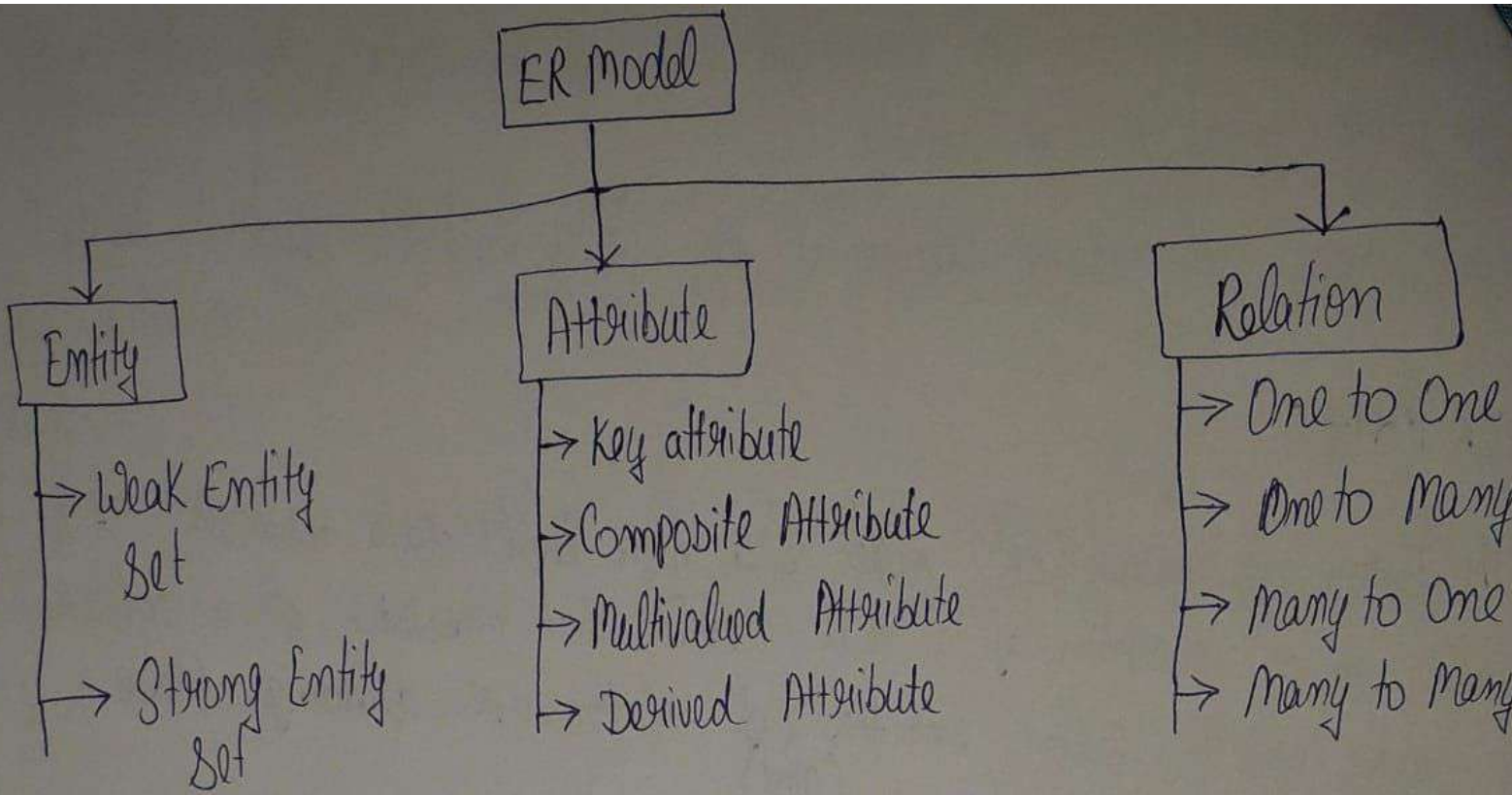
- The Entity relationship (ER) model is a high level data model. It is based on a perception of a real world that consists of a collection of basic objects called entities and relationships among these objects.
- It developes a conceptual design for the database. It also developes a very simple and Easy to design view of data.

For example :

Suppose we design a school database. In this database, the student will be an entity with attribute like address, name, id, age etc.



⇒ Components of ER Model Diagram :-

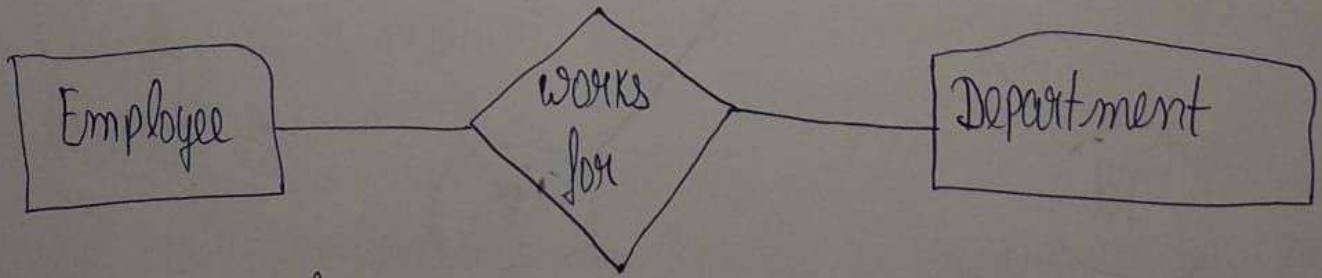


Entity :

- It is a thing or object in the real world that is distinguisable from all other object.
- Anything about which we store information is called an Entity

Entity Set :

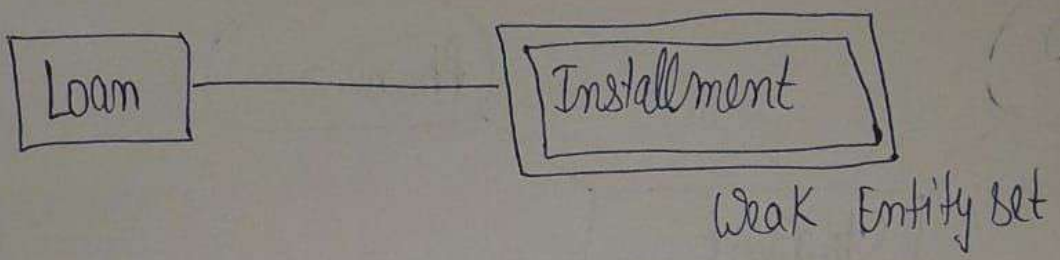
- It is a set of Entities of the same type that store the same properties or attributes.
- An entity set can be represented as rectangles.



⇒ Types of Entity Set :-

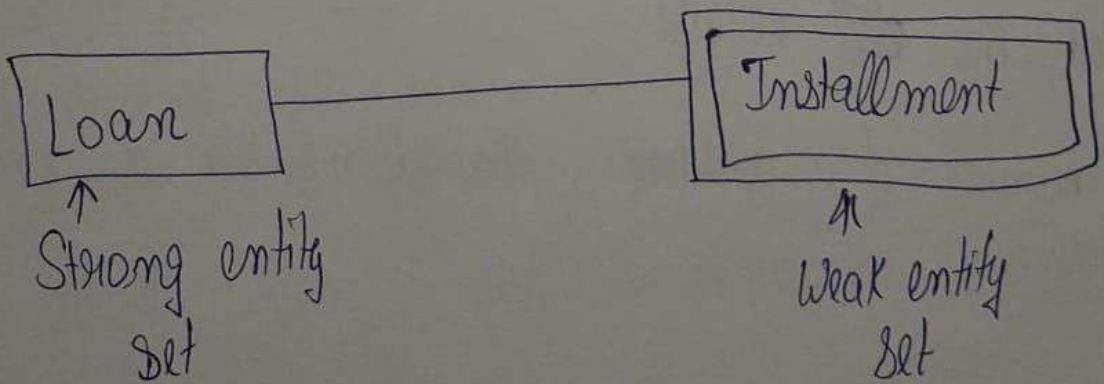
Weak Entity Set :-

An entity that depends on another Entity called a weak Entity set
- The weak Entity set doesn't contain any key attribute of its own.
- Weak Entity set is represented by a double rectangle.



2. Strong Entity Set :-

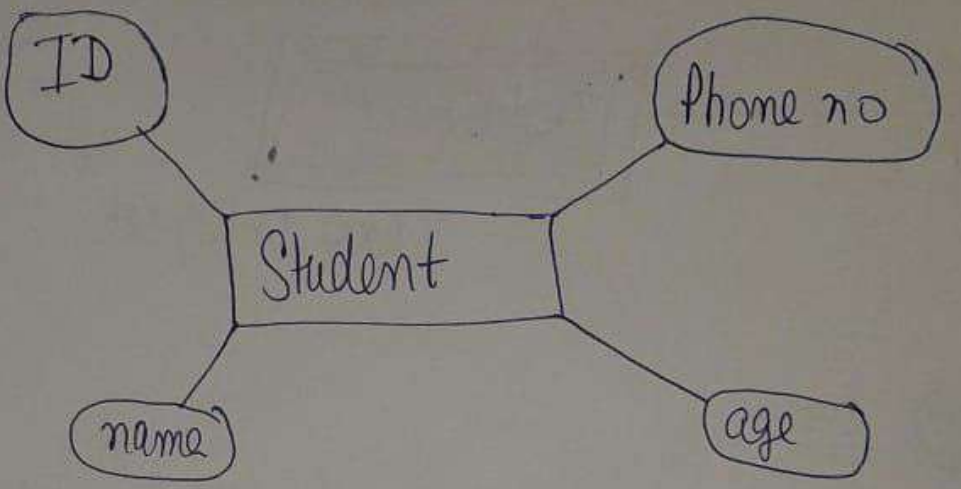
- A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
- Primary key exists for a strong entity set.
- Single rectangle is used to representing a strong entity set.



=> Attributes of ER Model :-

- The attribute is used to describe the property of an Entity.
- An entity set may contain any number of attribute.
- Attributes are represented in an elliptical shape.

For example :- ID, age, contact number, name etc can be attributes of a student.



Types of attributes

(a) Simple attribute :-

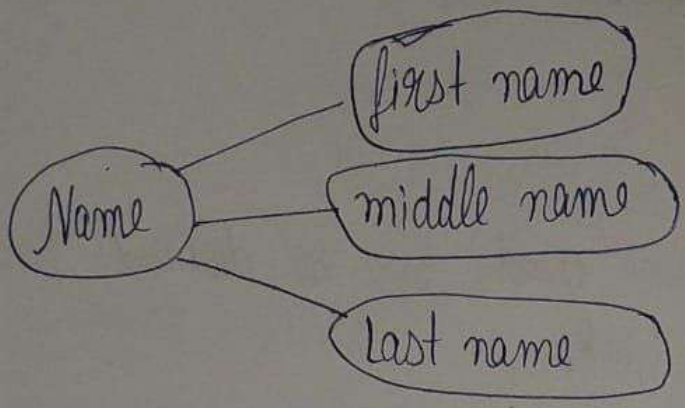
- An attribute that can not be further subdivided into components is a simple attribute.
- It is represented by ellipse.

Ex :- The roll number of a student, the id number of an Employee.

(b) Composite attribute :- An attribute that can be split into

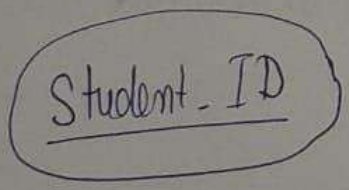
mponents is a composite attribute.

The composite attribute is represented by an ellipse and those ellipse are connected with an ellipse.



ⓒ Key attribute :-

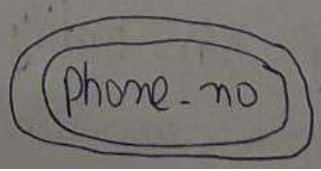
- The key attribute is used to represent the main characteristics of an entity.
- It represent a primary key.
- The key attribute is represented by an ellipse with the next underlined.



ⓓ Multi valued Attribute :-

- An attribute can have more than one value. These attributes are known as a multivalued attribute.
- The double Ellipse is used to represent multivalued attribute.

For example :- A student can have more than one phone number.



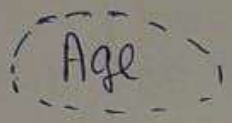
2) Derived attribute :-

An attribute that can be derived from other attribute is known as a derived attribute.

- It can be represented by a dashed Ellipse.

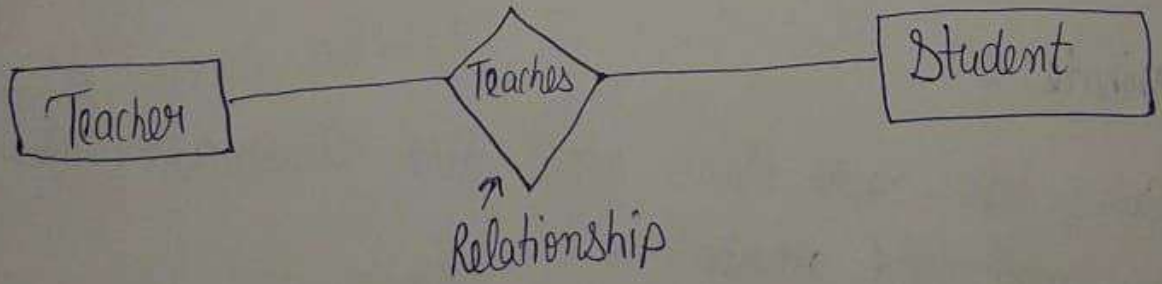
For example →

A person's age changes over time and can be derived from another attribute like date of birth.



⇒ Relationship / Mapping Constraints

- A relationship is used to describe the relation between entities.
- Diamond or rhombus box is used to represent the relationship.
- Mapping Cardinalities or Cardinality ratios, express the no. of entities to which another entity can be associated via a relationship set.



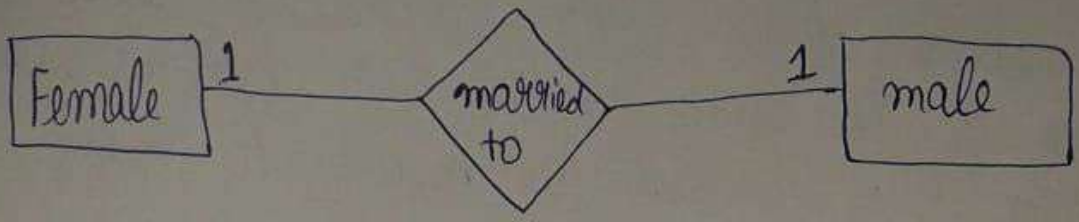
There are four types of mapping constraints or relationship.

1) One to one Relationship :-

When only one instance of an entity is associated with the relationship then it's known as one to one relationship.

example :-

A female can marry to one male and a male can marry to one female.

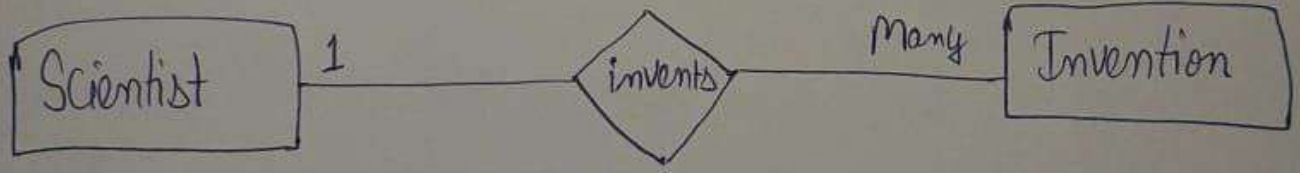


2. One to many relationship :-

When only one instance of the entity on the left and more than one instance of an entity on the right associates with the relationship then this is known as one to many relationship.

Example :-

Scientist can invent many inventions, but the invention is done by the only specific scientist.

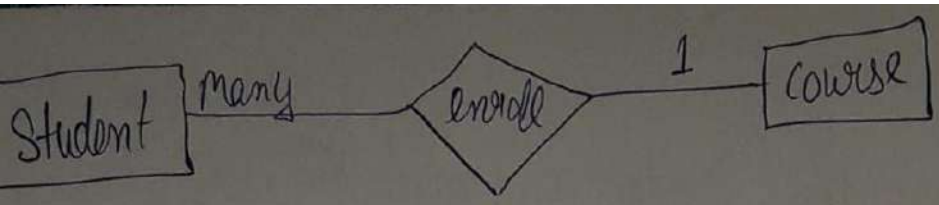


3. Many to one Relationship :-

- When more than one instance of the Entity on the left and only one instance of an entity on the right associates with the relationship is known as many to one Relation.

Example :-

Student enrolls for only one course, but a course can have many students.

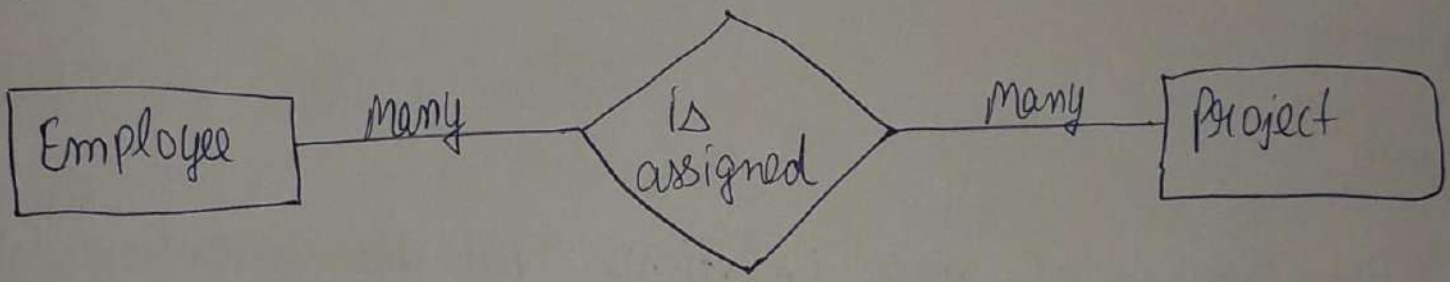


1. Many to Many :-

When more than one instance of an entity on the left and more than one instance of an entity on the right associates with the relationship then it is known as many to many relationship.

Example :-

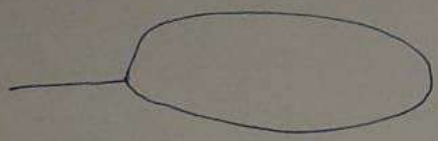
Employee can assign by many projects and project can have many employee.



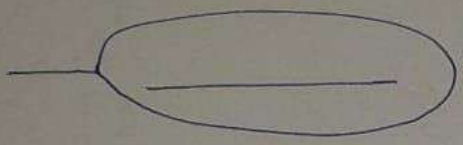
Section - A

→ Notations of ER Diagram :-

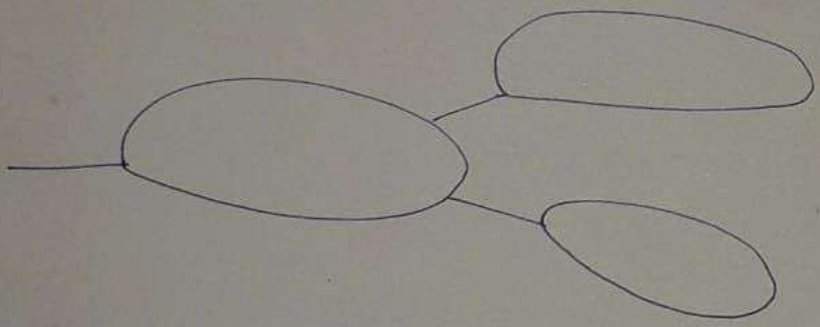
- Database can be represented using the notations. In ER Diagram, many relations are used to express the cardinality.
- These notations are as follows.



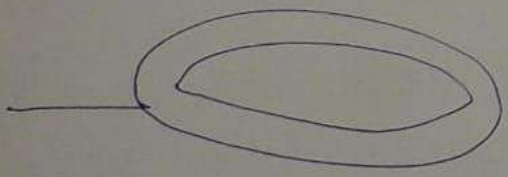
Attributes



Key Attributes



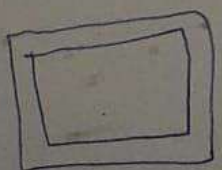
Composite Attribute



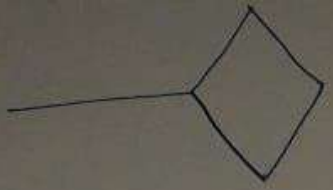
Multivalue attribute

Derived attribute

Strong Entity set

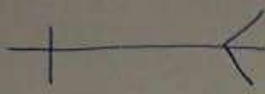


Weak Entity set



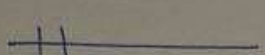
Relationship

Links (one to one)

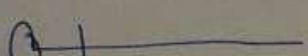


one to Many

Many to one



One and only one



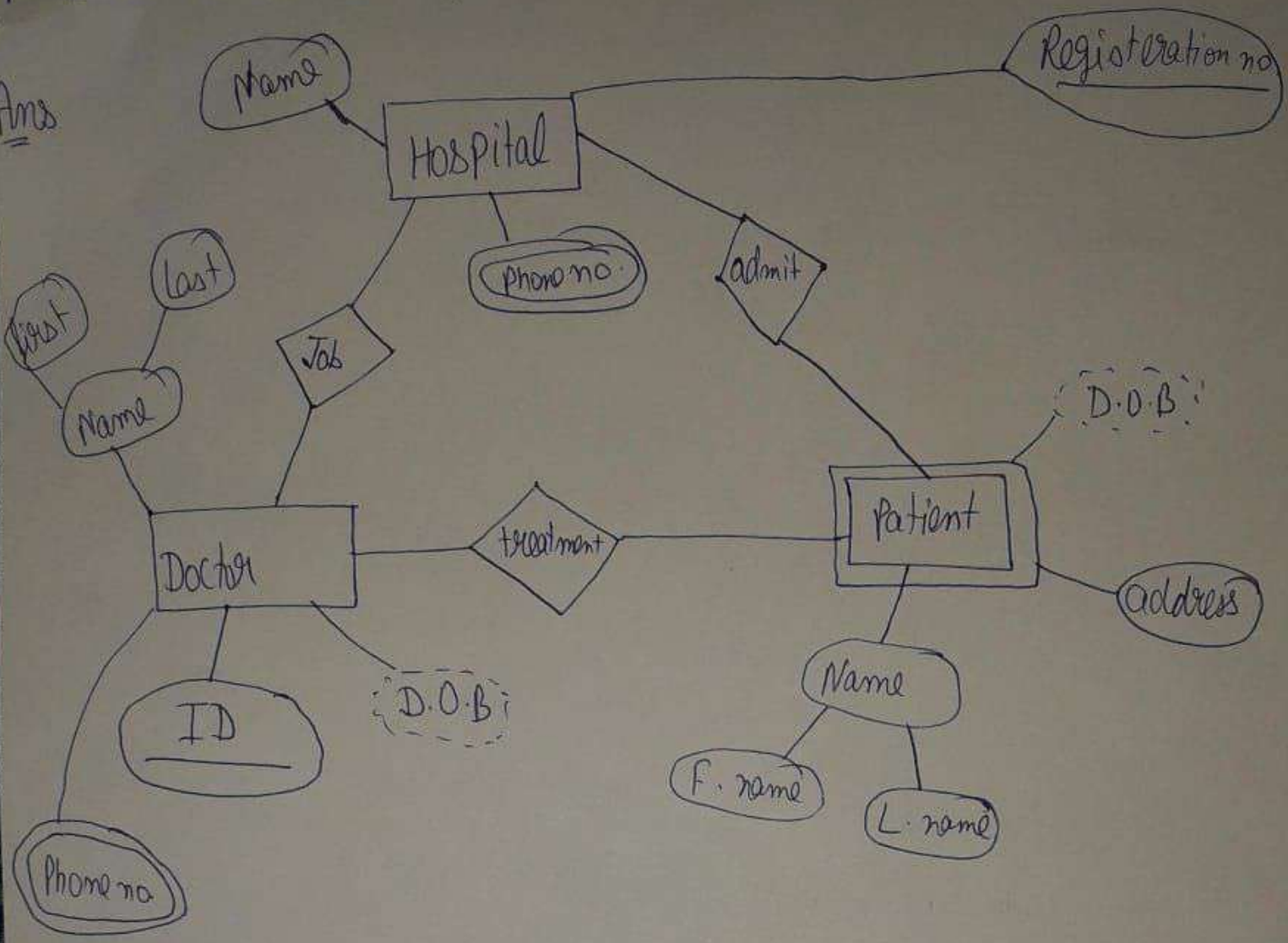
Zero or one

Zero or many

Ques Construct an ER Diagram for a hospital with a set of

patients and a set of medical doctors.

Ans



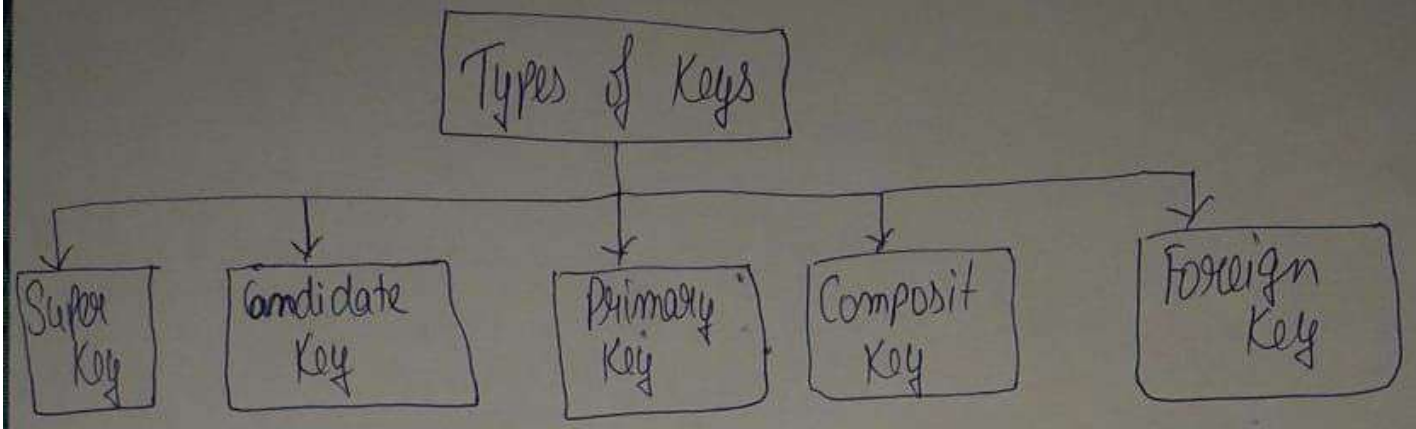
⇒ Keys :-

- A Key is a value which can always be used to uniquely identify an object instance.
- Key is used to uniquely identify any record or row of data from the table.
- It is also used to establish and identify relationships between tables.

Example :-

ID is used as a Key in the student table because it is unique for each student. In the person table, passport number, license

number are keys since they are unique for each person.



1. Super Key :-

- A Super Key is a set of one or more attributes that taken collectively, allow us to identify uniquely an entity in the entity set.

For example :- In student table with attribute
(S_Rollno., S_Name, S_Branch, S_Year)

Super Key ⇒ S_1 → S_Rollno., S_Name
S_2 → S_Rollno., S_Branch
S_3 → S_Rollno., S_Year
S_4 → S_Rollno., S_Name, S_Branch

2. Candidate Key :-

- The minimal set of attributes that can uniquely identify a table is known as a candidate key.
- Candidate Key can be define as the minimum no. of super key that identifies the record uniquely.
- It must contain unique values.

- Every table must have at least a single candidate key.

For example :-

In student table with attribute
(S_Rollno., S_Name, S_Branch, S_Year)

Candidate Key ⇒ C1 ⇒ S_Rollno
C2 ⇒ S_Rollno, S_name

3. Primary Key :-

- Primary key can be define as the minimum no. of candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.

- It is a unique key.
- It can identify only one tuple at a time.
- It has no duplicate values, it has unique values
- It can not be NULL.
- Primary keys are not necessarily to be a single column, more than one the column can also be a primary key for a table.

For example ⇒ In student table with attribute (S_Rollno, S_Name, S_Branch, S_Year)

Primary Key ⇒ P1 ⇒ S_Rollno

4. Composite Key :- Whenever a primary key consists of

more than one attribute it is known as a composite key

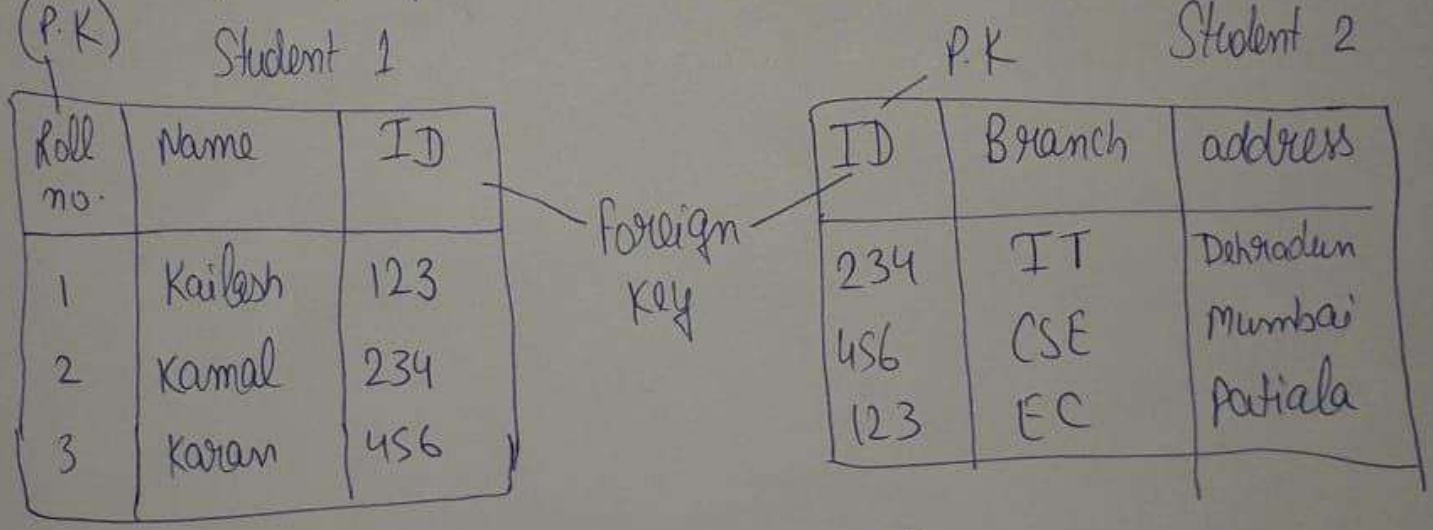
For example :-

In student table with attribute (S_Roll no, S_ID, S_Name, S_Branch)

Composit Key => S_Roll no, S_ID

5. Foreign Key :-

- A foreign key is a column whose value are the same as the primary key of another table.
- It combines two or more relations (table) at a time.
- They act as a cross reference between the table.
- foreign keys are the column of the table used to point to the primary key of another table.



=> EER Model (Extended / Enhance ER Model) :-

- EER model stands for enhance or extension entity relationship model.
- It is an extension of ER model.
- In compare of ER model, this EER model makes

Complex real-world relationships more easy due to its additional concepts like Generalization, Specialization, aggregation.

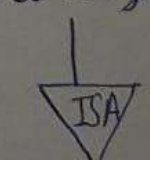
EER model is a high level data model which provides following features :-

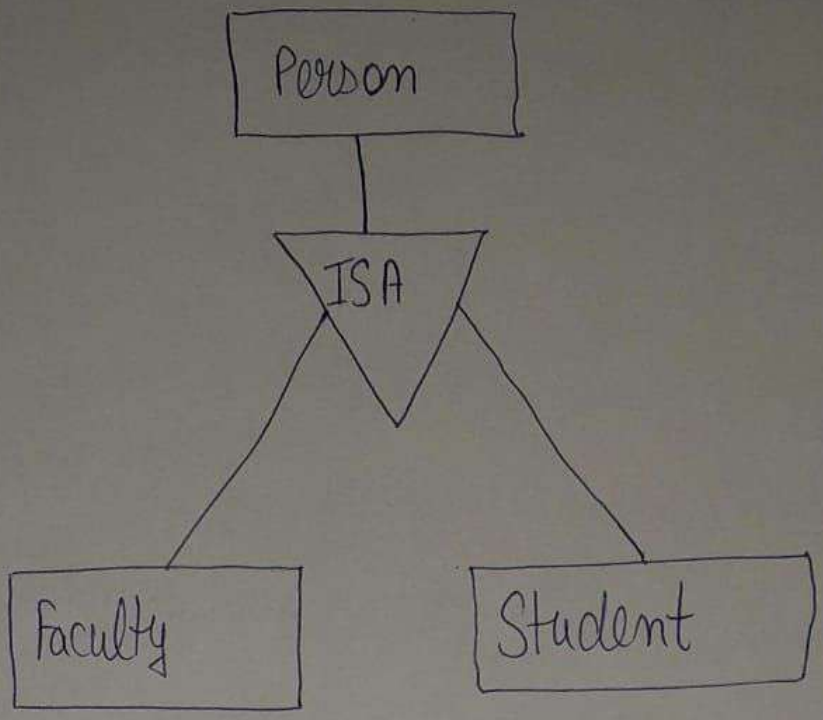
1) EER creates a design more accurates to database schema.
2) It includes all the modeling features of ER model.
3) EER model is widely used to design complex database in large applications.

1. Generalization :-

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- Generalization is a relationship that exists between a high level entity set and one or more lower level entity set.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization entities are combined to form a more generalized Entity such as subclass are combined to make a superclass.

For example :- Faculty and student Entities can be generalized and create a higher level Entity Person.



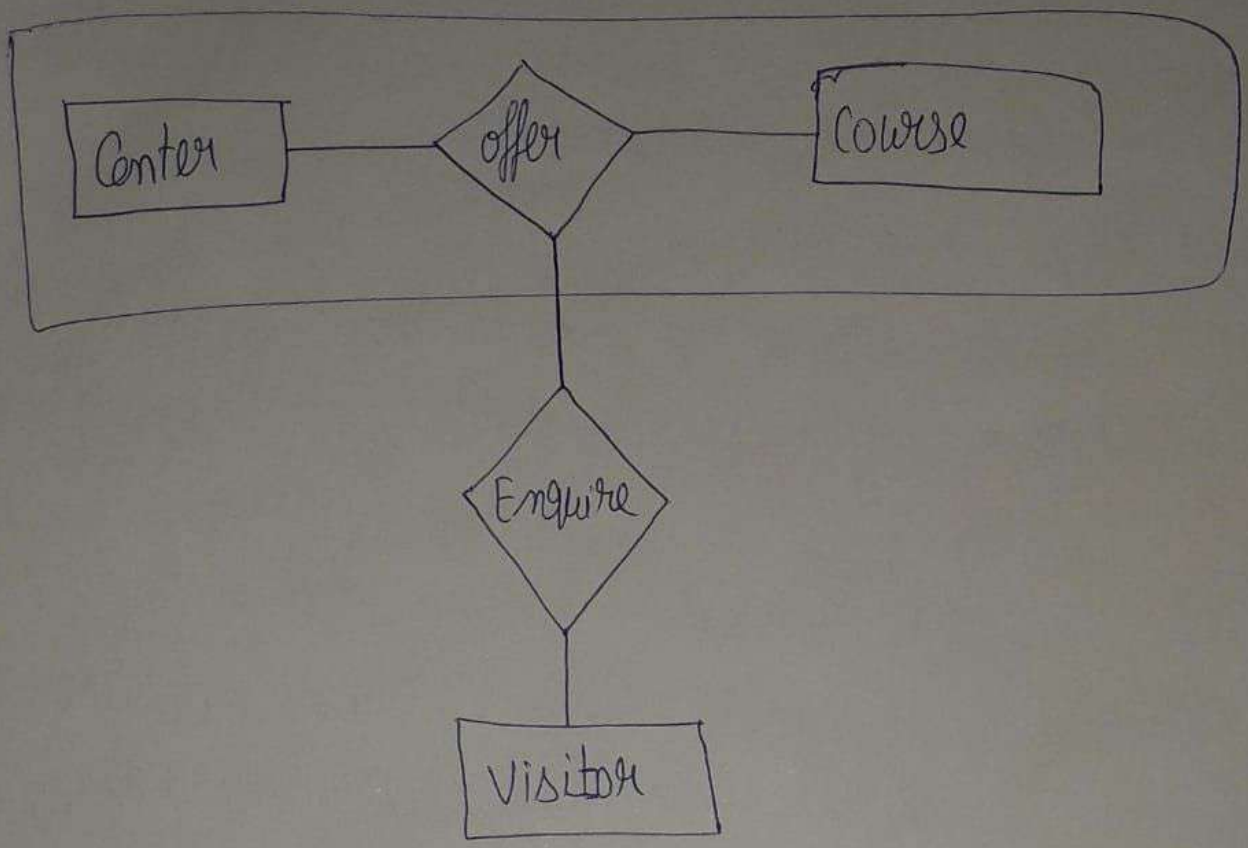


2. Aggregation :-

- Aggregation is a technique to express relationship among relationship
- Through E-R Modeling we can not express relationship among relationships. Thus, we use the concept of aggregation for this purpose.
- Aggregation is an abstraction through which relationships are treated as Entities.
- In aggregation, the relation between two entities is treated as a single Entity.
- In aggregation, relationship with its corresponding entities is aggregated into a higher level Entity.

For example :- Center Entity offers the course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never Enquiry

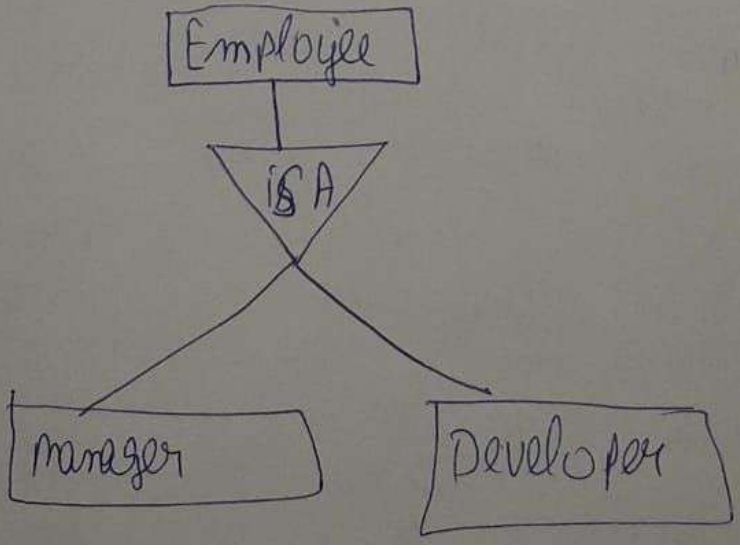
about the course only or just about the center instead he will ask the Enquiry about both.



3. Specialization :-

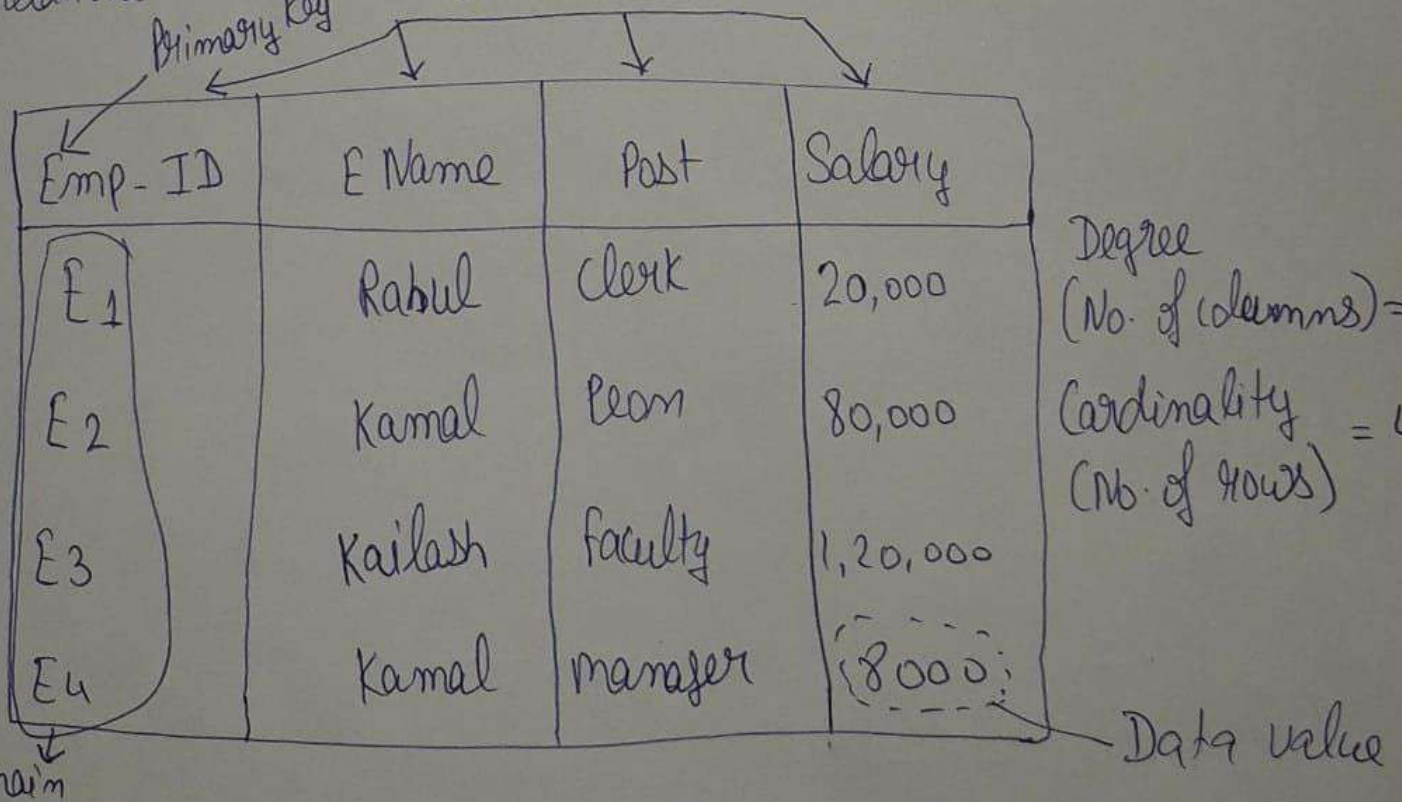
- It is opposite of generalization. It creates an sub-entity from high level entity.

Example :- Employee is a higher level entity can specialize it into sub-entities like manager and developer based on attributes.



⇒ RDBMS :-

- RDBMS stands for Relational database management system.
- RDBMS is a database management system that is based on the relational model as Introduced by Dr. E.F. Codd.
- RDBMS stores data in the form of related tables.
- An important feature of relational systems is that a single database can be spread across several tables.
- All modern database management system like SQL, MYSQL server, IBM DB2, ORACLE, and Microsoft access are based on RDBMS.
- A relational database is the most commonly used database. It contains several tables and each table has its primary key.
- Due to a collection of an organised set of tables, data can be accessed easily in RDBMS.
- Everything in a relational database is stored in the form of relations.



- Table / Relation :-

- Everything in a relational database is stored in the form of relations.
- The RDBMS database uses tables to store data.
- A table is a collection of related data entries and contains rows and columns to store data.

Properties of Relational tables :-

- Value are atomic
- Column value are of the same Kind
- Each row is unique.
- Each column has a unique name.
- The sequence of rows and columns is insignificance.

- Row or Record :-

- A row of a table is also called a record or tuple.
- Row contains the specific information of each entry in the table.
- It is a horizontal entity in the table.

Properities of a row =>

- No two tuples are identical to each other in all their entries.
- All tuples of the relation have the same format and the same number of entries.
- The order of the tuple is irrelevant. They are identified by

their content, not by their position.

- Column / attribute / fields :→

- A column is a vertical entity in the table which contains all information associated with a specific field in a table.

Properties of an attributes :-

- Every attribute of an relation must have a name.
- Null values are permitted for the attributes.
- Default values can be specified for an attribute automatically inserted if no other value is specified for an attribute.

- Data item / cells :-

- The smallest unit of data in the table is the individual data item.
- It is stored at the intersection of tuples and attributes.

- Degree :-

- The total no. of attributes that comprises a relation is known as the degree of the table.

- Cardinality :-

- The total number of tuples at any one time in a relation is known as the table's cardinality.
- The relation whose cardinality is zero is called an Empty table.

Domain :-

The domain refers to the possible values each attribute can contain.

- It can be specified using standard datatypes such as Integers floating numbers etc.

⇒ Dr. E.F. Codd's Rules for RDBMS :-

- Dr. E.F. Codd is an IBM researcher who first developed the relational data model in 1970.
- In 1985, Dr. Codd published a list of 12 rules that define an ideal relational database and has provided a guideline for the design of all relational database.

Rule 1 :- The Information Rule

- This rule simply requires that all data should be presented in table form. This is the basis of relational model.

Rule 2 :- Guaranteed Access Rule

- Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value) and attribute-name (column-~~name~~ value).

Rule 3 :- Systematic treatment of NULL value

- The NULL values in a database must be given a systematic and uniform treatment. This is very important rule

because a NULL can be interpreted as one the following :-
data is missing, data is not known or data is not appliable

Rule 4 :- Active Online Catalog

- The structure description of the entire database must be stored in an online catalog, known as data dictionary which can be accessed by authorized users.
- Users can use the same query language to access the catalog which they uses to access the database itself.

Rule 5 : Comprehensive data Sub-Language Rule

- A database can only be accessed using a language having linear syntax that support data definition, data manipulation and transaction management operations.

Rule 6 :- View updating Rule

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

Rule 7 :- High level Insert, Update and Delete

- A database must support high-level insertion, updation and deletion. This must not be limited to a single row, that is.
- It must also support union intersection and minus operations to yield sets of data records.

Rule 8: Physical data Independence

- The data stored in a database must be independent of the applications that access the database.
- Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applicati

Rule 9: Logical data Independence

- The logical data in a database must be independent of its user views (Applications). Any change in logical data must not affect the applications we it.

Rule 10: Integrity Independence

- The database language (like SQL) should support constraints on user input that maintain database integrity.
- No component of a primary key can have a null value.

Rule 11: Distribution Indepence

- The end user must not be able to see that the data is distributed over various locations users should always get the impression that the data is located at one site only.
- This rule has been regarded as the foundation of distributed database system.

Rule 12: Non-subversion Rule

- There should be no way to modify the database structure other than through the multiple row database language (SQL) most databases today.
- Support administrative tools that allows some direct manipulation

of the data structure.

Section-A

⇒ Relational Model Concept :-

- Relational model can represent as a table with columns and rows.
- Each row is known as a tuple.
- Each table of the column has a name or attribute.

Domain ⇒ It contains a set of atomic values that an attribute can take.

Attribute ⇒ It contains the name of a column in a particular table. Each attribute is a domain value.

Relational instance ⇒ In the relational database system, the relational instance is represented by a finite set of tuples. Relation instance do not have duplicate tuples.

Relational schema :- A relational schema contains the name of the relation and name of all columns or attributes.

Relational Key ⇒ In relational Key each row has one or more attributes. It can identify the row in the relation uniquely.

Example: Student Relation

Name	Roll_No	Phone_No	Address
Ram	123	34567	Delhi
Kamal	234	25678	Dehradhun
Kailash	345	12345	Haridwar

- In the given table, Name, Roll no, phone_no are the attributes.

- The instance of schema student has 3 Tuples.

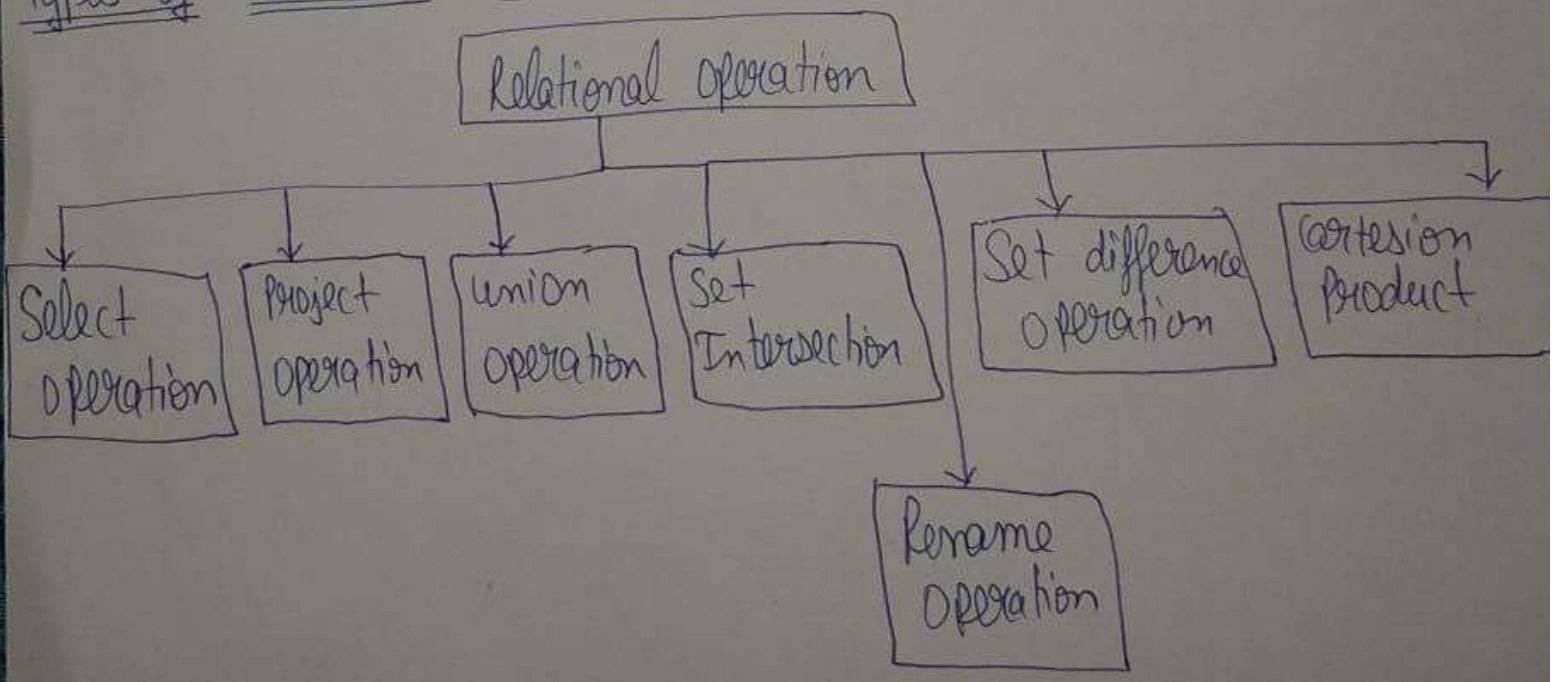
⇒ Properities of relations :-

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value.
- Each attribute contains a distinct name.
- Attribute domain has no significance.
- Tuple has no duplicate value.

⇒ Relational Algebra :-

- Relational algebra is a procedural query language.
- It gives a step by step process to obtain the result of the query.
- Relational algebra mainly provides theoretical foundation for relational databases and SQL.
- It uses operators to perform queries.

Types of Relational operation :



1> Select operation (σ) :-

- The select operation is used to select a subset of the tuples from a relation that satisfies a select condition.
- σ (Sigma) is used to denote select operation.
- The select operator is univory such that it is applied to a single relation.

The general format of select operation is

$$\sigma \ <\text{select condition}> (R)$$

where σ is used for selection prediction

R is used for Relation

Select condition is the relational operators like $=, \geq, <, \leq$

Ex => Student

Name	Roll no.	address
Aman	02	Batinda
Karan	04	Patiala
Arun	08	Delhi
Ankit	13	Bombay

query 1 => Give all information of student having Rollno is 04.

Sol σ Rollno = 04 (Student)

query 2 :- Find all the information of student having name

is Arun and address is delhi.

Sol σ Name = "Arun" and address = "Delhi" (Student)

2. Project operation (Π)

- Project operation selects certain columns from the table and discard the other columns.
- This operation shows the list of those attributes that we wish to appear in the result.
- Rest of the attributes are Eliminated from the table.

The general format for project operation is

$$\Pi < \text{attribute list} > (R)$$

Where Π is the symbol used to represent the project operation and attribute list is the list of attributes from the attribute of Relation R.

Example :- Student

Name	Roll no	Address
Aman	02	Patiala
Karan	04	Ludhiana
Amit	13	Bathinda

Query 1 => Find student Name in the student table.

Sol => Π name (Student)

Query 02 ⇒ Find student Name and address list.

Sol ⇒ Π Name, address (student)

3. Union operation ($\cup$) ⇒

- It performs the binary union between two given relations and is define as

$$R \cup S$$

- Where R and S are either database relations or relation result set (temporary relation).

Union operation to be valid, the following conditions must hold -

(i) R and S must have the same number of attributes.
(ii) Attributes domains must be compatible
(ii) Duplicate tuples are automatically eliminated.

Example -

$$\Pi_{name}(\text{Student 1}) \cup \Pi_{name}(\text{Student 2})$$

4. Set Intersection ($\cap$) ⇒

- It performs binary intersection between two given applicate and is defined as :-

$R \cap S$

Where R and S are either database relations or relation result set.

Example ⇒ Π_{name} (Student 1) $\cap$ Π_{name} (Student 2)

5) Set difference (-) ⇒

- The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

$R - S$

Notations is to finds all the tuples that are present in R but not in S.

Example :- Π_{name} (Student 1) - Π_{name} (Student 2)

6. Cartesian Product (X) ⇒

- Combines information of two different relations into one.

$R \times S$

Where R and S are relations and their oput will be defined as -

$R \times S = \{q E \mid q \in R \text{ and } t \in S\}$

Ex:

$\sigma_{Name = Kamal}$ (Student 1 X Student 2)

4. Rename Operation (P) :=>

- Rename operation is used to rename the output of a relation
- Rename operation is denoted with small Greek letter rho (P)

$$P_x(E)$$

Where the result of expression E is saved with name of x

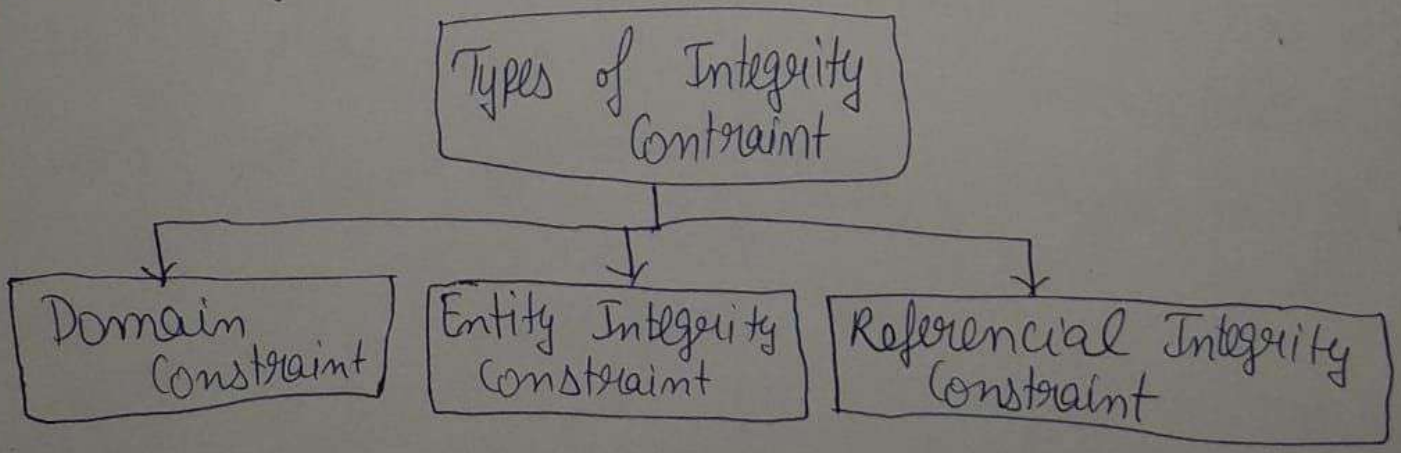
Example :-

Query to rename the table student to Employee and its attributes name, address, phone no

$$P_{Employee(name, address, phone\ no)}(Student)$$

=> Integrity Constraints :- (Most Important)

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating and other processes have to be performed in such a way that data integrity is not affected.



1. Domain Constraint :-

- Domain constraints can be defined as the definition of valid set of values for an attribute.
- The datatype of domain includes string, character, integer, time, date etc.
- The value of the attribute must be available in the corresponding domain.

Example :-

ID	Name	Age
101	Aman	20
102	Arun	(A)
103	(1)	30
104	Karan	35

Not allowed because age is an integer attributes.

Not allowed because Name is an string attribute

2. Entity Integrity constraint :-

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example :-

Emp_ID	Emp_Name	Salary
123	Kamal	30,000
245	Karan	40,000
562	Arun	50,000
NULL	Ram	90,000

Not Allowed as primary key can't contain Null value

3. Referential Integrity Constraint :-

- A referential integrity constraint is specified between two tables.
- In the referential integrity constraints, if a primary key in Table 1 refers to the primary key on Table 2, then every value of the foreign key in Table 1 must be null or be available in table 2.

P.K (Table 1) — Foreign Key

E_No	Name	Age	D_No
1	Kamal	20	11
2	Aman	30	24
3	Arun	30	18
4	Ram	25	13

Not allowed as D.No 18 is not defined as primary key of table 2 and table 1 D.No is a foreign key defined

Relationships

(Table 2)

D.No	D_Location
11	Delhi
24	America
13	USA

P.K

Functional Dependency :-

- The functional dependency is a relationship that exists between two attributes.
- It typically exists between the primary Key and non-primary Key attributes within a table.

$$X \rightarrow Y$$

- The left side of functional dependency is known as determinant, the right side of the production is known as dependent.

For example :-

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp-Address.

Here Emp-Id attribute can uniquely identify the Emp_Name attribute of Employee table because if we known the Emp-Id, we can tell that Employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id

Types :-

(i) Trival functional dependency
(ii) Non-trival functional dependency

i) Trivial functional dependence :

- $A \rightarrow B$ has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: $A \rightarrow A$, $B \rightarrow B$

Ex => Consider a table with two columns employee_id and employee_Name.

{Employee_id, Employee_Name} → Employee_id is a trivial functional dependency as employee_id is a subset of {Employee_id, Employee_Name}.

Also, Employee_id → Employee_id and Employee_Name → Employee_Name are trivial dependencies too.

(ii) Non-trivial functional dependencies :-

- $A \rightarrow B$ as a Non-trivial functional dependency if it is not a subset of A.
- When A intersection B is NULL, then $A \rightarrow B$ is called as ~~column~~ Complete non-trivial functional dependencies.

Ex => Consider a table with three column Emp_id, Emp_name and Emp_age.

Emp_id => Emp_name is a non trivial functional dependency becaus Emp_name is not a subset of Emp_id.

Similarly

Emp_id, Emp_name → Emp_age is non trivial because Emp_age is not a subset of Emp_id, Emp_name.

⇒ Fully functional dependency :-

ABC → D {D is fully functional dependent on ABC}

{D can not dependent on any subset of ABC}

BC → D
C → D
A → D

not possible because BC can not determine D
C can not determine D
A can not determine D

Only ABC determine D means D is fully functional dependent on ABC

Example

Id std	Name	Id prof	Grade
S_1	Pinky	P_2	5
S_2	Lucky	P_1	6

Id std, Id prof ⇒ Identifi-cation key

Id std, Id prof → Grade

fully functional dependent

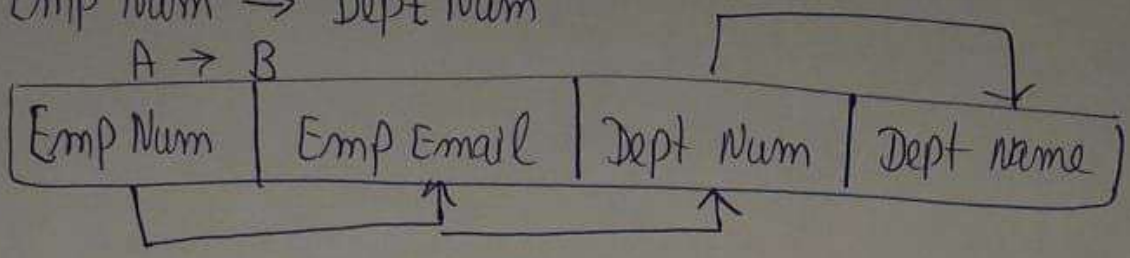
⇒ Transitive dependency

Consider attribute A, B & C
Where A → B & B → C if functional dependency are transitive also have the functional dependency : A → C

C is transitively dependent on A through B

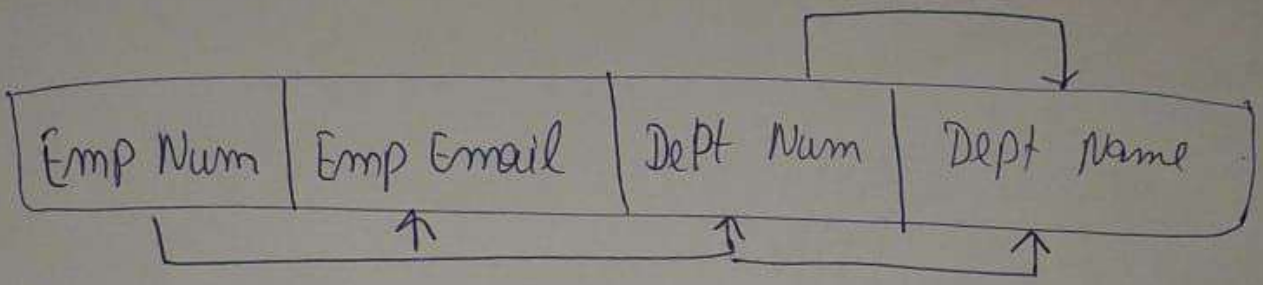
Eg: Emp Num → Dept Num

A → B



Dept Num → Dept Name

B → C



Emp Num → Dept Num A → B - Principal → teacher

Dept Num → Dept Name B → C teacher → std

Emp Num → Dept Name A → C Principal → std

⇒ Normalization :-

- Normalization is a method to breaking down a large, complex tables into smaller, related tables and define relationship between them.
- It reduce the redendency of data in a table also remove inconsistency problem and unwainted risk.

Advantage :

1> Maintain data integrity.

2> Reduce the structure of table.

3> Avoid null value.

(iv) Structuring of data, show that model is flexible and easy to use.

Types

1. First normal form
2. Second normal form
3. Third normal form
4. BCNF
5. Fourth normal form
6. Fifth normal form

1. First normal form :- (1NF)

- Data must be stored in a table with rows and column, and each column should contain individual values.

Project table

Sroll	SName	Course	Teacher
1	Ankit	C, C++	BCA, Mr. A
2	Ankush	Java, C	Mr. C, Mr. A
3	Neha	C++, Java	Mr. B, Mr.

After 1NF

Sroll	S Name	Course	Teacher
1	Ankit	C	BCA
1	Ankit	C++	Mr. A
2	Ankush	Java	Mr. C
2	Ankush	C	Mr. B
3	Neha	C++	Mr. A
3	Neha	Java	Mr. C

2. Second Normal form (2NF) :-

1) Must be in 1NF
2) All & non-Key attributes should be fully dependent on the primary Key (no partial dependencies).

Project table :-

Sroll	S Name	Teacher	Course
1	Ankit	Mr. A, Mr. B	C, C++
2	Ankush	Mr. C, Mr. A	Java, C
3	Neha	Mr. B, Mr. C	C++, Java

1NF table

Sroll	S Name	Teacher	Course
1	Ankit	Mr. A	C
1	Ankit	Mr. B	C++
2	Ankush	Mr. C	Java
2	Ankush	Mr. A	C

3	Neha	Mr. C	C++
3	Neha	Mr. B	Java

2 NF table

Primary Key

Sroll	S Name
1	Ankit
2	Ankush
3	Neha

Table 1 (Student)

F.K

Table 2 (Enrollment)

Sroll	Course	Teacher
1	C	Mr. A
1	C++	Mr. B
2	Java	Mr. C
2	C	Mr. A
3	C++	Mr. C
3	Java	Mr. B

(3) (3NF) Third Normal form :-

- Must be in 2NF
- Non Key attributes should not depend on other non-Key attributes

Table 1 : Student (Same as 2NF)

Sroll	S Name
1	Ankit
2	Ankush
3	Neha

Table 2: Courses

Course	Teacher
C	Mr. A
C++	Mr. B
Java	Mr. C

Table 3: Enrollments

Sroll	Course
1	C
1	C++
2	Java
2	C
3	C++
3	Java

4. Boyce Codd Normal form (BCNF) :-

- Must be in 3NF.
- If a table contain more than one composite key attributes and should be possible the one attribute of both composite key are matched.

E-Code	Name	P_Code	Hours
E_1	Raj	P_2	48
E_2	Ravi	P_5	100
E_3	Mohan	P_6	15
E_4	Shusil	P_2	250
E_4	Shusil	P_5	75
E_1	Ravi	P_5	40

E-Code + P-Code → Composite 1

↑ ↓

Name + P-code → Composite 2

Employee

P.K

E-Code	Name
E_1	Raj
E_2	Ravi
E_3	Mohan
E_4	Sushil

Project table

F.K

E-Code	P-Code	Hour
E_1	P_2	48
E_2	P_5	100
E_3	P_6	15
E_4	P_2	250
E_4	P_5	75
E_1	P_5	40

5. (4NF) Fourth Normal form :-

- Must be in BCNF or 3NF.
- We must ensure there are no multi valued dependencies, where one column is dependent on another, independent of other attributes.

6. (5NF) Fifth Normal form :-

- It is also called project join normal form.
- Must be in 4NF.
- If we decompose table further to eliminate redendency and when we rejoin the decompose table means Candidate Key we should not be lossing the original data.

Chapter - Transaction Management Section - B

- Transaction is a collection of operations that are executed as a single unit of work. Transaction do not violet any database ~~connection~~ constraint.
- Each transaction ensure that these operations are either all completed successfully or all rolled back.

⇒ ACID properties of transaction :-

- Transaction is a very small unit of program and it may contain several low level task.
- In DBMS transactions are defined four key properties called ACID properties.

(i) Atomicity :-

- A transaction must be completed entirely or not. If any part of a transaction fails the entire is rolled back.

Example :-

In a bank transfer if money is debited from the account but can't be credited to other account, the entire transaction should be failed or canceled.

(ii) Consistency :- Ensure that a transaction brings the database

from one consistent state to another maintaining all predefined rules and constraints.

Example

If there is a constraint that account balance cannot be negative a transaction should never violate this rule.

(iii) Isolation :-

- Ensures that transactions are executed independently without interface from other transactions.

Example

Two transactions trying to update the same account balance. Simultaneously should not impact each other.

(iv) Durability :-

- Once a transaction is committed the changes are permanently saved in the database even in the case of system failure.

Example :

Once a bank balance transaction is successful the updated balance should be saved even, if the system crashes immediately afterward.

⇒ Transaction state :-

- In DBMS state of transaction refers to its current status

within it's life cycle.

- Transactions go through several states from the moment they start until they either complete successfully or terminate

The transaction state can be classified into 5 different states :-

- ↳ Active state
- ↳ Partially committed state
- ↳ Committed state
- ↳ Failed state
- ↳ Aborted state

Active state :-

Transaction is in progress, executing its operations.

Partially Committed :-

Transaction has completed its final step and is ready to commit changes.

Committed :-

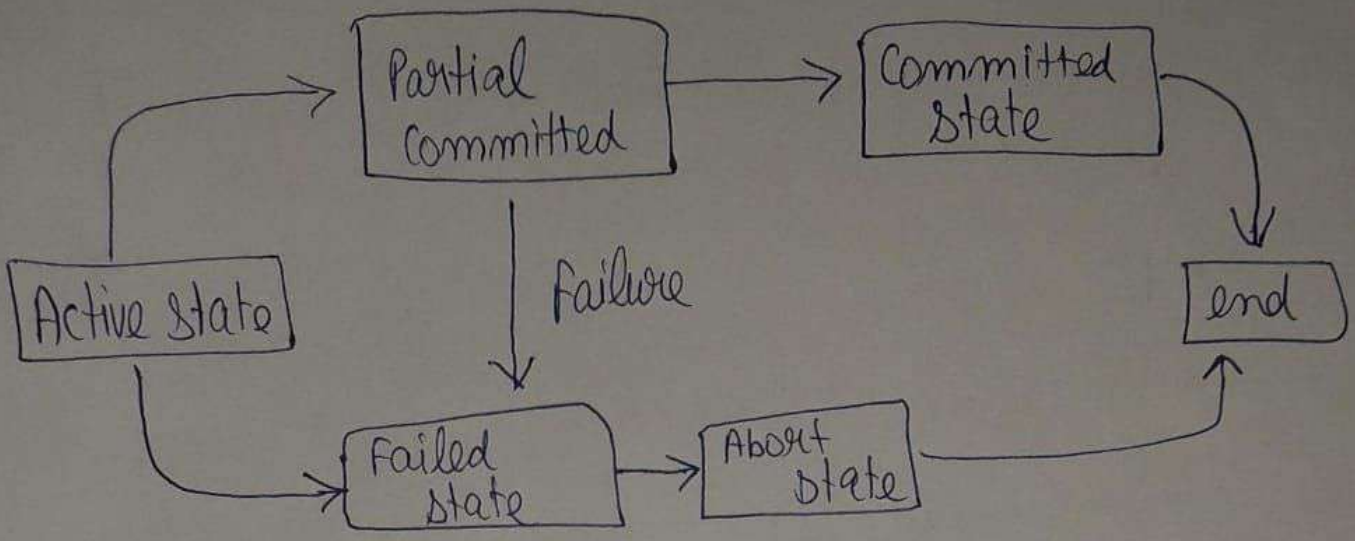
Transaction has secussfully saved changes to the database permanently.

Failed :-

An error has occured and the transcation can't continue

bortted -

Transaction has been rolled back and all changes undone it may be terminated or restarted.



⇒ Concurrency control :-

- Multiple users can access and use the same database at one time, which is known as the concurrent execution of the database.
- It ensures that database transactions are performed concurrently and accurately.
- It confirms that produce correct results without violating data integrity of the respective database.

Example

- Two people buy a movie ticket for the same movie and the same show time.
- There is only one seat left, for the movie show in that particular theature.
- Concurrency problem occur.

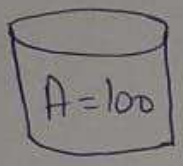
Without concurrency control it is not possible to buy ticket. It provides a ticket to the buyer who has completed the transaction process first.

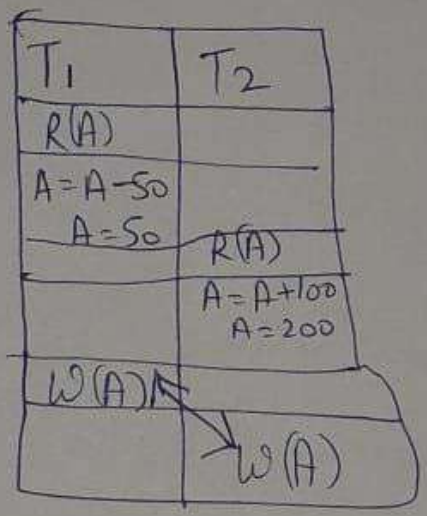
⇒ Concurrency control problems :-

1. Lost update (W - W Problem) :-

- It occur when multiple transactions select the same row and update the row and update the row based on the value selected.

Example :

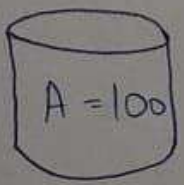


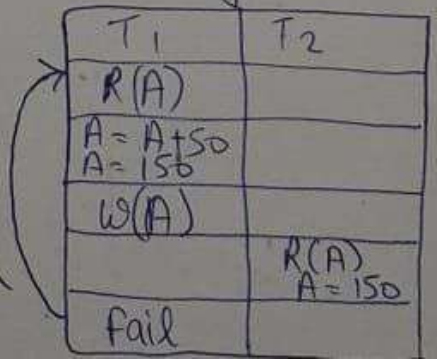


2. Dirty Read problem :-

- When one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction.

Example :

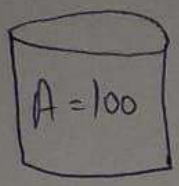




3. Unrepeatable Read Problem :

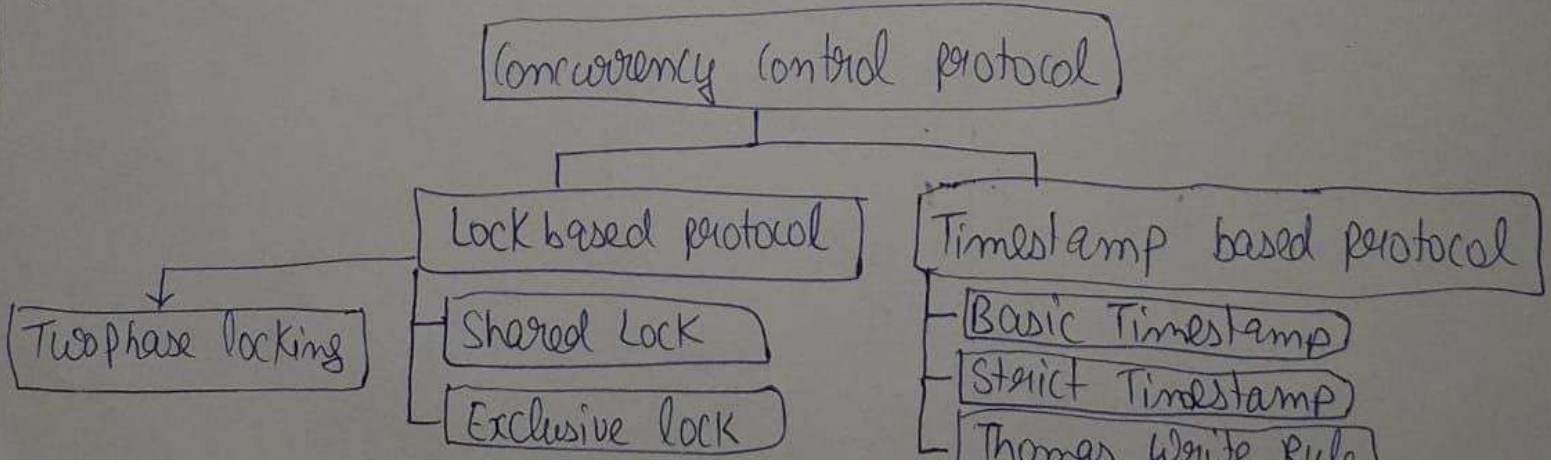
- Also known as Inconsistent Retrievals problem that occurs when in a transaction, two different values are read for the same database item.

Example :



T_1	T_2
R(A) A=100	
	R(A)
	A = A+100 A=200
	W(A)
R(A) A=200	

- Concurrency control is the working concept that is required for controlling and managing the concurrent execution of database operations.
- It avoiding the inconsistencies in the database.
- The concurrency control protocols ensures the atomicity, consistency, isolation, and durability of the concurrent execution of the database transactions.



Two phase locking :-

The two phase locking protocol is a concurrency control mechanism used in database management system to ensure serializability of transaction

It guarantees that transactions execute in a sequential manner preventing data inconsistency and race conditions.

Phases of 2-Phase locking protocol :-

1. Growing phase :-

- A transaction acquires all the necessary locks (read or write).
- No locks can be released in this phase.
- This phase ends when the transaction obtains its last required lock (lock point).

2. Shrinking phase :-

- A transaction releases locks but cannot acquire new ones.
- Once a transaction starts releasing locks, it must release all remaining locks before completion.

Example :-

Consider two transactions T_1 and T_2 accessing a shared database.

Without 2PL (Data Inconsistency) :-

① T_1 reads a balance of ₹500.

T_2 also reads the same balance ₹500.
T_1 withdraws ₹100 and updates the balance to ₹400.
T_2 deposits ₹200 but updates balance to ₹700 (ignoring T_1's withdrawal.
The final balance is ₹700, but it should be ₹600.

With 2PL (Ensuring correct execution order):-

1. T_1 acquires an exclusive lock on the balance before reading.
2. T_2 must wait until T_1 releases the lock.
3. T_1 updates the balance and releases the lock.
4. T_2 then reads the updated balance and deposits ₹200
5. The final balance is correctly updated to ₹600.

Thus, 2PL prevents incorrect updates and ensures transaction serializability.

Types of Locks in 2PL :-

1. Shared lock (S-Lock):-
   - Allows multiple transactions to read the same data.
   - No transaction can modify the data while holding a shared lock.
2. Exclusive lock (X-Lock):-
   - Only one transaction can modify the data at a time.
   - Other transactions must wait until the lock is released.

Timestamp ordering protocol :-

The timestamp ordering protocol is used to order the transactions based on their timestamps.

The order of transaction is nothing but the ascending order of the transaction creation.

The priority of the older transaction is higher that's why it executes first.

To determine the timestamp of the transaction, this protocol uses system time, logical counter or unique value.

The timestamp ordering protocol also maintains the timestamp of last read and write operation on a data.

Transactions	T_1	T_2	T_3
Timestamp =	10	20	30
Priority = (High to low)	(older)		(Younger

Basic timestamp ordering protocol functions :-

1. TS (Ti) : Indicate timestamp of transaction Ti

Example : T1 : 10, T2 : 20, T3 : 30

2. RS - TS(X) : Last (latest) transition no. which perform read successfully.

Example : R_TS(A) = 30

Transactions	T_1	T_2	T_3
Timestamp	10(0)	20	30(Y)
	R(A)		
		R(A)	
			R(A)

R_TS(X)

)_TS(X) : Last (latest) transition no. which perform write successfully.

Example : W_TS(A) = 20

Transactions	T_1	T_2	T_3
Timestamp	10(0)	20	30(Y)
	W(A)		
			W(A)
		W(A)	

W_TS(X)

Timestamp Ordering Rules: Read ()

Check the following condition whenever a transaction Ti issues a Read (X) operation:

) If W_TS(A) > TS(Ti) then operation rejected & rollback. (Not Allow)

) Otherwise execute R(A) operation. Set R_TS(A) = Last (R_TS(A), TS(Ti)). (Allow)

a. Not allow

T_1	T_2
10(0)	20(Y)
	W(A) A=30
R(A) A=30	
	Fail

A=30

=30

After Rollback A=40

Rollback

b. Allow

T_1	T_2
10(0)	20(Y)
W(A) A=10	
Commit	
	R(A) A=10

A=10

R_TS(A) = 20

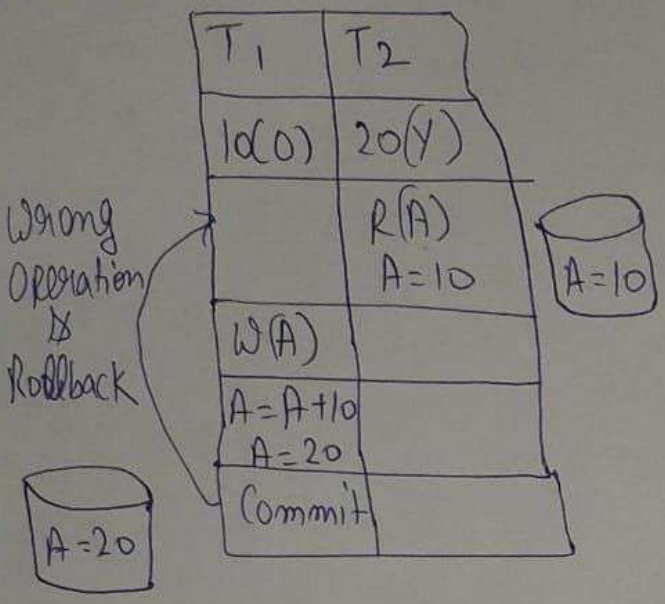
: Check the following condition ~~which~~ whenever a transaction Ti issues a Write (X) operation :-

a: If R_TS(A) > TS(Ti) then operation rejected & rollback. (Not allow)

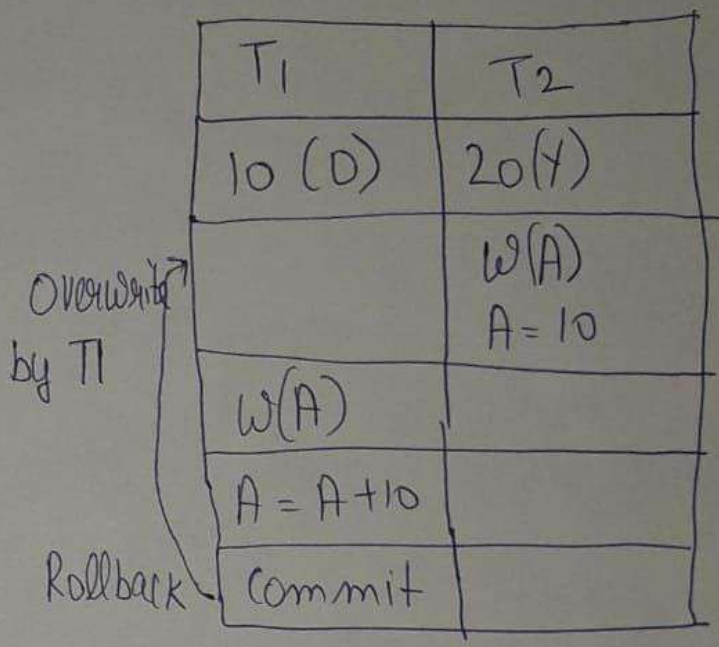
b: If W_TS(A) > TS(Ti) then operation rejected & rollback. (Not allow)

c: Otherwise execute Write(A) operation. Set W_TS(A) = last(TS(Ti))). (Allow)

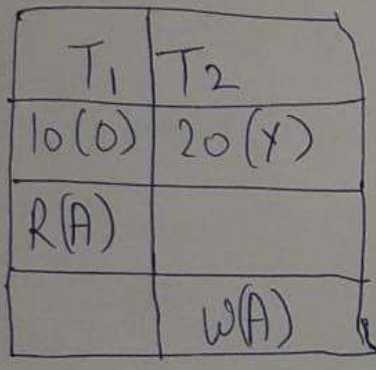
a. Not Allow

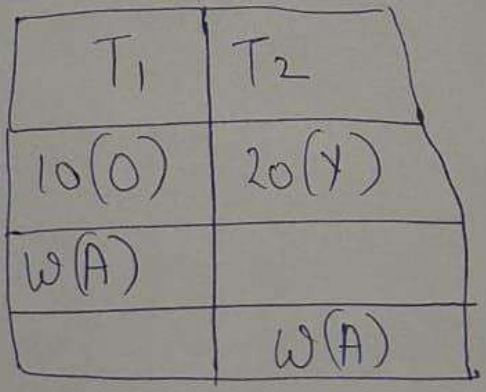


b. Not Allow



c. Allow





⇒ Serializability :-

- Serializability in DBMS, ensure the execution of concurrent transactions is consistent and results in the same outcome as if transactions were executed serially).

(one after another)

This concept is crucial for maintaining data integrity and consistency in multi user database environments.

Types :-

↳ Conflict Serializability

↳ View

① Conflict - It occurs when two transactions try to read / write same data item.

② View - A schedule is view, if the final outcome (read/write operations) is the same as in or serial schedule.

Chapter - Database Recovery Section - B

⇒ Database Recovery :-

- Database recovery ensures data consistency and durability even when unexpected failure occur. The primary goal is to restore the database to the last correct state before the failure.

Recovery techniques rely heavily on logs (a sequential record of all database changes) and checkpoints (periodic snapshots of the database state.

Key objectives of Recovery :

- Undo (Rollback): Reverse the changes of failed transactions.

Redo (Rollforward): Reapply changes of committed transactions that were not saved to disk.

Minimize data loss and ensure ACID properties (Atomicity, Consistency, Isolation, Durability).

Recovery Techniques :-

- Log based recovery techniques

The two main log-based recovery techniques:

A. Deferred update (No Undo/ Redo Technique) :

- Mechanism:

(i) Transaction modifications are not written to the database immediately.

(ii) All updates are recorded in log buffers first.
(iii) Only after the transaction commits, the changes are applied to the database from the log.

(iv)

- Recovery Process :-

(i) No Undo needed:

If a transaction fails before commit, it never modified the database, so no rollback is required.

(ii) Redo Needed:

If a transaction ~~fails~~ ~~before~~ committed but its changes were not saved to disk, the system reapplies the changes from the log.

Example :

- Transaction T_1 (committed) :- Updates A = 800, B = 1700 must be redone if not yet saved
- Transaction T_2 (not committed) :- Ignored (no changes were written to disk)

Advantages :-

- Simple recovery (no need to undo incomplete transactions).
- Suitable for short transactions with few updates.

Disadvantages :-

- Requires large buffer space to hold uncommitted changes.

- Not efficient for long running transactions.

B. Immediate Update (Undo / Redo or Undo / No Redo Technique)

- Mechanism :-

(i) Changes are immediately written to the database (before commit) but logged first (Write Ahead logging - WAL).

(ii) The log stores both old and new values (e.g, [writes, T_1, A, 1000, 800]).

- Recovery process :-

(i) Undo :-

If a transaction fails before commit, its changes are rolled back using old values from the log.

(ii) Redo :-

If a transaction committed but some changes were lost, they are reapplied from the log.

- Variants :-

(i) Undo/No Redo: All changes are forced to disk before commit (rarely used).

(ii) Undo / Redo : Common method where some changes may still be in memory at commit time.

- Example :-

(i) Transaction T1 (committed): If updates were not fully saved,

redo them (A = 800, B = 1700).

ii) Transaction T2 (not committed): Undo its changes (C = 2000 restored).

Advantages :-

i) Allows early disk writes, improving performance.
ii) Works well for both short and long transactions.

Disadvantages :-

(i) More complex recovery (requires both undo and redo operations).

⇒ Recovery in different Environments :-

A. Single User Environment SUE) :-

- Only one transaction executes at a time.
- Recovery steps:
  1. Scan the log from the last checkpoint.
  2. Redo all (commands) committed transactions & if changes were lost.
  3. Ignore / Restart incomplete transactions (Deferred update) or undo them (Immediate update).

B. Multiuser (concurrent) Environment :-

- Multiple transactions execute simultaneously, requiring concurrency control (eg. strict 2PL)
- Recovery steps:

1. Checkpointing: Identifies which transactions were active

at the time of failure.

2. Redo: Reapply committed transactions in the order they were logged.

3. Undo: Roll back uncommitted transactions in reverse order (to handle dependencies).

Example

- T1 (committed before checkpoint): Ignored already saved.
- T3 (committed after checkpoint): Redo its updates.
- T4 (not committed): Undo its changes.

⇒ Key Concepts in Recovery :-

A. Checkpoints

- Periodic snapshots of the database state.
- Reduces recovery time by limiting log scanning to the most recent checkpoints.

B. Write-Ahead logging (WAL)

- Rule: Log records must be written to disk before the actual database changes.
- Ensures that recovery can reconstruct changes even if the system crashes mid-transaction.

⇒ Shadow paging :-

- Shadow paging is a technique used in database management

Systems (DBMS) for crash recovery and maintaining data consistency.

Shadow paging is a method where the system keeps two version of the database pages:-

- Current Page table :- The page table that users are working with.
- Shadow Page table :- A copy of the page table that reflects the database state before the transaction started.

How it works:

1. When a transaction begins, a shadow page is created.
2. All changes are made to a new copy of the pages.
3. The original data remains unchanged in the shadow pages.
4. When the transaction is successfully completed:
   - The current page table becomes the new database state.
   - The shadow page table is discarded.

Advantages :-

- Easy to implement.
- No need for complex logging (like in Write-Ahead logging)
- Good for simple, smaller databases.

Disadvantages :-

- Uses more storage (due to page copying).
- Not suitable for large scale databases.
- Slower performance when many changes are made.

What are data constraints in Oracle? Write down all types of constraints on row and column levels with their commands?

Ans

- Constraints in Oracle are rules applied to columns in a table to ensure validity, accuracy and integrity of the data.
- Oracle supports several types of constraints to restrict the type of data that can be stored in tables.

These constraints can be applied at two levels:

- Column level: Directly with the column definition.
- Table (Row-level): Defined after all column definitions, useful for multi column constraints like composite keys.

Types of constraints in Oracle :-

1. Not Null constraint :-

- Ensures that a column cannot store NULL values.
- Applied only at the column level.

```
CREATE TABLE students (
  student_id NUMBER NOT NULL,
  name VARCHAR2(50) NOT NULL
);
```

2. Unique Constraints :-

- Ensures that all values in a column are distinct (no duplica

Column - level

```
CREATE TABLE users (
    email VARCHAR2(100) UNIQUE
);
```

Table - level

```
CREATE TABLE users (
    user_id NUMBER,
    email VARCHAR2(100),
    CONSTRAINT unique_email UNIQUE
                (email)
);
```

3. Primary Key constraint :-

- Uniquely identifies each row in the table.
- It combines UNIQUE + NOT NULL
- Only one primary key allowed per table.

Column - level

```
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    name VARCHAR2(50)
);
```

Table - level

```
CREATE TABLE enrollment (
    student_id NUMBER,
    course_id NUMBER,
    CONSTRAINT pk_enroll PRIMARY
    KEY (student_id, course_id)
);
```

4. Foreign Key constraint :-

- Creates a relationship b/w two tables.
- Ensure the value in a column matches a value in another table's primary key.

```
CREATE TABLE departments (
 dept_id NUMBER PRIMARY KEY,
 dept_name VARCHAR2(50)
);

CREATE TABLE Employees (
 empl_id NUMBER PRIMARY KEY,
 emp_name VARCHAR2(50),
 dept_id NUMBER,
CONSTRAINT fk_dept FOREIGN KEY(dept_id)
 REFERENCES departments(dept_id)
);
```

5. Check Constraint :-

- Ensures that all values in a column meet a specific condition.

```
CREATE TABLE orders (
   order_id NUMBER,
   quantity NUMBER,
   CONSTRAINT check_qty CHECK (quantity > 0)
 );
```

- It is used to restrict values such as only positive numbers, specific ranges, etc.

6. Default Constraint :-

- Assigns a default value to a column when no value is specified.

```
CREATE TABLE Customers(
  Customer_id NUMBER,
  Country VARCHAR2(50) DEFAULT 'India'
);
```

- It is useful for assigning default values like Country, States etc.

⇒ Explain the client - server architecture of Oracle?

Ans

- Client - server architecture is a type of network architecture where multiple clients request and recieve services from a centralized server.
- It is the most common in the web applications, databases and networked (model used) systems.
- This architecture divides tasks and workloads between servic providers and servica requesters.

→ Components of client server Architecture :-

1. Client:

- A client is a device (like a computer, smartphone, or application) that requests services or data from a server.
- Example: A web browser is a client that requests web pages from

a web server.

2. Server :-

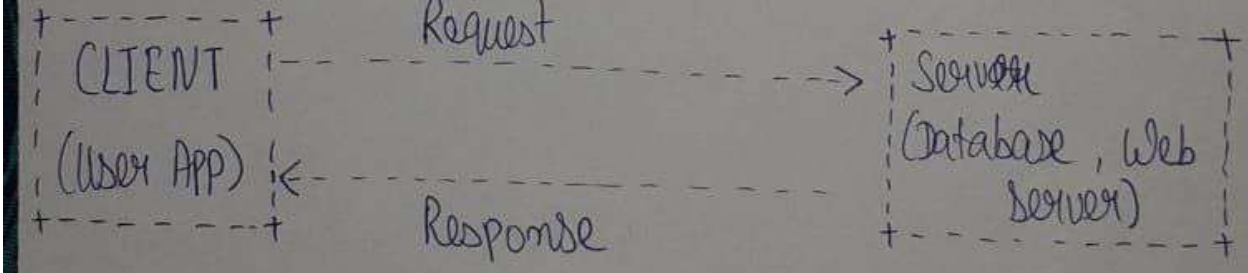
- A server is a high performance system that processes client requests and sends back the required information or services.
- Example : A web server hosts websites and responds to browser requests.

3. Network :-

- The connection medium (usually the internet or LAN) that allows communication between client and server.

How it works :

1. The client sends a request to the server (eg asking for a webpage)
2. The server processes the request performs necessary actions like retrieving data, and sends a response back.
3. The client recieves and displays the response to the user.



→ Types of client server Architecture :-

1. 1-Tier Architecture :

- Client and server are on the same machine. example : MS access

2. 2-Tier Architecture :

- Client directly communicates with the database server.
- Example: Client application + oracle DB.

3. 3-Tier Architecture :

- Includes client, application server, and database server.
- Example: Web browser (client) → Web server → Database server.

Advantages of client server architecture:

1. Centralized Resources:

- Data and files are stored on a central server, making management easier.

2. Security:

Easier to apply and manage security policies at the server level.

3. Scalability:

- Servers can be upgraded to handle more clients.

4. Data Recovery:

- Backup and recovery is easier from a central server.

5. Performance :-

Servers are powerful systems designed for handling many requests efficiently.

Disadvantages :-

- If the server goes down, clients cannot access services.
- Requires investment in powerful server systems and maintenance.
- Performance dependency depends on the speed and reliability of the network.

⇒ Write a short note on oracle functions ?

Ans

- In Oracle, functions are built in tools used in SQL to perform operations on data. They take input, process it and give a result.
- These functions are commonly used in SELECT, WHERE, ORDER BY and HAVING clauses.

Types of oracle functions :

1. Single-Row functions

- It works on one row at a time.
- Return one result for each row.
- Common types:

(i) String functions

- UPPER (name) → Converts text to uppercase.
- LOWER (name) → Converts text to lowercase
- LENGTH (name) → Return number of characters

(ii) Number functions

- ROUND(salary, 2) → Rounds the number to 2 decimal places.
- MOD(10, 3) → Returns the ~~current date~~ remainder (1).

(ii) Data functions

- SYSDATE - Returns the current date
- ADD_MONTHS(SYSDATE, 2) → Adds 2 months to the current date.

2. Group functions (Aggregate functions)

- Works on multiple rows together.
- Return one result for a group of rows.
- Examples:
  - SUM(salary) → Adds all salary values
  - AVG(marks) → Finds average marks
  - MAX(age) → Finds highest age
  - MIN(age) → Finds lowest age
  - COUNT(*) → Counts total rows

⇒ Triggers :-

- A trigger is a special kind of stored program in Oracle that automatically runs when a specific event happens in the database like an INSERT, UPDATE or DELETE operation on table.
- Triggers helps in automating actions, checking data or logging changes without manual work.

Why use Triggers?

- To automatically perform actions when data changes.
- To enforce business rules.
- To maintain audit trails.
- To check or restrict data changes.

Types of Triggers :-

1. Before trigger :-

- Executes before the data is changed.
- Used to validate or modify data.

2. After trigger :-

- Executes after the data is changed.
- Often used for logging or backup.

3. Row-level trigger :-

- Executes once for each row affected