DBMS TERMINOLOGY

What is a database system?

A Database Management System (DBMS) is system software used to manage the organization by storage, access, modification and integrity of data in a structured database.
A DBMS makes it possible for end users to create, read, update and delete data in a database systematically.
The DBMS essentially serves as an interface between the database and end users, ensuring that data is consistently organized and remains easily accessible.

Why do we need DBMS?

Manages big amounts of data
A database stores and manages a large amount of data on a daily basis. This would not be possible using any other tool such as a spreadsheet as they would simply not work.

Improved Data Sharing and Data Security
Proper database management systems help increase organizational accessibility to data, which in turn helps the end users share the data quickly and effectively across the organization.

Accurate
A database is pretty accurate as it has all sorts of built- in constraints, checks etc. This means that the information available in a database is guaranteed to be correct in most cases.

Easy to update data
In a database, it is easy to update data using various Data Manipulation languages (DML) available. One of these languages is SQL.

Effective Data Integration
Implementing a data management system promotes an integrated picture of an organization's operations. It becomes easy to see how processes in one segment of the organization affect other segments.

Easy to research data
It is very easy to access and research data in a database. This is done using Data Query Languages (DQL) which allow searching of any data in the database and performing computations on it.

File System vs DBMS

Base	DBMS	Flat File System
Definition	DBMS is a collection of interrelated data and software programs to access those data.	Flat file system stores data in a plain text file. Here, the records are specified in a single line.
Data Redundancy	There is no problem of data redundancy.	There is a main problem of data redundancy.
Cost	DBMS software is very costly and regular updates make it costly.	Flat file are cost effective
Use	Mostly, large organizations use DBMS who can afford it and have a large number of clients and employees to be managed.	Small organizations use it as it is cost effective and who have to deal with a small number of clients and employees.
Views	Views are created and employees can't see all information available, hence there is security.	Any information can be seen by anyone, hence there is no security.

INSTANCE

The data stored in a database at a particular moment of time is called an instance of the database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, let's say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow so the instance of the database tomorrow will have 200 records in the table.

SCHEMA

Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

SUBSCHEMA

subschema is a subset or proper subset of the schema and inherits the same property that a schema has. The plan (or scheme) for a view is often called subschema. It gives the users a window through which he or she can view only that part of the database, which is of interest to him/her.

What is a data administrator?

A **data administration** (also known as a database administration manager, data architect, or information center manager) is a high level function responsible for the overall management of data resources in an organization. In order to perform its duties, the DA must know a good deal of system analysis and programming.

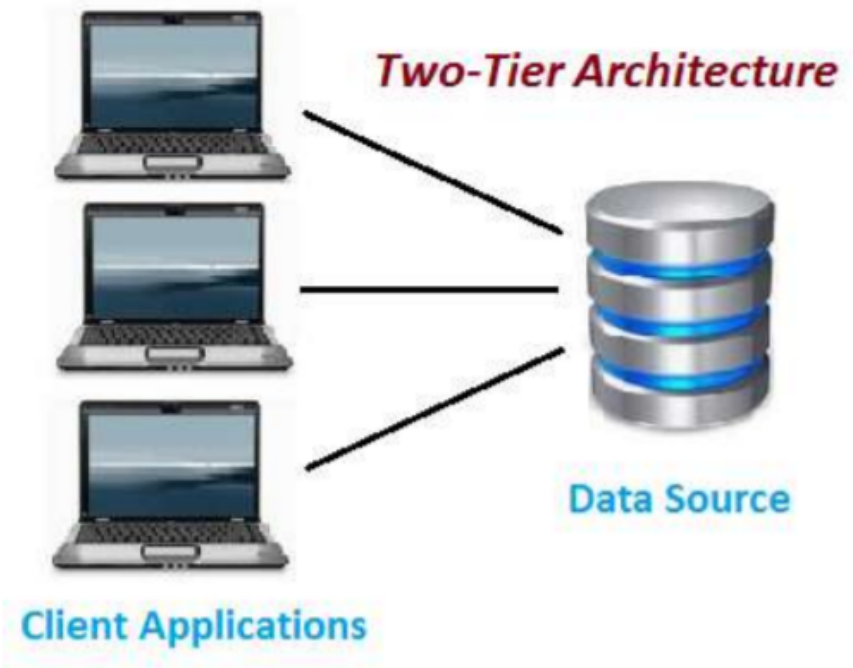
Functions of a database administrator

1. Selection of hardware and software
    - Keep up with current technological trends
    - Predict future changes
    - Emphasis on established off the shelf products
2. Managing data security and privacy
    - Protection of data against accidental or intentional loss, destruction, or misuse
    - Firewalls
    - Establishment of user privileges
    - Complicated by use of distributed systems such as internet access and client/ server technology.

Database Architecture

Two Tier Architecture

DBMS architecture 2 layer architecture Client-Server The client that runs the application and the server that handles the database back-end Multiple users able to access the DB simultaneously. Server is processing data while the client for business logic and presentation.



Two parts:
1) Client Application (Client Tier)
2) Database (Data Tier)

On client application side the code is written for saving the data in the SQL server database

Advantages:

- Easy to maintain and modification is bit easy
- Communication is faster

Disadvantages:

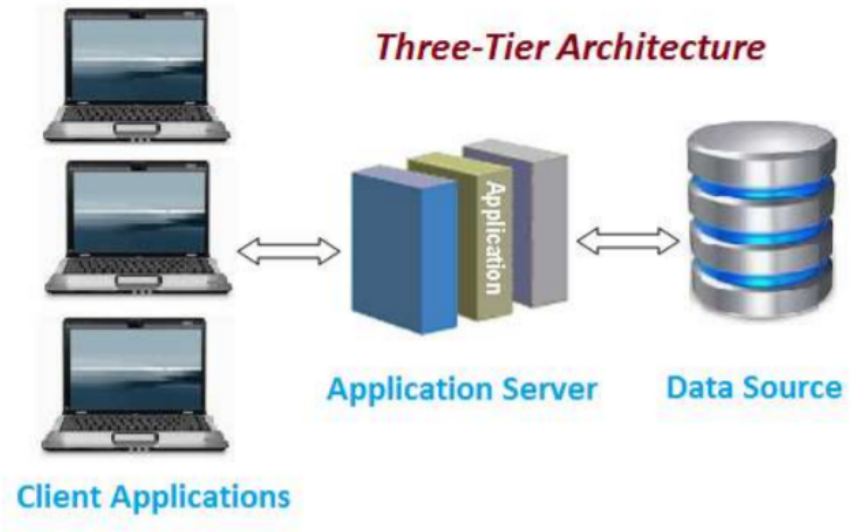
Performance will be degrade upon increasing the users
Cost-ineffective

Three Tier Architecture

- These Applications run on the Traditional Client/Server Model But from an Application server.
- Client only Displays the GUI and data, but has no part in producing results
- Database Server Serves to few Connections

Three layers:

1) Client layer
2) Business layer
3) Data layer



Client Layer
Contains the UI part of our application.
This layer is used for the design purpose where data is presented to the user or input is taken from the user

Business layer
All business logic written like validation of data, calculations, data insertion etc.
This acts as an interface between Client layer and Data Access Layer.
Make communication faster between client and data layer.

Data layer
Actual database comes in the picture.
Contains methods to connect with the database and to perform insert, update, delete, get data from the database based on our input data.

Advantages

- Performance :Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
- Scalability: Each tier can scale horizontally
- Better Reuse
- High degree of flexibility in deployment platform and configuration
- Improve Data Integrity
- Improved Security – Client is not direct access to the database.
- Easy to maintain and modification is bit easy, won't affect other modules

Interview Questions

Q 1. What is DBMS used for? (TCS)
DBMS, commonly known as Database Management System, is an application system whose main purpose revolves around the data. This is a system that allows its user to store the data, define it, retrieve it and update the information about the data inside the database.

Q 2. Why is the use of DBMS recommended? Explain by listing some of its major advantages.(WIPRo)
Some of the major advantages of DBMS are as follows:

- Controlled Redundancy
- Data Sharing
- Backup and Recovery Facility
- Enforcement of Integrity Constraints
- Independence of data

Q 3.What is Schema? (Amazon)
Schema is a structural view of a database. Moreover, a schema diagram refers to representing a database as a diagram. It represents the relationship between the tables. However, it does not show the data present in the tables.

Q 4.What is Instance? (Adobe)
An instance of the database is the data in the database at a specific moment of time. The database schema defines the variable declarations in tables and the values of these variables at a specific time are called the instance of the database.

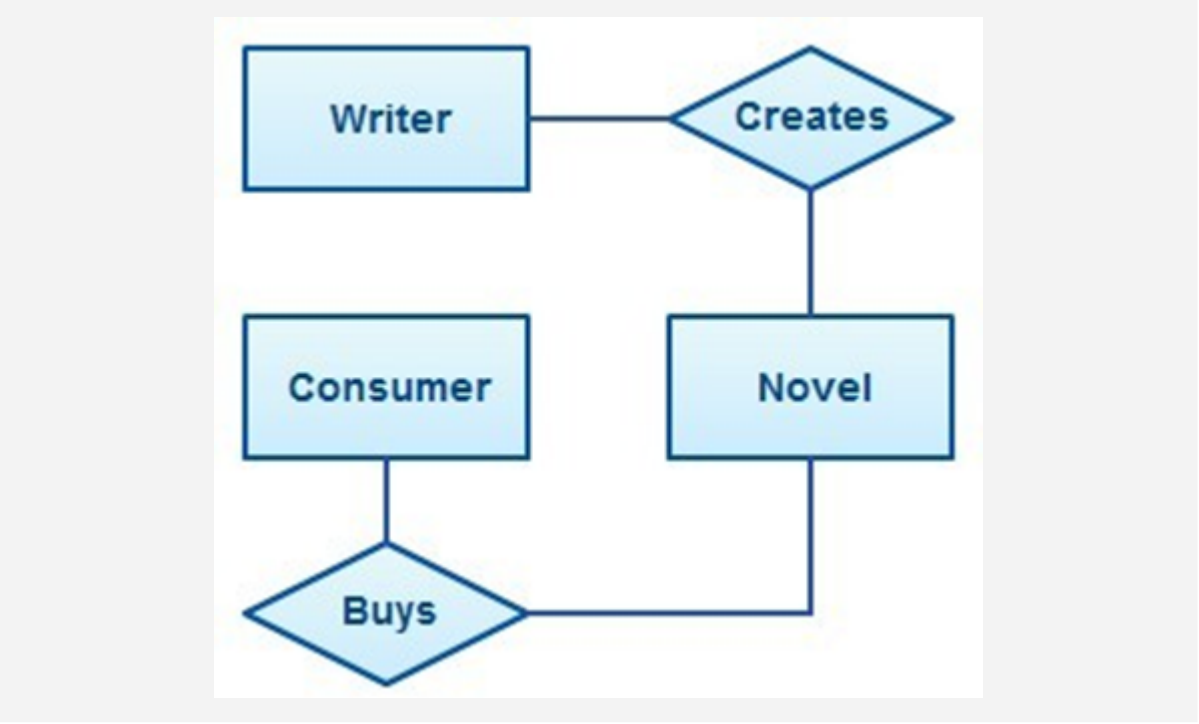
Q 5.Explain the three layers of 3 tier architecture.
Three layers:
1) Client layer
2) Business layer
3) Data layer

ER MODEL

Introduction to ER Models

What are ER Models
An Entity Relationship Diagram (ERD) is a visual representation of different data using conventions that describe how these data are related to each other.

For example, the elements writer, novel, and consumer may be described using ER diagrams this way:



The elements inside rectangles are called **entities** while the items inside diamonds denote the **relationships** between entities.

ER Diagram
ER-modeling is a data modeling technique used in software engineering to produce a conceptual data model of an information system.
Diagrams created using this ER-modeling technique are called Entity-Relationship Diagrams, or ER diagrams or ERDs.

Entity

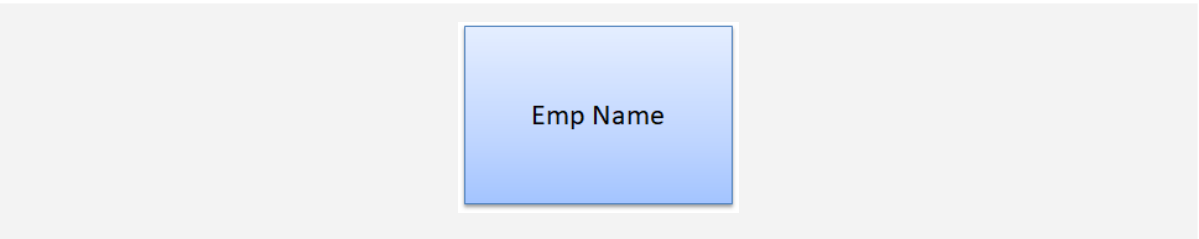
It is a collection of objects. An entity is an object that is distinguishable from other objects by a set of attributes. An entity may be an 'object' with a physical existence. This is the basic object of the E-R Model, which is a 'thing' in the real world with an independent existence.

For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Types of entity
Entities based on their characteristics are classified as follows:

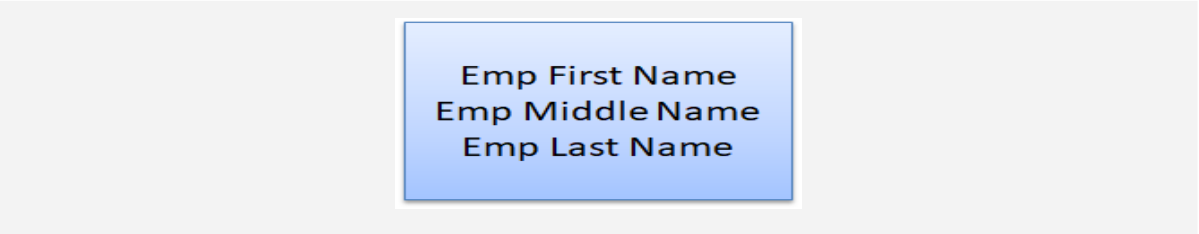
1. Strong Entities
2. Weak Entities
3. Recursive Entities
4. Composite Entities

Strong Entities
An entity that exists independently of other entity type



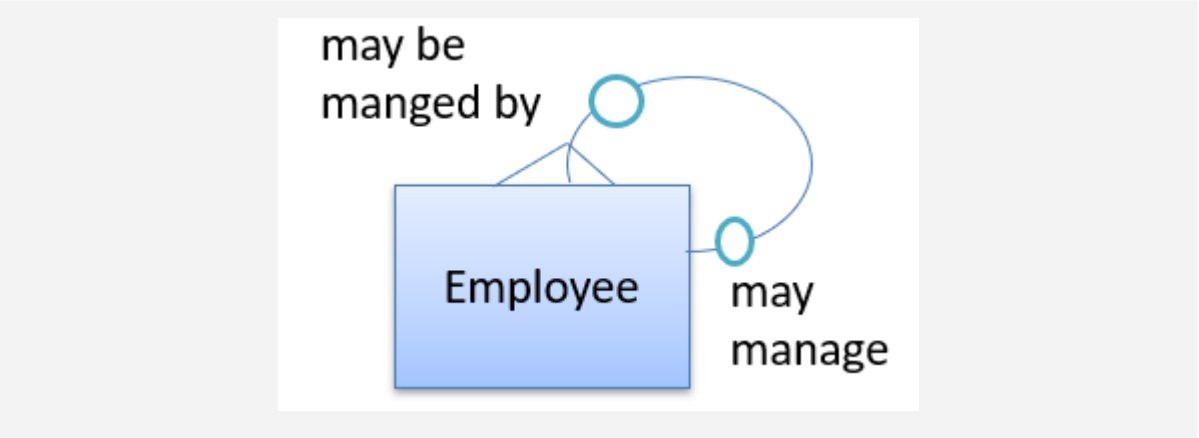
Weak Entities

An entity type whose existence depends on some other entity type.



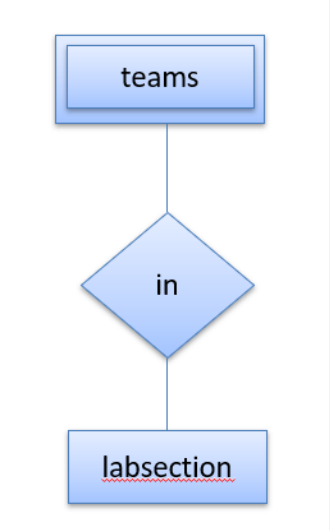
Recursive Entities

A recursive entity is one in which a relation can exist between occurrences of the same entity set. This occurs in a unary relationship. Let us take the example of an employee who is also a manager. But a manager is also an employee, whose details will be held in the employee entity.



Composite Entities

If a Many to Many relationship exists we must create a bridge entity to convert it into 1 to Many. Bridge entity composed of the primary keys of each of the entities to be connected. The bridge entity is known as a composite entity. A composite entity is represented by a diamond shape within a rectangle in an ER Diagram.



Entity Sets

An entity set is a collection of similar types of entities. An entity set may contain entities with attributes sharing similar values.

For example, a Students set may contain all the students of a school;

Likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Weak Entity

A weak entity is an entity that depends on the existence of another entity.

In more technical terms it can be defined as an entity that cannot be identified by its own attributes.

It uses a foreign key combined with its attributes to form the primary key.

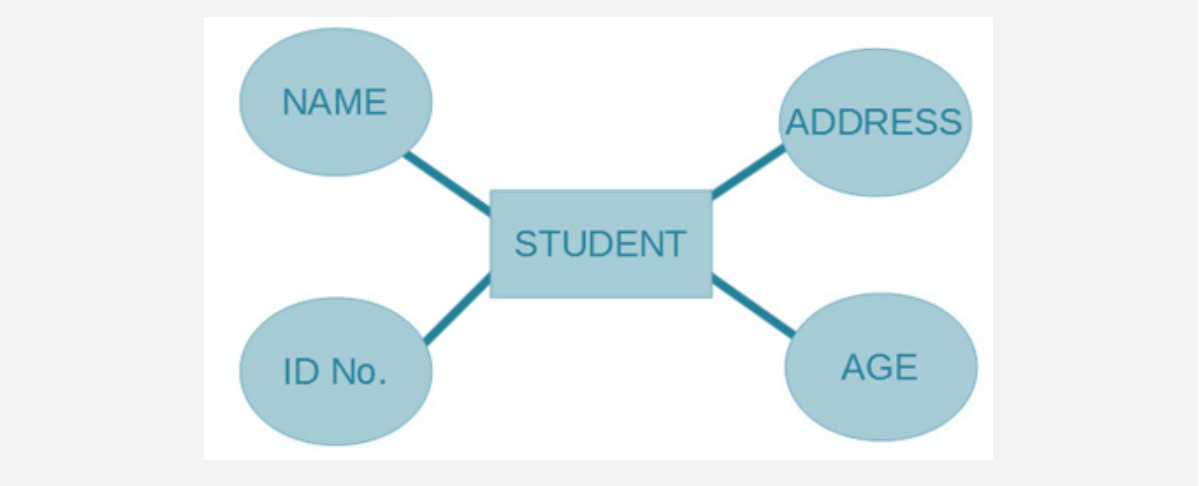
An entity like order item is a good example for this.

The order item will be meaningless without an order so it depends on the existence of the order.

Attribute

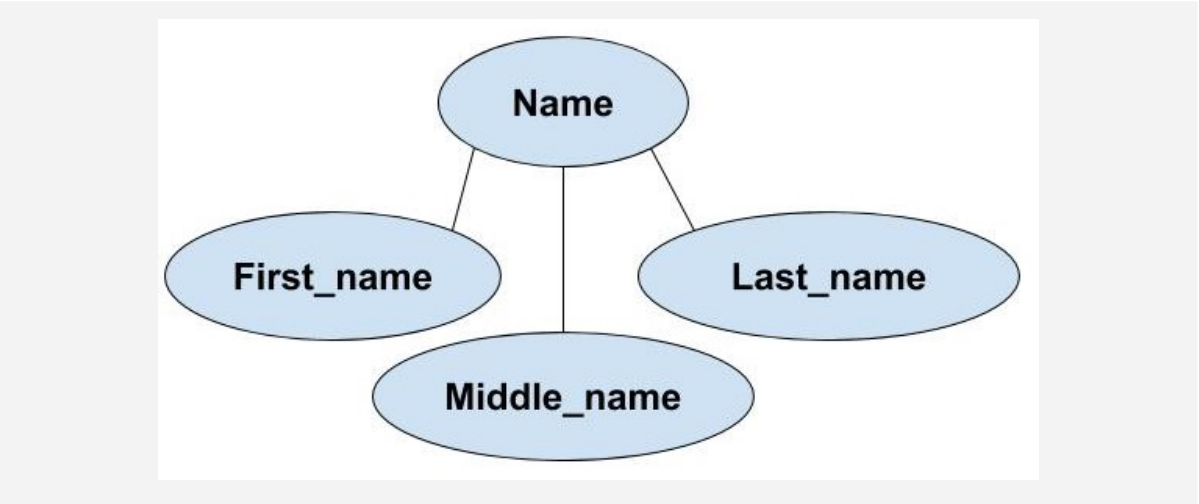
Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes.

For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



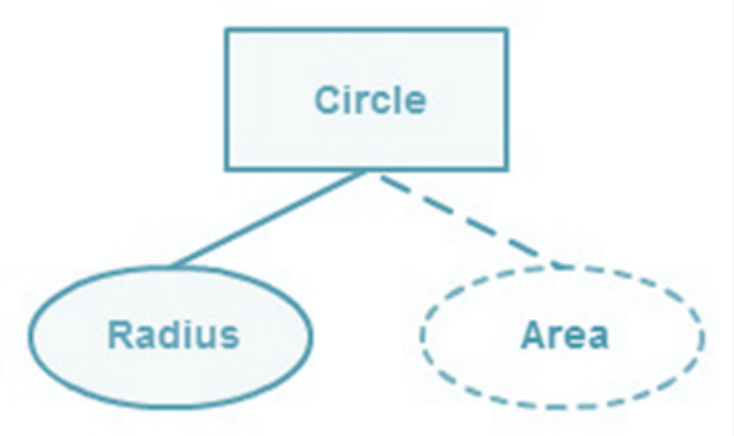
Simple attributes:
Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

Composite attribute :
If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is composite attributes are represented by eclipses that are connected with an eclipse.Composite attributes are made of more than one simple attribute.
For example, a student's complete name may have first_name and last_name.



Derived attribute :
Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.

For example, average_salary in a department should not be saved directly in the database, instead it can be derived.
For another example, age can be derived from data_of_birth.
For example for a circle the area can be derived from the radius.



Multi-valued attribute :

Multi-valued attributes may contain more than one value.

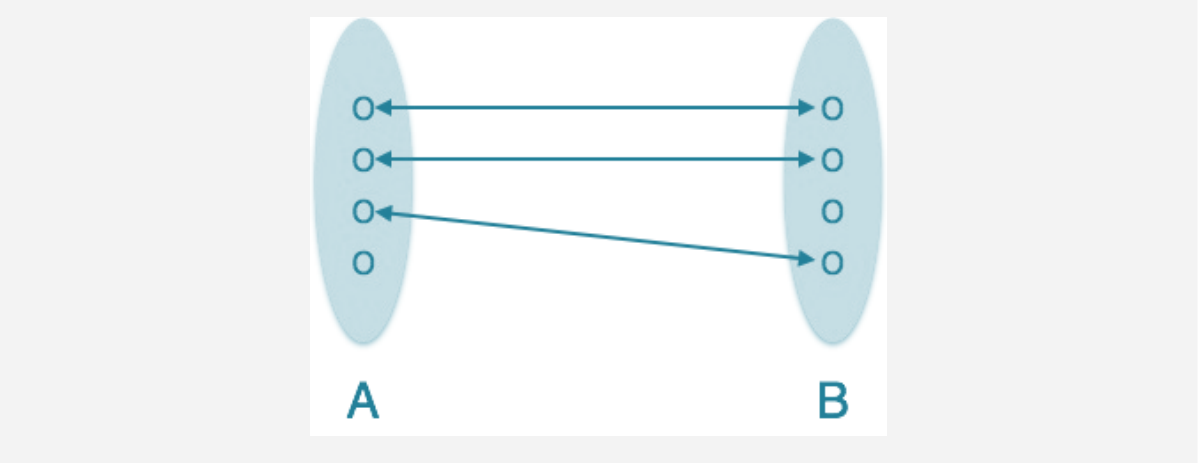
For example, a person can have more than one phone number, email_address, etc.

For example a teacher entity can have multiple subject values.

Relationships

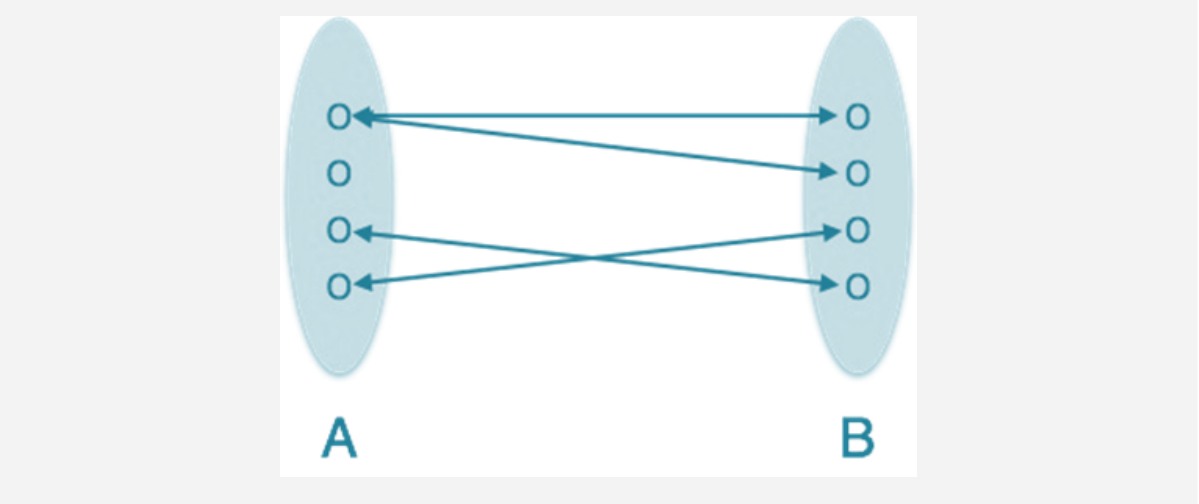
One-to-one :

One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



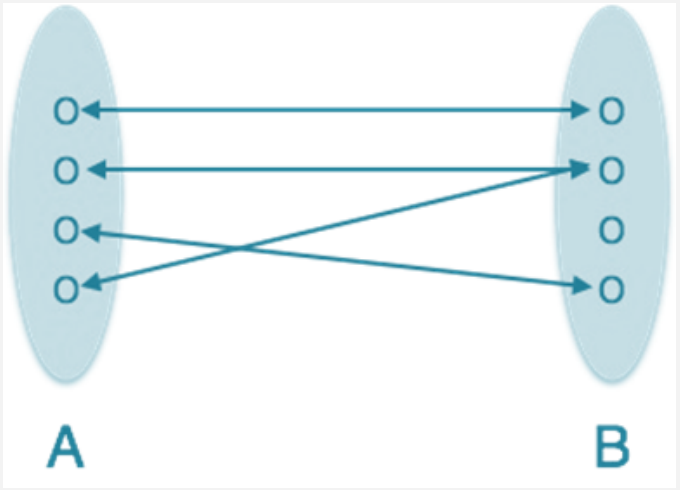
One-to-many :

One entity from entity set A can be associated with more than one entity of entity set B however an entity from entity set B, can be associated with at most one entity.



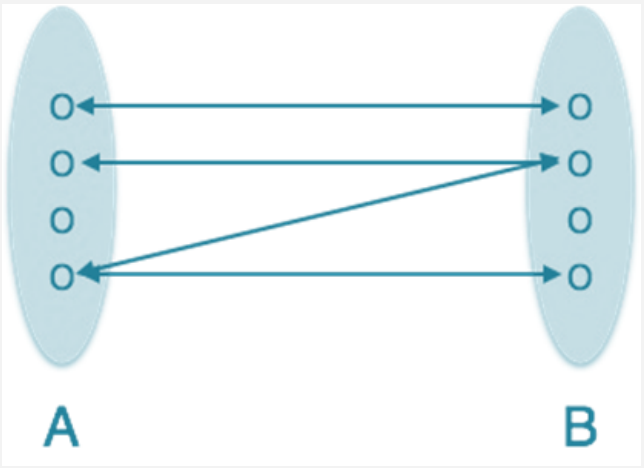
Many-to-one:

More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



Many-to-many :

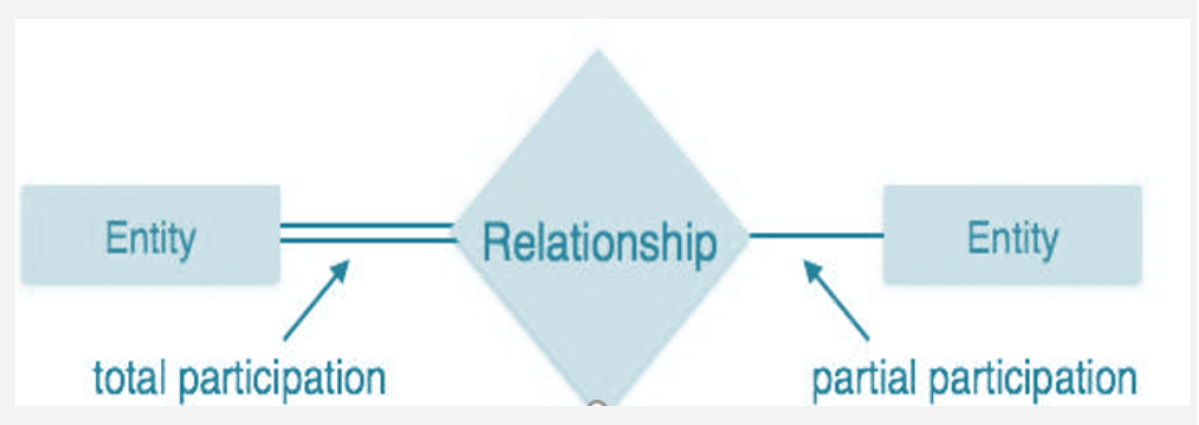
One entity from A can be associated with more than one entity from B and vice versa.



Total and Partial Partition

Total Participation is when each entity in the entity set occurs in at least one relationship in that relationship set.
Entity fully participates in the relationship indicated by double lines drawn from entity to relationship.

Partial Participation is all entities do not involve in the relationship
Entity partially participates in the relationship indicated by single lines drawn from entity to relationship.



Creating an ER Diagram

Entities: An entity is an object or concept about which you want to store information.

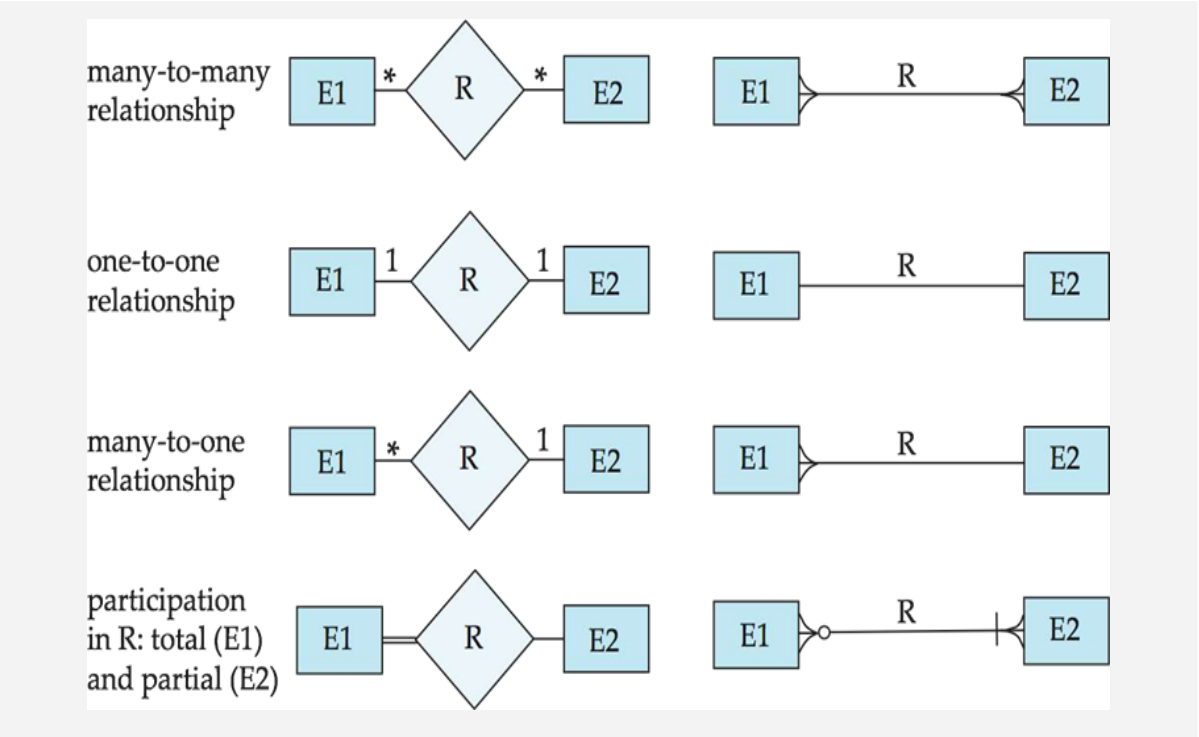
A **weak entity** is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Relationships: Relationships are represented by diamond shape, and show how two entities share information in the database.

Attributes: Attributes are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.
For example, an employee's social security number might be the employee's key attribute

A **multivalued attribute** can have more than one value.
For example, an employee entity can have multiple skill values.

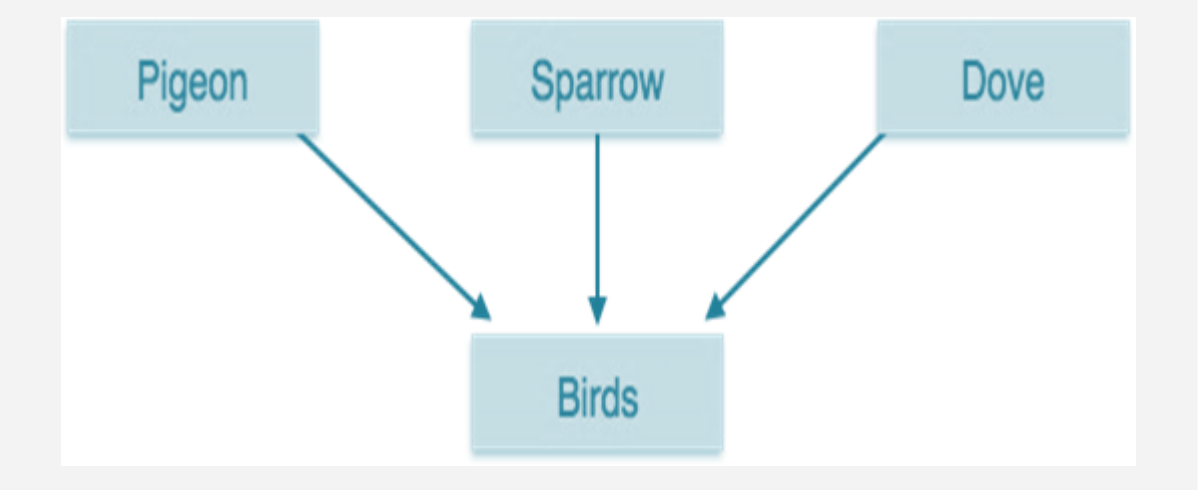
Relationships:



Generalization

The process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization.

In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics.
For example, pigeons, house sparrows, crows and doves can all be generalized as Birds.

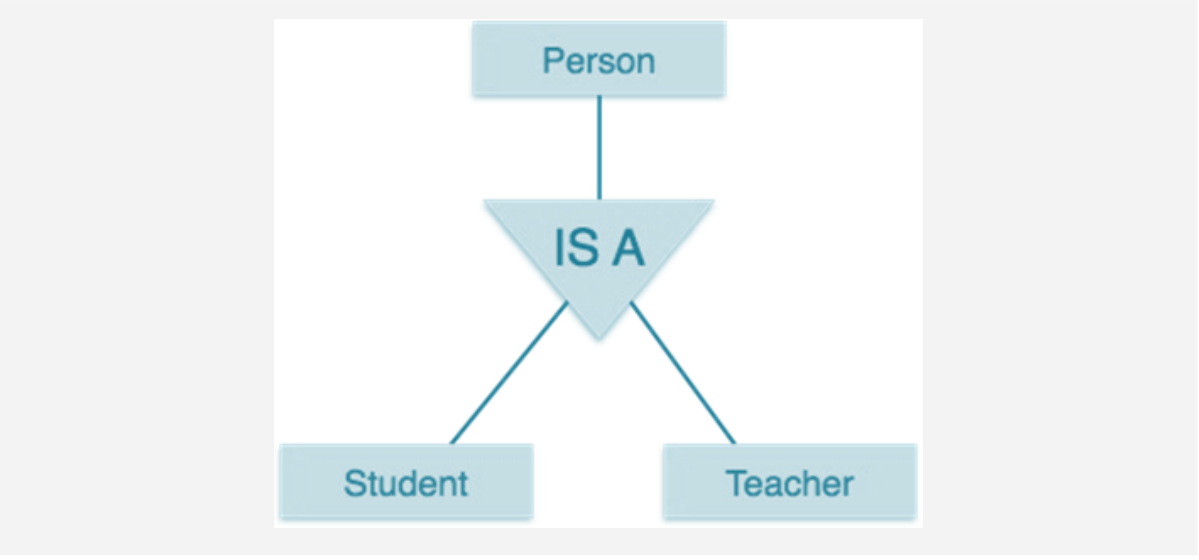


Specialization

Specialization is the opposite of generalization. In specialization, a group of entities is divided into subgroups based on their characteristics.

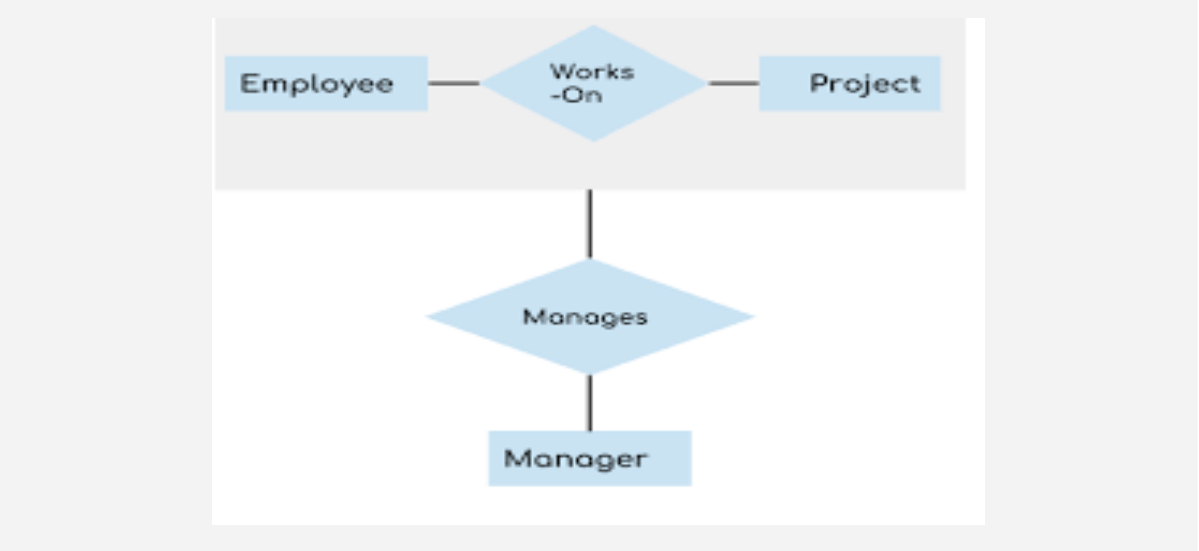
Take a group 'Person' for example. A person has a name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as an employee, employer, customer, or vendor, based on what role they play in the company.

Similarly, in a school database, persons can be specialized as teacher, students, or a staff, based on what role they play in school as entities.

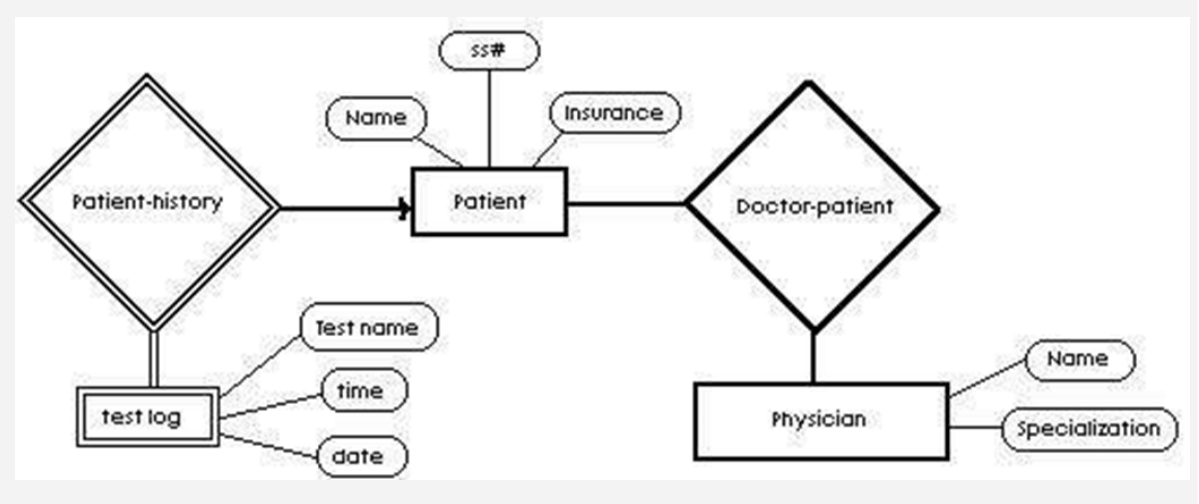


Aggregation

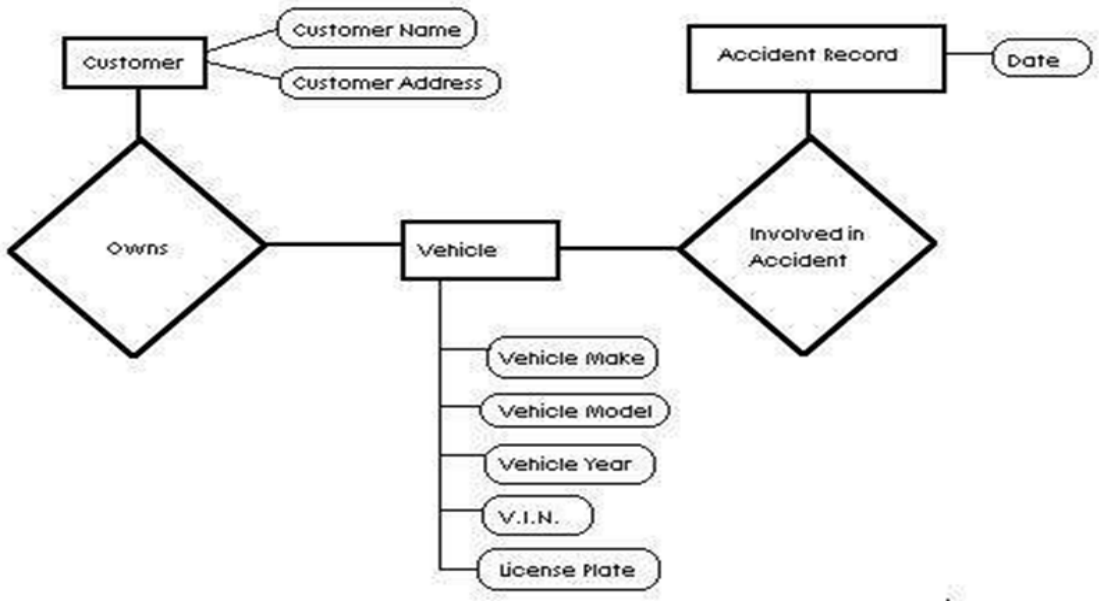
Aggregation refers to the process by which entities are combined to form a single meaningful entity.
Aggregation is an abstraction for building composite objects from their component objects.
Aggregation is used to represent a relationship between a whole object and its component parts.
If we need to express a relationship among relationships, then we should use aggregation



Problems:
Construct an ER diagram for a hospital with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examinations conducted.



Construct an ER diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.



Creating an ER Diagram

Entities: An entity is an object or concept about which you want to store information.

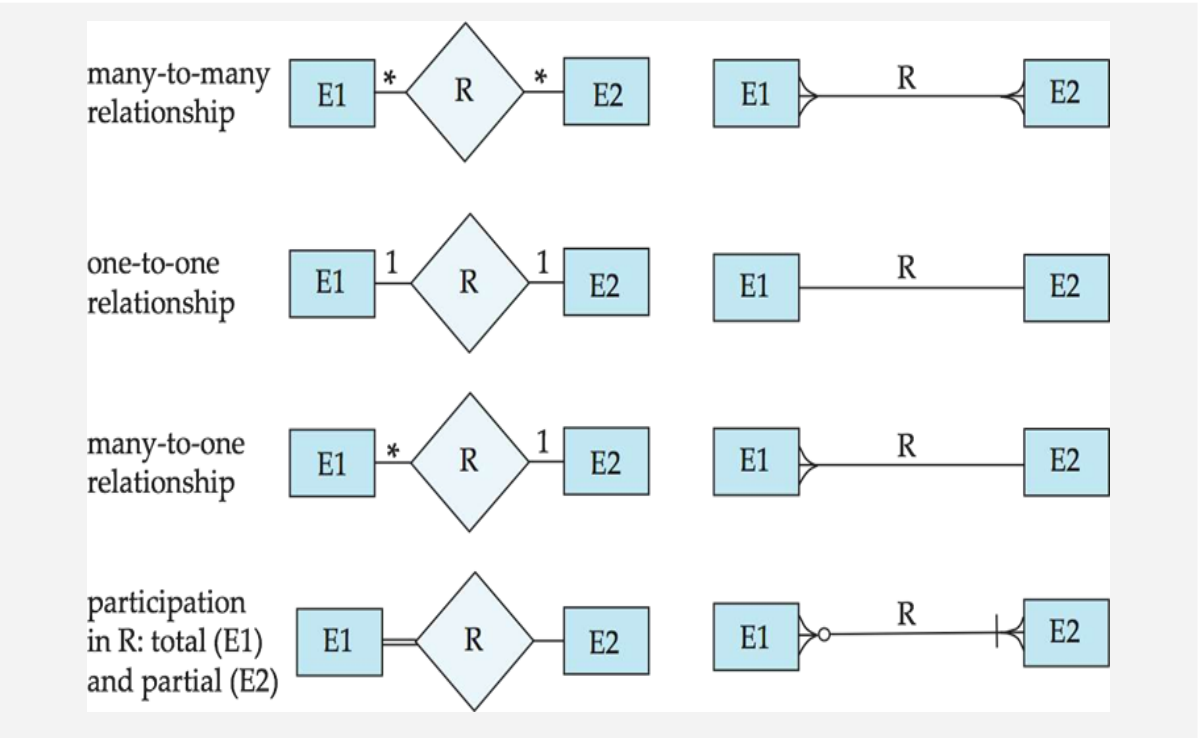
A **weak entity** is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Relationships: Relationships are represented by diamond shape, and show how two entities share information in the database.

Attributes: Attributes are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.
For example, an employee's social security number might be the employee's key attribute

A **multivalued attribute** can have more than one value.
For example, an employee entity can have multiple skill values.

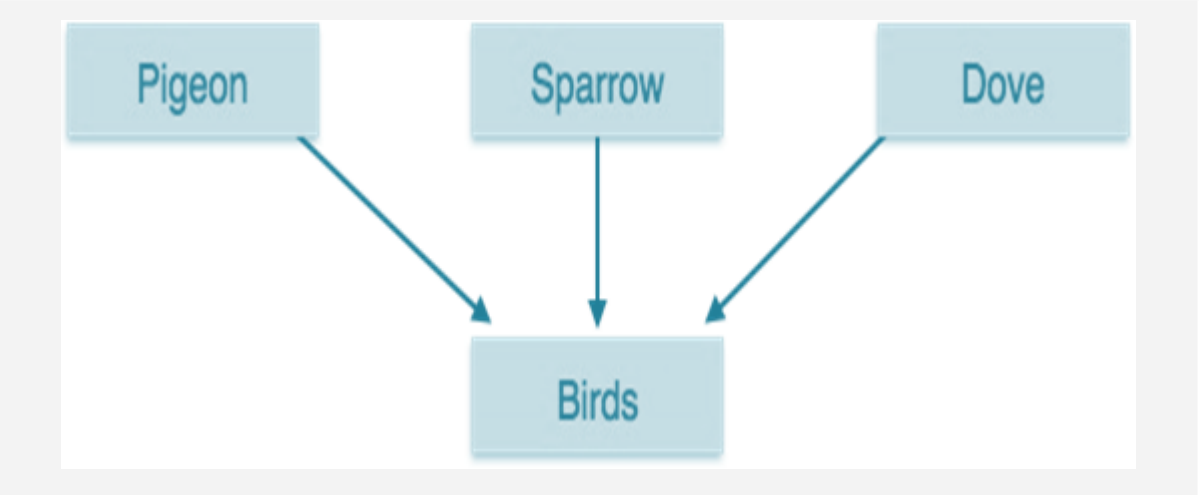
Relationships:



Generalization

The process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization.

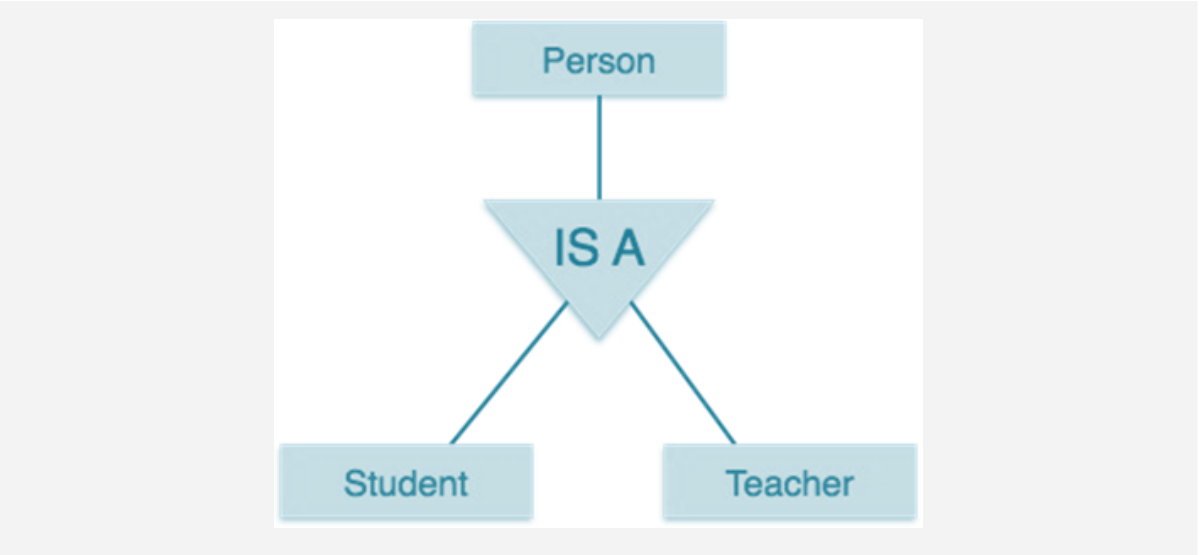
In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics.
For example, pigeons, house sparrows, crows and doves can all be generalized as Birds.



Specialization

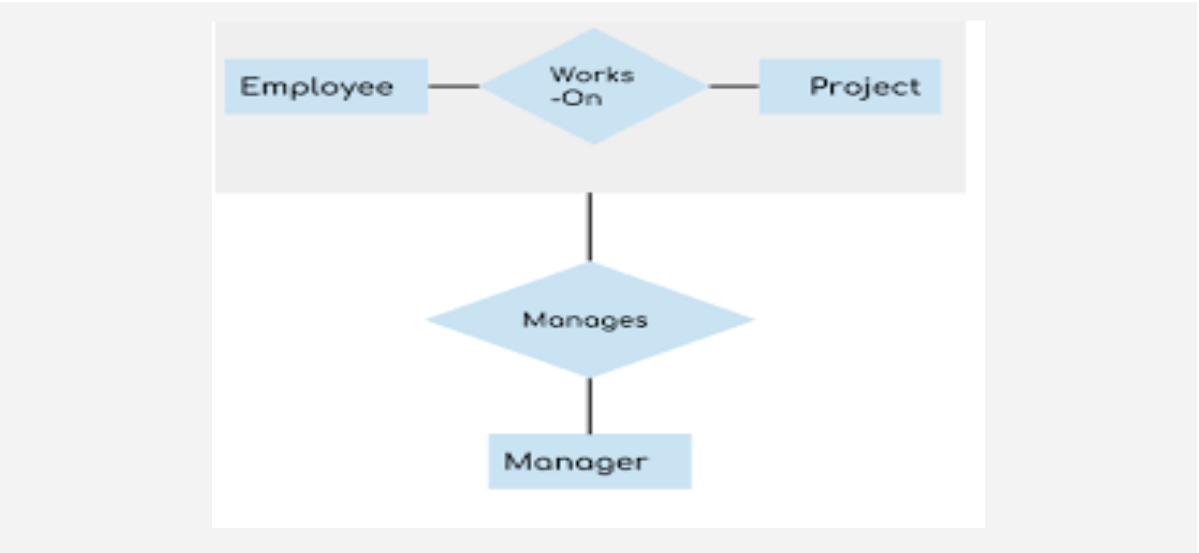
Specialization is the opposite of generalization. In specialization, a group of entities is divided into subgroups based on their characteristics.

Take a group 'Person' for example. A person has a name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as an employee, employer, customer, or vendor, based on what role they play in the company.
Similarly, in a school database, persons can be specialized as teacher, students, or a staff, based on what role they play in school as entities.



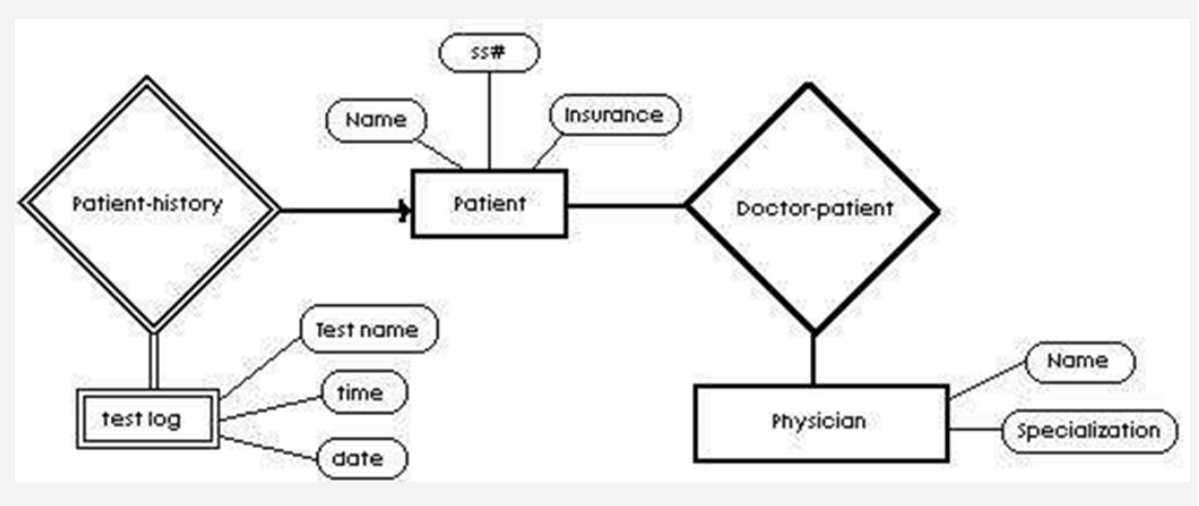
Aggregation

Aggregation refers to the process by which entities are combined to form a single meaningful entity.
Aggregation is an abstraction for building composite objects from their component objects.
Aggregation is used to represent a relationship between a whole object and its component parts.
If we need to express a relationship among relationships, then we should use aggregation

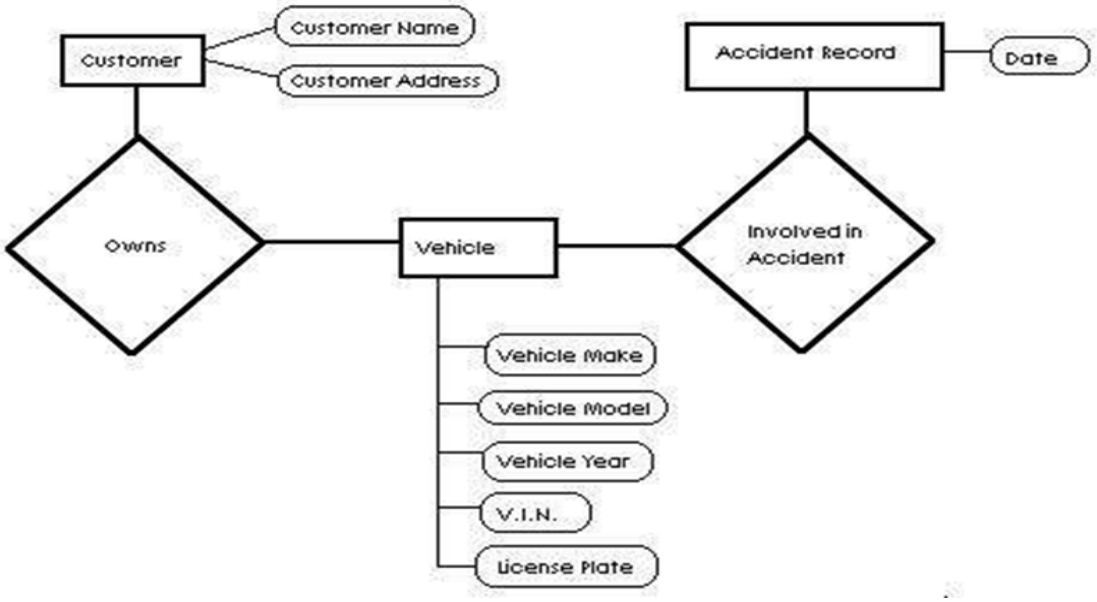


Problems:

Construct an ER diagram for a hospital with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examinations conducted.



Construct an ER diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.


Customer
Customer Name
Customer Address
Accident Record
Date
Owns
Vehicle
Involved in Accident
Vehicle Make
Vehicle Model
Vehicle Year
V.I.N.
License Plate

Interview Question

Q 1. What is cardinality? (Capgemini)
One to One, One to many, and many to many are different types of cardinalities. In a database, high cardinality means more unique values are stored in a column and vice versa.

Q 2. What Does ERD Stand for, and What is it? (TCS)
ERD stands for Entity Relationship Diagram and is a logical entity representation, defining the relationships between the entities. Entities reside in boxes, and arrows symbolize relationships.

Q 3. What is an entity? (Wipro)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “Entity”.

Q 4. Explain a real life example of generalization. (Amazon)
For example, Saving and Current account types entities can be generalised and an entity with name Account can be created, which covers both.

Q 5. How does generalization differ from specialization? (TCS,Infosys)
Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity.
Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entities.

Q 6.How is the degree assigned to a relationship? (Adobe)
Depending upon the number of entities involved, a degree is assigned to relationships.

Q 7. What is a composite entity? (Veritas)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “composite Entity”.

Q 8. Explain derived attributes.

Please refer to the notes for definitions of the aforementioned operating systems under the heading “derived attribute”.

RDBMS

RDBMS stands for Relational Database Management System.
RDBMS data is structured in database tables, fields and records.
Each RDBMS table consists of database table rows.Each database table row consists of one or more database table fields.
RDBMS stores the data into a collection of tables, which might be related by common fields (database table columns).
RDBMS also provides relational operators to manipulate the data stored into the database tables.

Degree Of Relationship

The degree of relationship refers to the number of participating entities in a relationship.
Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a database system are binary.
Relationships between more than two entity sets are rare. Most relationships are binary.

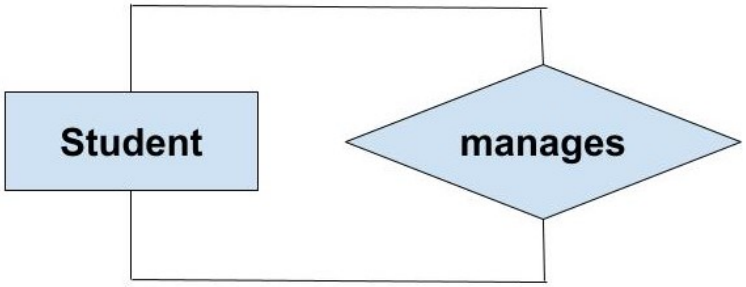
Types of degree

- UNARY
- BINARY
- TERNARY

UNARY

A relationship is unary when both the participating entity types are the same. Where such a relationship exists, we say the degree of the relationship is 1.

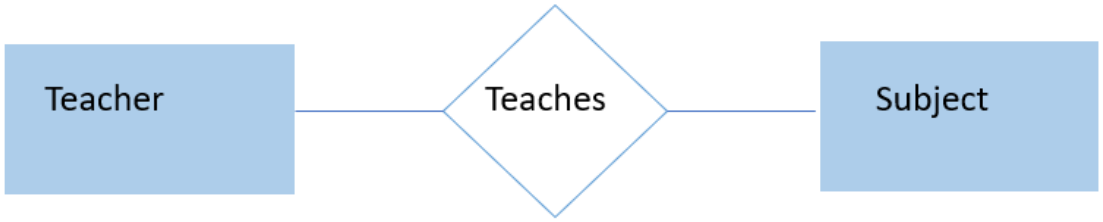
Example:
Here only one entity type in a relationship, and the minimum degree of a relationship is one.



Unary Relationship
(degree 1)

BINARY

If there are two entities involved in a relationship then it is referred to as a binary relationship.
When this type of a relationship is present, we say that the degree is 2. These can be easily converted into relational tables.

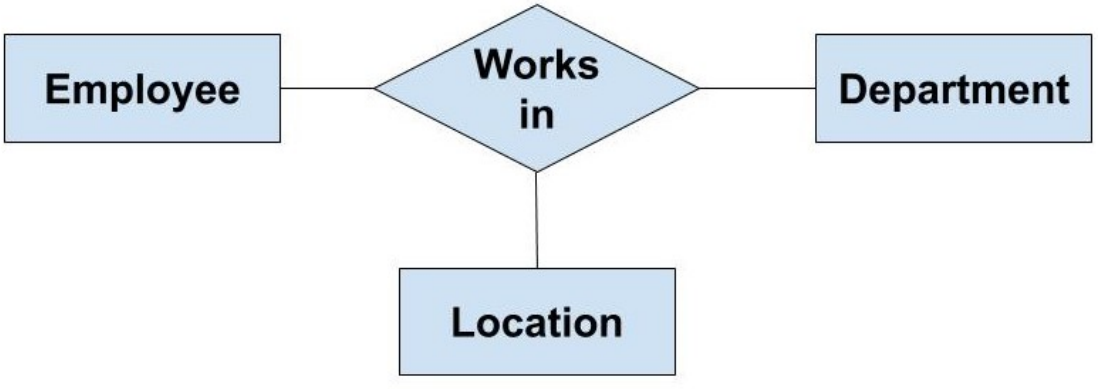


Binary Relation

Example:
For example the relationship “a Teacher teaches one or more Subjects” represents a binary relationship.

Ternary Relationship

If there are three entities involved then it is called a ternary relationship and so on.
When this type of a relationship is present we say that the degree is 3.
As the number of entities increases in the relationship, it becomes complex to convert them into relational tables.



Ternary Relationship (degree 3)

There are three entity types: ‘Employee’, ‘Department’ and ‘Location’. The relationship between these entities are defined as an employee works in a department, an employee works at a particular location. So, we can see we have three entities participating in a relationship so it is a ternary relationship. The degree of this relation is 3.

DATABASE SCHEMA

The term "database schema" can refer to a visual representation of a database, a set of rules that govern a database, or to the entire set of objects belonging to a particular user.
It consists of a list of attributes and instructions that informs the database engine how the data is organized and how the elements are related to each other.

Database Schema Type

There are three main database schema types that define different parts of the schema: logical,View and physical.

Logical Schema

A logical database schema represents how the data is organized in terms of tables. It also explains how attributes from tables are linked together. Different schemas use a different syntax to define the logical architecture and constraints.

Physical Schema

The physical database schema represents how data is stored on disk storage. In other words, it is the actual code that will be used to create the structure of your database. In MongoDB with mongoose, for instance, this will take the form of a mongoose model. In MySQL, you will use SQL to construct a database with tables.

View Schema

The view level design of a database is known as view schema. This schema generally describes the end-user interaction with the database systems.

Integrity Constraint

Keys

Super key
An attribute or a combination of attributes that is used to identify the records uniquely is known as Super Key. A table can have many Super Keys.

Example:

EmpSSN	**EmpNum**	**Empname**
1001	AB05	Shown
2001	AB06	Roslyn
3001	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.
The combination of “SSN” and “Name” is a super key of the following entity set customer.
Because:
The value of attributes “SSN” and “Name”, can uniquely identify that particular customer in the customer entity set, which is the pool of all customers.

Candidate key
The candidate key is a set of one or more attributes whose set of values can uniquely identify an entity instance in the entity set.
Any attribute in the candidate key cannot be omitted without destroying the unique property of the Candidate key.
It is a minimal Super Key.

Properties of Candidate key:

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Example:– ID

– Name, Address

ID	**Name**	**Address**
101	Robot	Boston
201	Mark	Newyork

For above table we have only two Candidate Keys (i.e. Irreducible Super Key) used to identify the records from the table uniquely.
ID Key can identify the record uniquely and a similar combination of Name and Address can identify the record uniquely, but neither Name nor Address can be used to identify the records uniquely as it might be possible that we have two employees with similar names or two employees from the same house.

Composite Key

If we use multiple attributes to create a Primary Key then that Primary Key is called Composite Key (also called a Compound Key or Concatenated Key).

Example of Composite Key, if we have used “Name, Address” as a Primary Key then it will be our Composite Key. ›

Alternate Key

Alternate Key can be any of the Candidate Keys except for the Primary Key.

Example of Alternate Key is “Name, Address” as it is the only other Candidate Key which is not a Primary Key.

Primary Key

A PRIMARY KEY constraint is a rule that the values in one column or a combination of columns must uniquely identify each row in a table.
No primary-key value can appear in more than one row in the table.
No column that is part of the primary key can contain a null.
A table can have only one primary key.

Example:

```
CREATE TABLE Persons (
        ID int NOT NULL,
        FirstName varchar(255),
        Age int,
   PRIMARY KEY (ID)
);
```

FOREIGN KEY (REFERENTIAL INTEGRITY) Constraints

FOREIGN KEY constraints are also called "referential integrity" constraints.
Foreign Key constraints designate a column or combination of columns as a foreign key.
A foreign key links back to the primary key (or a unique key) in another table, and this link is the basis of the relationship between tables.

DEPARTMENTS - Parent

DEPARTMENT_ID	**DEPT_NAME**	**MANAGER_ID**	**LOCATION_ID**
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	**FIRST_NAME**	**LAST_NAME**	**DEPARTMENT_ID**
100	Steven	King	90

101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

In the tables shown, the primary-key of the DEPARTMENTS table, department_id, also appears in the EMPLOYEES table as a foreign-key column.

Integrity constraint

Entity Integrity Constraint
The entity integrity constraint states that primary keys can't be null.
There must be a proper value in the primary key field.
This is because the primary key value is used to identify individual rows in a table. If there were null values for primary keys, it would mean that we could not identify those rows.

On the other hand, there can be null values other than primary key fields. Null value means that one doesn't know the value for that field. Null value is different from zero value or space.

Example:
In the Car Rental database in the Car table each car must have a proper and unique Reg_No. There might be a car whose rate is unknown - maybe the car is broken or it is brand new - i.e. the Rate field has a null value. See the picture below. The entity integrity constraints ensure that a specific row in a table can be identified.

Referential Integrity Constraint
The referential integrity constraint is specified between two tables and it is used to maintain the consistency among rows between the two tables.

The rules are:
1. You can't delete a record from a primary table if matching records exist in a related table. 2. You can't change a primary key value in the primary table if that record has related records.
3. You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.
4. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.

Examples
Rule 1
You can't delete any of the rows in the CarType table that are visible in the picture since all the car types are in use in the Car table.
Rule 2
You can't change any of the model_ids in the CarType table since all the car types are in use in the Car table
Rule 3
The values that you can enter in the model_id field in the Car table must be in the model_id field in the CarType table.
Rule 4
The model_id field in the Car table can have a null value which means that the car type of that car in not known

Interview Question

Q 1. Give a quick idea of the term RDBMS? (Adobe)

It helps in storing or managing data across multiple tables. The best part is that you can define relationships among different data entries using tables. Relationships are generally expressed through values, not pointers.

Q 2. How will you define a relational database model? (Accenture)

It defines the relationship among different databases and how they are connected. When multiple databases are connected, it creates flexibility and can be used within a software app as needed.

Q 3. What are the components of RDBMS? (Vmware)

Each relation in an RDBMS is given a “Name” that will be unique among others. There are rows and columns in each relation; columns represent attributes and rows as Tuples.
Name => Attributes => Tuples

Q 4. What is an E-R Model? (Infosys)
It consists of entities and relational objects. Entities can be understood by the collection of attributes in the database.

Q 5. Tell me something about various data abstraction levels? (Goldman Sachs)

In RDBMS, data can be abstracted at three different levels.
They are given below -.
Physical Level -> Logical Level -> View 1, View 2 & View 3
The physical level is available at the bottom, giving you a detailed idea of the data storage. The Logical level at the next stage finds the logic among data tables and how to group similar data for easy access. At the top, there is a view level that gives information about the complete database and various views of a database.

Q 6. How are RDBMS preferable options over the DBMS? (Amazon)

It minimizes redundancy and integrity can be maintained. It maintains data consistency and allows data sharing to other databases. It follows a set of rules to satisfy storage standards and maintains security.

Q 7. What are the two specific rules that you should follow for each RDBMS to maintain data integrity? (Flipkart)

These are divided into two categories that are necessary to learn by every RDBMS expert.

- To maintain the integrity of an Entity, the Primary key should never contain the NULL values.
- To maintain referential integrity, the foreign key should be used that is defined as the Primary key for another table.

Q 8. Does RDBMS follow an object-oriented approach or not? (Amazon)

Well, I think RDBMS follows the object-oriented approach. The object-oriented model is defined based on the collection of objects. An object instantiates the value stored in instance variables. If objects are sharing the same properties or methods, then they can be grouped to form a class. The same process is followed in RDBMS as well.

Q 9.What are the different features of an RDBMS?(Oracle)

Name- Every relation in a relational database should have a name which is unique among all other relations.
Attributes- Each column in a relation is called an attribute.
Tuples-Each and every row in a relation is called a tuple. A tuple defines a collection of attribute values.

- 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.

Functional Dependency

Functional Dependency

Functional dependency concept is a relationship that exists when one attribute determines differently another attribute.
A functional dependency (FD) on a relation schema R is a constraint X → Y, where X and Y are subsets of attributes of R, which indicates that Y is dependent on X.
The table features are said to depend on each other when the table attribute separately identifies another similar table attribute.

For example:
Suppose we have a student table with attributes: **Stu_Id, Stu_Name, Stu_Age.**
Here the Stu_Id attribute uniquely identifies the **Stu_Name** attribute of the student table because if we know the student id we can tell the student name associated with it.

Functional dependency and can be written as :
Stu_Id->Stu_Name .
We can say Stu_Name is functionally dependent on Stu_Id.

Types of Functional Dependencies

- Trivial functional dependency
- Non-trivial functional dependency

Trivial functional dependency

The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.

It can be written as :
A ->B is trivial functional dependency if B is a subset of A.

The following dependencies are also trivial: A->A & B->B .

For example:
Consider a table with two columns *Student_id* and *Student_Name.*
{Student_Id, Student_Name} -> Student_Id is a trivial functional dependency as Student_Id is a subset of {Student_Id, Student_Name}.
Also, *Student_Id -> Student_Id & Student_Name -> Student_Name* are trivial dependencies too.

Non-trivial functional dependency

If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called a non-trivial Functional dependency.

Example :
An employee table with three attributes: *emp_id, emp_name, emp_address.*

The following functional dependencies are non-trivial:
emp_id -> emp_name (emp_name is not a subset of emp_id)
emp_id -> emp_address (emp_address is not a subset of emp_id)

On the other hand, the following dependencies are trivial:
{emp_id, emp_name} -> emp_name [emp_name is a subset of {emp_id, emp_name}]

Completely non trivial FD:
If a Functional dependency X->Y holds true where X intersection Y is Null then this dependency is said to be completely non trivial functional dependency.

Multivalued dependency
Multivalued dependency occurs when there are more than one independent multivalued attribute in a table.
A multivalued dependency is a full constraint between two sets of attributes in a relation. In contrast to the functional dependency, the multivalued dependency requires that certain tuples be present in a relation.

Transitive dependency

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.

X -> Z is a transitive dependency if the following three functional dependencies hold true: X->Y Y does not ->X Y->Z

A transitive dependency can only occur in a relation of three or more attributes. This dependency helps us normalize the database in 3NF (3rd Normal Form).

Normalization

Normalization:
A process of organizing the data in the database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.
A process of organizing data into tables in such a way that the results of using the database are always unambiguous and as intended. Such normalization is intrinsic to relational database theory. It may have the effect of duplicating data within the database and often results in the creation of additional tables.

Advantages of Normalization

- Elimination of data redundancy makes the database to be compact reducing the overall amount of space a database consumes.
- Enforcement of referential integrity on data ensuring data to be consistent across all tables.
- Maintenance becomes easier and faster since the data are organized logically in a normalized database in a flexible way.
- Searching and sorting of records is easier and faster because data will appear in a separate, smaller table when a database is normalized allowing us to easily find them.

Difference between Normalization and Denormalization

- Normalization and denormalization are two processes that are completely opposite.
- Normalization is the process of dividing larger tables into smaller ones reducing the redundant data, while denormalization is the process of adding redundant data to optimize performance.
- Normalization is carried out to prevent database anomalies.

- Denormalization is usually carried out to improve the read performance of the database, but due to the additional constraints used for denormalization, writes (i.e. insert, update and delete operations) can become slower. Therefore, a denormalized database can offer worse write performance than a normalized database.

- It is often recommended that you should “normalize until it hurts, denormalize until it works”.
- Normalizing data means eliminating redundant information from a table and organizing the data so that future changes to the table are easier.

Interview Questions

Q 1.What are different types of Normalization Levels or Normalization Forms? (Goldman sachs)
The different types of Normalization Forms are:

- **First Normal Form:** Duplicate columns from the same table need to be eliminated. We have to create separate tables for each group of related data and identify each row with a unique column or set of columns (Primary Key)
- **Second Normal Form**: First it should meet the requirement of first normal form. Removes the subsets of data that apply to multiple rows of a table and place them in separate tables. Relationships must be created between the new tables and their predecessors through the use of foreign keys.
- **Third Normal Form:** First it should meet the requirements of the second normal form. Remove columns that are not depending upon the primary key.
- **Fourth Normal Form:** There should not be any multi-valued dependencies.

Q 2.What is Normalization? (TCS)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “Normalization”.

Q 3.What is Denormalization? (Amazon)
De-normalization is the process of optimizing the read performance of a database by adding redundant data or by grouping data. Denormalization is used in OLAP systems.

Q 4. What is BCNF? (Infosys)
Answer: BCNF is the Boyce Codd Normal form. It is the higher version of 3Nf which does not have any multiple overlapping candidate keys.

Q 5. What is a functional dependency in the DBMS?(Matercard)
This is basically a constraint which is useful in describing the relationship among the different attributes in a relation.

Example: If there is some relation ‘R1’ which has 2 attributes as Y and Z then the functional dependency among these 2 attributes can be shown as Y->Z which states that Z is functionally dependent on Y.

Q 6.What are the goals of normalization? (ADOBE)
It means decomposing (dividing/breaking down) a 'big' un-normalized table (file) into several smaller tables by:

- Eliminating insertion, update and delete anomalies.
- Establishing functional dependencies.
- Removing transitive dependencies.
- Reducing non-key data redundancy

Q 7.How do you convert normalization to an ER diagram? (Amazon)
Normalization utilizes association among attributes within an entity table to accomplish its objective. Since an ERD also utilizes association among attributes as a basis to identify entity type structure, it is possible to apply normalization principles during the conceptual data modeling phase.

Q 8.When is functional dependency said to be fully functional dependent? (L&T)
To fulfill the criteria of fully functional dependency, the relation must meet the requirement of functional dependency.
A functional dependency ‘A’ and ‘B’ are said to be fully functional dependent when removal of any attribute say ‘X’ from ‘A’ means the dependency does not hold anymore.

Q 9.Explain trivial functional dependency. (Siemens)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “trivial functional dependency”.

Q 10. Explain the transitive property.(Accenture)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “Transitive property”.

Types of Normalization

Types of Normal Form

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF

1st Normal Form (1NF)

The values in each column of a table are atomic (No multi-value attributes allowed).
Each table has a primary key: minimal set of attributes which can uniquely identify a record
There are no repeating groups: two columns do not store similar information in the same table.

Example of a table not in 1NF :

Genres	**Name**	**Movie**
Marvel	Kevin Feige	The Avengers
		Captain America
DCEU	Zack Snyder	Batman Vs SuperMan
		Suicide Squad

This table contains Attribute values which are not single. This is not in Normalised form.
To make it into 1NF we have to decompose the table into atomic elements.

Table in 1NF after eliminating:

Genres	Name	Movie
Marvel	Kevin Feige	The Avengers
Marvel	Kevin Feige	Captain America
DCEU	Zack Snyder	Batman Vs Superman
DCEU	Zack Snyder	Suicide Squad

Second Normal Form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of the table.

Prime attribute :an attribute, which is a part of the prime-key, is known as a prime attribute.
Non-prime attribute : an attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

Example of a table not in 2NF:

Studio	Movie	Budget	city
Marvel	Avengers	100	New York
Marvel	Captain America	120	New York

DCEU	Batman Vs Superman	150	Gotham
DCEU	Suicide Squad	75	Gotham

Here the Primary key is (studio, movie) and the city depends only on the studio and not on the whole key.
So, this is not in 2NF form.

Solution of 2 NF

Old Scheme ->{Studio, Movie, Budget, City}
New Scheme -> {Movie, Studio, Budget}
New Scheme ->{Studio, City}

Movie	**Studio**	**Budget**
The Avengers	Marvel	100
Captain America	Marvel	120
Batman Vs Superman	DCEU	150
Suicide Squad	DCEU	75

Studio	**City**
Marvel	New York
DCEU	Gotham

Now the 2 tables are in 2NF form.

Third normal form 3 NF

This form dictates that all non-key attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes. For a table to be in 3NF, there are two requirements

- The table should be second normal form
- No attribute is transitively dependent on the primary key

Example of a table not in 3nf

Studio	CityTemp	Studio City
Marvel	96	New York
DCEU	99	Gotham
Fox	96	New York
Paramount	95	Hollywood

Here Studio is the primary key and both studio temp and city depends entirely on the Studio.

1. **Primary Key** ->**{Studio}**
2. **{Studio}** -> **{Studio City}**
3. **{StudioCity}** ->**{CityTemp}**
4. **{Studio}** -> **{CityTemp}**
5. **CityTemp transitively depends on Studio hence violates 3NF**

It is called **transitive dependency**.

Solution of 3NF
Old Scheme -> {Studio, StudioCity, CityTemp}
New Scheme-> {Studio, StudioCity}
New Scheme ->{StudioCity, CityTemp}

Studio	Studio City
Marvel	New York
DCEU	Gotham
FOx	New York
Paramount	Hollywood

Studio City	CityTemp
New York	96
Gotham	95
Hollywood	99

Boyce Codd Normal Form (BCNF) – 3.5NF

BCNF does not allow dependencies between attributes that belong to candidate keys. BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.
Third normal form and BCNF are not same if the following conditions are true:

- The table has two or more candidate keys
- At least two of the candidate keys are composed of more than one attribute
- The keys are not disjoint i.e. The composite candidate keys share some attributes.

Example of BCNF

Scheme -> {MovieTitle, MovieID, PersonName, Role, Payment }

Key1 -> {MovieTitle, PersonName}

Key2 ->{MovieID, PersonName}

MovieTitle	MovieID	PersonName	Role	Payment
The Avengers	M101	Robert Downet Jr.	Tony Stark	200m
The Avengers	M101	Chris Evans	Chris Rogers	120m
Batman Vs Superman	D101	Ben Afflek	Bruce Wayne	180m
Batman Vs Superman	D101	Henry Cavill	Clarke Cent	125m
A walk to remember	P101	Mandy Moore	Jamie Sullivan	50m

Dependency between MovieID & MovieTitle Violates BCNF

Solution of BCNF

Place the two candidate primary keys in separate entities

Place each of the remaining data items in one of the resulting entities according to its dependency on the primary key.

New Scheme ->{MovieID, PersonName, Role, Payment}

New Scheme ->{MovieID, MovieTitle}

MovieID	PersonName	Role	Payment
M101	Robert Downey Jr.	Tony Stark	200m
M101	Chris Evans	Chris Rogers	125m
D101	Ben Afflek	Bruce Wayne	175m
D101	Henry Cavill	Clarke Cent	120m
P101	Mandy Moore	Jamie Sullivan	50m

MovieID	MovieTitle
M101	The Avengers
D101	Batman VS Superman
P101	A walk to remember

4 NF

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.
It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce- Codd Normal Form (BCNF).
It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Example of 4NF
Scheme -> {MovieName, ScreeningCity, Genre)

MovieName	**ScreeningCity**	**Genre**
The Avengers	Los Angeles	Sci-Fi
The Avengers	New York	Sci-Fi
Batman vs Superman	Santa Cruz	Drama
Batman vs Superman	Durham	Drama
A Walk to remember	New York	Romance

Many Movies can have the same Genre and Many Cities can have the same movie.
So this table violates 4NF .

Solution of 4NF
Move the two multi-valued relations to separate tables Identify a primary key for each of the new entities.
New Scheme -> {MovieName, ScreeningCity}
New Scheme -> {MovieName, Genre}

MovieName	**ScreeningCity**
Batman vs Superman	Santa Cruz
The Avengers	Los Angeles
A Walk to remember	New york
Batman vs Superman	Durham
The Avengers	New york

MovieName	**Genre**
Batman vs Superman	Drama
The Avengers	Sci-Fi
A Walk to remember	Romance

We split the table into two tables with one multivalued value in each.

Interview Questions

Q 1. What is Relational Algebra? (Wipro)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “Relational Algebra”.

Q 2. Types of operations in relational algebra.(Accenture)
The fundamental operations in relational algebra are **select, project, union, set difference, Cartesian product, and rename.**

Q 3. Explain the unary and binary operation in relational algebra. (TCS)
The select, project, and rename operations are called unary operations, because they operate on one relation.
The other three operations operate on pairs of relations and are, therefore, called binary operations.

Q 4.Why relational algebra is important in DBMS? (Mastercard)
Relational algebra is very important for several reasons: 1. it provides a formal foundation for relational model operations. Whereas the algebra defines a set of operations for the relational model, the relational calculus provides a higher-level declarative language for specifying relational queries.

Q 5. What are different set operations? (Amazon)

1. **Union**
2. **Intersection**
3. **Set difference**
4. **Cartesian product**

Q 6. Explain Union and intersection between relations.(Capgemini)
Please refer to the notes for definitions of the aforementioned operating systems under the heading “Union And Intersection”.

Q 7. Explain the different unary operators in relational algebra.(Vmware)

1. Select
2. Project
3. Rename

Q 8. Difference between select and project operation.(Adobe)

The Select operation operates horizontally on the table, on the other hand the Project operator works on a single table vertically.

Q 9. What is a cartesian product in relational algebra?(Veritas)

Please refer to the notes for definitions of the aforementioned operating systems under the heading “Cartesian product”.

Q 10. Explain join operation. (Amdocs)

Please refer to the notes for definitions of the aforementioned operating systems under the heading “Join”.

Relational Algebra

Relational algebra is a **procedural query language** which works on relational models. Procedural query language tells what data to be retrieved and how to be retrieved.

A fundamental property is that every operator in the algebra accepts (one or two) relation instances as arguments and returns a relation instance as the result.

A relational algebra expression is recursively defined to be a relation, a unary algebra operator applied to a single expression, or a binary algebra operator applied to two expressions.

The fundamental operations in the relational algebra are **select, project, union, set difference, Cartesian product, and rename.**
The select, project, and rename operations are called unary operations, because they operate on one relation.
The other three operations operate on pairs of relations and are, therefore, called binary operations.

Set Operation

1. **Union**
2. **Intersection**
3. **Set difference**
4. **Cartesian product**

Union

- Union of two relations R and S (R U S) defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.
- R and S must be union-compatible.
- If R and S have I and J tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (I + J) tuples.
- UNION is symbolized by ∪ symbol.

Example:

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∪ Managers

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38
9297	O'Malley	56

Intersection

- The intersection of two relations R and S(R ∩ S), defines a relation consisting of the set of all tuples that are in both R and S.
- R and S must be union-compatible.
- Represented using basic operations: R ∩ S = R – (R – S)

Example:

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∩ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

Set Difference

- The difference of two relations R and S (R – S), define a relation consisting of the tuples that are in relation R, but not in S.
- R and S must be union-compatible.
- It is denoted by(-).
- Represented using basic operations: R -S

Example:

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

Cartesian product

- The Cartesian product of two relations R and S (R X S) , defines a relation between every tuple of relation R with every tuple of relation S.
- It is denoted by (X).
- Represented using basic operations: R X S

Example:

Student

Sr No	Name	Grade
1	Andrew	D
2	Robin	B

Data

Seat No	Age
18	25
11	21

Student x Data

Sr No	Name	Grade	Seat No	Age
1	Andrew	D	18	25
1	Andrew	D	11	21
2	Robin	B	18	25
2	Robin	B	11	21

Unary Relational operations

1. Select
2. Project
3. Rename

Select Operation

- The select operation is performed to select certain rows or tuples of a table, so it performs its action on the table horizontally.,
- The tuples are selected through this operation using a predicate or condition.
- It works on a single table and takes rows that meet a specified condition, copying them into a new table.
- Denoted by lower Greek letter sigma (σ).

Relation: σ predicate(R)

Example of Selection:

Employees

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Black	Lucy	40	3000
Verdi	Nico	36	4500
Smith	Mark	40	3900

σ Age<30 v Salary>4000 **(Employees)**

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Verdi	Nico	36	4500

In selection operation the comparison operators like <, >, =, <=, >=, <> can be used in the predicate

Project Operation

- The Select operation operates horizontally on the table. Conversely, the Project operator works on a single table vertically.
- It is a unary operation that returns a relation that includes a **subset of the attributes** of the operand.
- Since the relation is a set, any duplicate rows are eliminated.
- Projection is represented by a Greek letter (Π) .

Example of Projection:

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{\text{Surname, FirstName}}(\text{Employees})$

Surname	FirstName
Smith	Mary
Black	Lucy
Verdi	Mary
Smith	Mark

Rename

- This is a unary operator which changes attribute names for a relation without changing any values.
- Renaming removes the limitations associated with set operators.
- Representation: ρ old name(R) → New Name(R)

Example: ρ Father->Parent(Paternity)

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

$\rho_{Father \rightarrow Parent}(Paternity)$

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Join Operation

- The JOIN operation, denoted by ⋈, is used to combine related tuples from two relations into single "longer" tuples.
- The join operator allows the combination of two relations to form a single new relation.
- The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation.

Natural Join

An Equijoin of the two relations R and S over all common attributes x. One occurrence of each common attribute is eliminated from the result.
Hence the degree is the sum of the degrees of the relations R and S less the number of attributes in x

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
sales	Brown

$r1 \bowtie r2$

Employee	Department	Head
Smith	sales	Brown
Black	production	Mori
White	production	Mori

Theta Join

Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.

The predicate F is of the form R.ai θ S.bi where θ may be one of the comparison operators (<, ≤, >, ≥, =, ≠).

Interview Questions

Q 1. What is SQL? (TCS)
SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data composed of entities (variables) and relations between different entities of the data.

Q 2. What are the different commands in SQL? (Myntra)

- Data Definition Language (DDL) – It allows you to perform various operations on the database such as CREATE, ALTER, and DELETE objects.
- Data Manipulation Language(DML) – It allows you to access and manipulate data. It helps you to insert, update, delete and retrieve data from the database.
- Data Control Language(DCL) – It allows you to control access to the database. Example – Grant, Revoke access permissions.
- Transaction control Language(TCL)
  Example - Rollback ,commit

Q 3. What are different types of DBMS? (Amazon)
There are two types of DBMS:

- Relational Database Management System: The data is stored in relations (tables). Example – MySQL.
- Non-Relational Database Management System: There is no concept of relations, tuples and attributes. Example – MongoDB

Q 4. What do you mean by table and field in SQL? (IBM)
A table refers to a collection of data in an organised manner in the form of rows and columns. A field refers to the number of columns in a table.

SQL

Introduction to Sql

What is SQL ?

SQL (Structured Query Language) is a computer language aimed to store, manipulate, and retrieve data stored in relational databases. It is a language to communicate with the database.

SQL is the standard language for Relational Database Systems.

Relational database means the data is stored as well as retrieved in the form of relations (tables).

Why is SQL needed?

- SQL is an exceptional programming language that is utilized to interface with databases. It works by understanding and analyzing databases that include data fields in their tables.
- SQL is simple and easy to learn
- Allows users to access information on relational database management systems.
- Allows users to interpret data.
- Allows users to define data in a database and use that data. Allows embedding in other languages using SQL modules, libraries and pre-compilers.
- Allows users to create and drop data and tables.
- Allows users to set permissions for tables and views.

SQL Commands

- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language
- TCL - Transaction Query Language

DDL (Data Definition language)

The data definition language (DDL) is used to specify the relation schemas as well as other information about the relations.

Commands:

Create : Create a new table, a view of the table or other objects in the database.
Alter : Modifies existing database objects such as tables.
Drop : Deletes an entire table,a view of table or other objects in the database.
Rename : Rename object.

DML (Data Manipulation Language)

The DML statements used for managing data within schema objects which deals with data manipulation.

Commands:

Select : retrieve data from a database
Insert : Creates a record
Update : Modifies record,updates existing data within a table
Delete : Deletes a record

DCL (Data Control Language)

The DCL Commands is used to control user access in a database. Using DCL command, it allows or restricts the user from accessing data in database schema.

Commands:

Grant : Gives Privilege to users.
Revoke : Take back privileges granted from users.

TCL (Transaction Query Language)

The TCL commands for specifying the beginning and ending of transactions.These are used to manage the changes made by DML statements.

Commands:

Commit : Permanently save any transaction into the database.
Rollback : Restores the database to the last committed state.

Creating Tables

CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.
To create a new table, you must have the CREATE TABLE privilege and a storage area for it.
The database administrator uses data control language (DCL) statements to grant this privilege to users and assign a storage area.

Syntax:
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);

The column_name parameters specify the names of the columns of the table.
The data_type parameter specifies what type of data the column can hold (e.g. varchar, integer, decimal, date, etc.).
The size parameter specifies the maximum length of the column of the table.

Example:

```
CREATE TABLE Person (
        PID int,
        FName varchar(255),
        );
```

Person

PID	FName

INSERT INTO Statement

The **INSERT INTO** statement is used to insert new records in a table.

SQL INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form does not specify the column names where the data will be inserted, only their values:

Syntax:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

The second form specifies both the column names and the values to be inserted:

Syntax:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Example:

```
INSERT INTO Customers (PID, FName, Country)
VALUES ‘1', 'Robot', 'India');
```

PID	FName	Country
1	Robot	India

ALTER TABLE STATEMENT

The SQL ALTER TABLE statement is used to add, modify, or drop/delete columns in a table. The SQL ALTER TABLE statement is also used to rename a table.

1. ADD TABLE

To add a column in a table, the ALTER TABLE syntax in SQL is:

```
ALTER TABLE table_name
ADD column_name datatype;
```

EXAMPLE:
ALTER TABLE Person
ADD LName varchar(255);

2.DROP TABLE

To drop a column in an existing table, the SQL ALTER TABLE syntax is:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

EXAMPLE:
ALTER TABLE Person
DROP COLUMN Country;

SQL Data Types

Character Data Types

Char

Char data types contains non-binary strings. length of character is fixed while creating a table.string is right-padded with spaces to the specified length when stored.A fixed-length string between 0 and 255 characters in length ,by default is 1.

Syntax: char(size)

Where size is length of string;

Varchar

Varchar contains contains non-binary strings. Columns are variable-length strings.
A variable size between 1 and 255 characters in length. You must define a length when creating a VARCHAR field.

Syntax: VARCHAR(SIZE)

Where size is length of string;

Text

Text values are considered as character strings having a character set.Variable length storage with maximum size of 2GB data

Syntax: TEXT(size)

Numeric Data Types

SMALLINT(size)

A small integer that can be signed or unsigned.Small signed range is from -32768 to 32767. Unsigned range is from 0 to 65535.

INTEGER(size)

An integer can be signed or unsigned. Signed Integer range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255).

BIGINT(size)

A BIG (large) integer that can be signed or unsigned. If signed, the allowable range is from - 9223372036854775808 to 9223372036854775807. If unsigned bigint , the allowable range is from 0 to 18446744073709551615.

FLOAT(L,D)

A FLOAT number that cannot be unsigned. we have to define the display length (L) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

FLOAT(v)

A floating point number.It uses the v value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If v is from 25 to 53, the data type becomes DOUBLE().

DECIMAL(L,D)

An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (L) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

DOUBLE(L,D)

A double precision floating-point number cannot be unsigned. DOUBLE(L,D); where L is length and D is the number of decimals.By default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

Date and Time Data Types:

DATE

A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31.
For example, NOVEMBER 20th, 1999 would be stored as 1999-11-20.

TIME

It stores the time in a HH:MM:SS format.

DATETIME

It is a combination of date and time in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59.
For example, 1:15 in the afternoon on November 20th, 1999 would be stored as 1999-11-20 13:15:00

TIMESTAMP

A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).

YEAR

It stores the year in 2 digit or 4 digit format. Range 1901 to 2155 in 4-digit format. Range 70 to 69, representing 1970 to 2069.

BINARY

Fixed length with maximum length of 8,000 bytes

BIT

A BIT data type is used to store bit values from 1 to 64. So, a BIT field can be used for booleans, providing 1 for TRUE and 0 for FALSE.

Fundamental Queries

Select Statement

Select command is used to view the records from the table. To view all the columns and all the rows '*'can be specified with the select statement. The name of the table is required while specifying the select.

Syntax :- Select * from <tablename>;

The introductory SELECT statement has three clauses: SELECT FROM WHERE

The SELECT clause specifies the table columns that are retrieved. (Projection in relational algebra)
The FROM clause specifies the tables accessed (cartesian product in relational algebra)
The WHERE clause filters the rows from the table. The WHERE clause is optional; if missing, all table rows are used. (selection in relational algebra)

For Example:

SELECT column_name,column_name FROM table_name;
SELECT * FROM table_name;

Where Statement

The WHERE clause is used to extract only those records that fulfill a specified criterion. It can also be used with UPDATE and DELETE commands.

Syntax:

```
SELECT column_name
FROM  table_name
WHERE condition ;
```

AND,OR and NOT Statement

AND is an operator that joins two conditions. Both conditions must be true for the row to be included in the result set.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_1 = value_1
 AND column_2 = value_2;
```

The **OR** operator displays a record if either the first condition OR the second condition is true.

Syntax:

```
SELECT column_name
FROM table_name
WHERE column_name = value_1
  OR column_name = value_2;
```

The **NOT** operator can put before any conditional statement to select rows for which that statement is false. NOT is commonly used with LIKE.

Syntax:

```
SELECT column_name
FROM table_name
WHERE NOT condition;
```

Handling NULL

The NULL keyword can also be used in predicates to check for null values.NULL value field is a valueless field.
If the field in the table is not selected, it is possible to add a new record or update the record without adding value to this field. After that, the field will be saved at NULL value.

Null value is identified with **Is NULL** OR **NOT NULL** Keyword.
IS NULL and IS NOT NULL are operators used with the WHERE clause to test for empty values.

Syntax :

```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NULL(NOT NULL);
```

IN, BETWEEN and LIKE

BETWEEN returns values that come within a given range. BETWEEN operator is inclusive: begin and end values are included.

Syntax of BETWEEN:

```
SELECT column_name
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

The **IN** keyword allows you to easily test if the expression matches any value in the list. It is used to help reduce the need for multiple OR cases.

Syntax for IN:

```
SELECT column_name
FROM table_name
WHERE column_name IN (value1, value2, ...value n);
```

The LIKE operator is used to search for a specified pattern in a column.
There are two operators that are used with the LIKE operator.

1. %: Percentage is used for representation of single, multiple or no occurrence.
2. _: The underscore is used for representation of a single character.

Syntax for LIKE:
SELECT column_name
FROM table_name
WHERE column_name ***LIKE*** pattern;

ORDER BY Statement

ORDER BY is a statement that sorts the result set by a particular column either ascending or descending.
It sorts the result in ascending order by default. For descending order, use DESC KEYWORD.

Syntax:
SELECT column_name
FROM table_name
ORDER BY column_name ASC | DESC;

LIMIT and OFFSET

LIMIT is a keyword that lets you specify the maximum number of rows the result set will have.

Syntax for LIMIT:

SELECT column_name(s)
FROM table_name
LIMIT number;

The OFFSET keyword is used to identify the starting point to retrieve lines from the result set. Basically, it releases the first set of records.

It is only used with ORDER BY.
Value must be greater than or equal to zero. It cannot be negative, else return error.

Syntax For OFFSET:

```
SELECT column_name
FROM table_name
WHERE condition
ORDER BY column_name
OFFSET rows_name ROWS;
```

DISTINCT

DISTINCT specifies that the statement is going to be a query that returns unique values in the specified column(s).

Syntax:

```
SELECT DISTINCT column_name
FROM table_name;
```

CASE EXPRESSION

The CASE statement returns value when the original condition is met Therefore, if the situation is true, we will stop reading and return the results. If there are no valid terms, it returns the value in the ELSE clause.

If it is not part of ELSE and there are no true terms, it returns NULL.

Syntax:

```
SELECT column_name,
  CASE
```

```
   WHEN condition THEN 'Result_1'
   WHEN condition THEN 'Result_2'
   ELSE 'Result_3'
  END
FROM table_name;
```

GROUP BY

The **GROUP BY** clause is used to group rows that have the same column values.

Syntax:

```
SELECT column_name
FROM table_name
WHERE condition
GROUP BY column_name;
```

HAVING CLAUSE

The **HAVING clause** is similar to where condition, but the WHERE keyword could not be used with aggregate functions.To group data with aggregate function, HAVING is used.

Syntax:

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > value;
```

Subquery

A subquery is a SQL query nested inside a larger query.

Where does the Subquery occur?

A subquery may occur in:

- A SELECT clause
- A WHERE clause

The subquery can be nested inside a SELECT,INSERT, UPDATE, or DELETE statement or inside another subquery.

A subquery is usually written inside the WHERE clause of another SQL SELECT statement.

You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.

Syntax:

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
  (SELECT column_name [, column_name ]
  FROM table1 [, table2 ]
  [WHERE])
```

Example:

Display the price and average price of all books:

SELECT Price, **(SELECT Avg(Price) FROM titles)** AS AveragePrice FROM titles

Types of Subquery

- Non Correlated Subqueries
- Correlated Subqueries

Non Correlated Subqueries

A noncorrelated subquery executes independently of the outer query. The subquery executes first, and then passes its results to the outer query.

For example:

SELECT name, street, city, state FROM addresses WHERE state IN (SELECT state FROM states);

A query's WHERE and HAVING clauses can specify non correlated subqueries if the subquery resolves to a single row, as shown below:

In WHERE clause
SELECT COUNT(*) FROM SubQ1 WHERE SubQ1.a = (SELECT y from SubQ2);

In HAVING clause
SELECT COUNT(*) FROM SubQ1 GROUP BY SubQ1.a HAVING SubQ1.a = (SubQ1.a & (SELECT y from SubQ2)

Correlated Subqueries

A correlated subquery typically obtains values from its outer query before it executes. When the subquery returns, it passes its results to the outer query.
A correlated subquery refers to one or more columns from outside of the subquery.

In the following example, the subquery needs values from the addresses.state column in the outer query:

```
SELECT name, street, city, state FROM addresses
    WHERE EXISTS (SELECT * FROM states WHERE states.state = addresses.state);
```

Aggregate Function

COUNT()

The COUNT function returns the number of values present in a particular column.

Syntax:
SELECT ***COUNT***(column_name)
FROM table_name
WHERE condition;

SUM()

The SUM function returns the sum of values in the specified column.

Syntax:
SELECT ***SUM***(column_name)
FROM table_name
WHERE condition;

AVG()

The AVG function returns the average of values in a specified column.

Syntax:
SELECT ***AVG***(column_name)
FROM table_name
WHERE condition;

MIN()

The MIN function returns the smallest value of a specified column.

Syntax:
SELECT ***MIN***(column_name)
FROM table_name
WHERE condition;

MAX()

The MAX function returns the maximum value of a specified column.

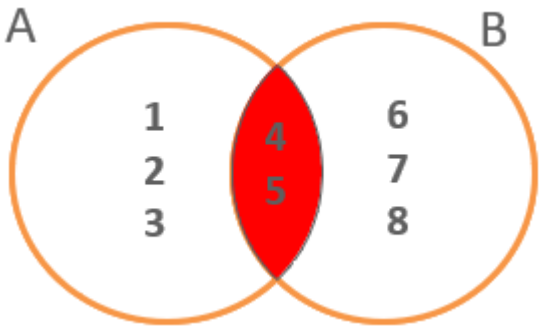
Syntax:
SELECT ***MAX***(column_name)
FROM table_name
WHERE condition;

Set Operators

UNION

The UNION operator returns all rows from both tables, after eliminating duplicates.

Syntax:
SELECT column1 FROM table 1
UNION
SELECT column 2 FROM table 2;



The result of listing all elements in A and B eliminating duplicates is {1, 2, 3, 4, 5, 6, 7, 8}.

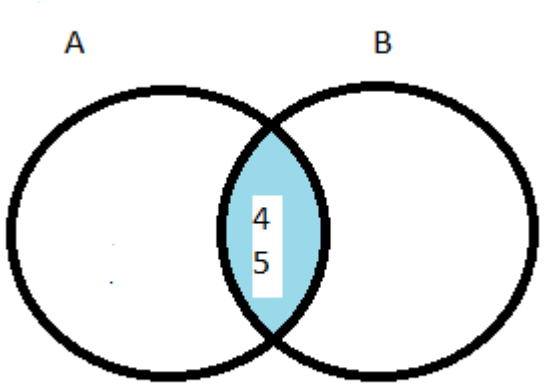
If you joined A and B you would get only {4, 5}. You would have to perform a full outer join to get the same list as above.

INTERSECTION

The INTERSECT operator returns all rows common to both tables.

Syntax:

```
SELECT a_id FROM a INTERSECT
SELECT b_id FROM b;
```



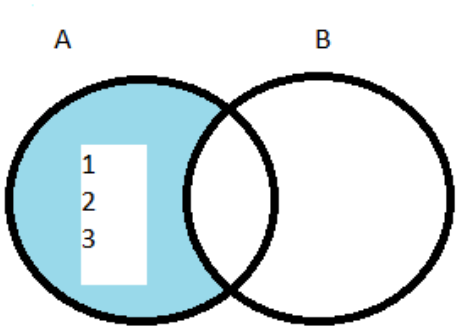
The result of listing all elements found in both A and B is {4, 5}.

EXCEPT

EXCEPT returns distinct rows from the left input query that aren't output by the right input query.

Syntax:

```
SELECT a_id FROM a EXCEPT
SELECT b_id FROM b;
```



The result of listing all elements found in A but not B is {1, 2, 3}.
The result of B MINUS A would give {6, 7, 8}.

SQL Joins

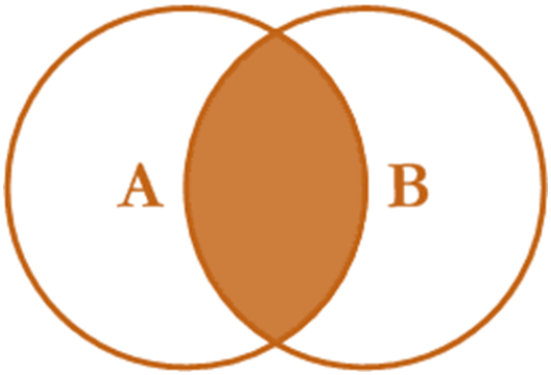
An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

Types of SQL join operations

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- OUTER JOIN

INNER JOIN

Inner join only takes those rows from the Cartesian Product Table where the join elements match fully.
Inner join condition will create result by combining all rows from both tables where the value of common filled will be the same.



Example:

Product

PID	PName
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	ProductID	Price
101	1	1000
102	2	800
103	5	400
104	2	600

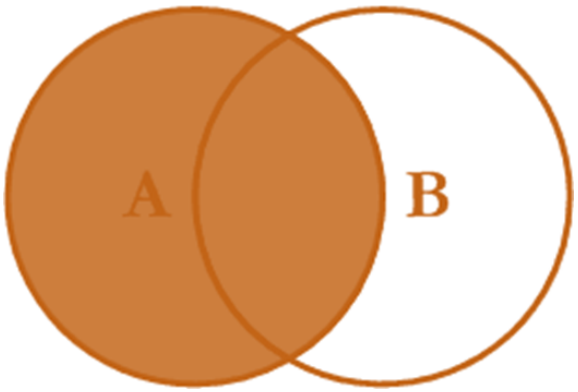
Query: Select * from Product p ***Inner join*** Sale s on p.PID = s.ProductID;

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800

2	Panjabi	104	2	600

LEFT JOIN

Left join takes those rows which are in the inner join output. And it also looks for the rows in the left table which are not in the inner join output. The rows are added to OUTPUT with null in right columns.



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

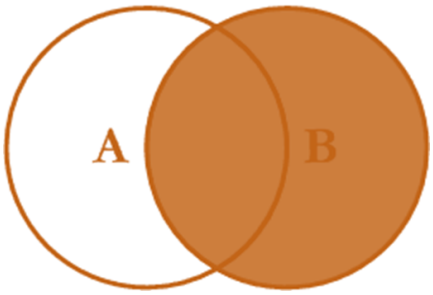
Query: Select * from Product p ***Left join*** Sale s on p.PID = s.ProductID;

So the output of the join table is shown below.
In the left table Product |PID = 3 | Pname = Lungi|
row could not be joined with any row of Sale table. So it is added with null value in right columns

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	104	2	600
3	Lungi	null	null	null

RIGHT JOIN

Right join takes those rows which are in the inner join output. Also looks for the rows in the right table which are not in the inner join output. The rows are added to OUTPUT with null in the left columns.



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

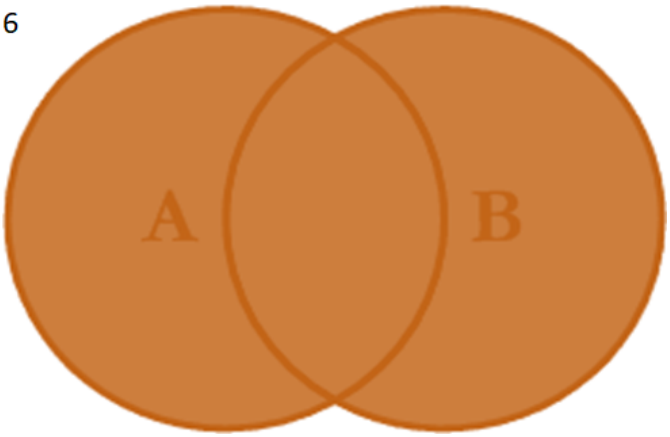
Query: Select * from Product p ***Right join*** Sale s on p.PID = s.ProductID;

So the output of the join table is shown below.
In the right table Sale |SID = 103|ProductID = 5|Price = 400|row could not be joined with any row of Product table. So it is added with null value in the left columns.

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	104	2	600
null	null	103	5	400

OUTER JOIN

Aside from inner join output Outer join looks for the rows in the left table which are not in inner join output. The rows are added to OUTPUT with null in right columns. Similarly the rows from the right table not in the inner join output are added to OUTPUT with null values in left columns.



Example:

Product

PID	PName
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	ProductID	Price
101	1	1000
102	2	800
103	5	400
104	2	600

Query: Select * from Product p ***Outer join*** Sale s on p.PID = s.ProductID

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Punjabii	102	2	800
2	Punjabi	104	2	600
3	Lungi	null	null	null
null	null	103	5	400

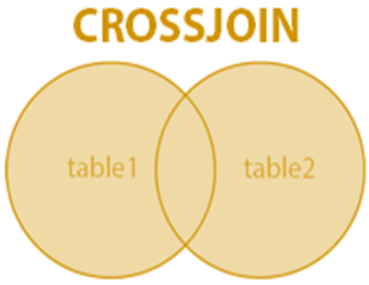
In the 4th row there is no joinable row on the right. So the right values are null. Similarly in the 5th row there is no joinable row in the left. So left values are null.

CROSS JOINS

The CROSS JOIN is used to generate a paired combination of each row of the first table with each row of the second table. This join type is also known as cartesian join.
A cross join produces a cartesian product between the two tables, returning all possible combinations of all rows.

Syntax:

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

Query: Select * from table 1 ***CROSS JOIN*** table 2;

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
1	Shirt	102	2	800
1	Shirt	103	5	400
1	Shirt	104	2	600
2	Panjabi	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	103	5	400
2	Panjabi	104	2	600
3	Lungi	101	1	1000

3	Lungi	102	2	800
3	Lungi	103	5	400
3	Lungi	104	2	600

Views

Views

A view, like a table, is a database object.
They are logical representations of existing tables or of another view.
You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

Creating View

Views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example:
Create a view to print names of all movies in capital letters.
CREATE VIEW Movies_upper(title) AS SELECT UPPER(movie_title) FROM Movies)

UPDATE VIEW

A view can be updated with the CREATE OR REPLACE VIEW statement.

Syntax:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

DROP VIEW

A view is deleted with the DROP VIEW statement.

Syntax:
DROP VIEW view_name;

Keys

Primary Key

A PRIMARY KEY constraint is a rule that the values in one column or a combination of columns must uniquely identify each row in a table.
No primary-key value can appear in more than one row in the table.
No column that is part of the primary key can contain a null.
A table can have only one primary key.

Example:

```
CREATE TABLE Persons (
        ID int NOT NULL,
        FirstName varchar(255),
        Age int,
   PRIMARY KEY (ID)
);
```

FOREIGN KEY (REFERENTIAL INTEGRITY) Constraints

FOREIGN KEY constraints are also called "referential integrity" constraints.
Foreign Key constraints designate a column or combination of columns as a foreign key.
A foreign key links back to the primary key (or a unique key) in another table, and this link is the basis of the relationship between tables.

DEPARTMENTS - Parent

DEPARTMENT_ID	**DEPT_NAME**	**MANAGER_ID**	**LOCATION_ID**
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	**FIRST_NAME**	**LAST_NAME**	**DEPARTMENT_ID**
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

In the tables shown, the primary-key of the DEPARTMENTS table, department_id, also appears in the EMPLOYEES table as a foreign-key column.

Interview Questions

Q 1. How to change a table name in SQL? (Vmware)
This is the command to change a table name in SQL:
ALTER TABLE table_name
RENAME TO new_table_name;

Q 2. What is the difference between DELETE and TRUNCATE statements? (Goldman Sachs)

DELETE	TRUNCATE
Delete command is used to delete a row in a table.	Truncate is used to delete all the rows from a table.
You can rollback data after using the delete statement.	You cannot rollback data.
It is a DML command.	It is a DDL command.
It is slower than a truncated statement.	It is faster.

Q 3. What is an Index? (Morgan Stanely)
An index refers to a performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and hence it will be faster to retrieve data.

Q 4. Explain different types of indexes in SQL. (Microsoft)
There are three types of index namely:
Unique Index:
This index does not allow the field to have duplicate values if the column is uniquely indexed. If a primary key is defined, a unique index can be applied automatically.

Clustered Index:

This index reorders the physical order of the table and searches based on the basis of key values. Each table can only have one clustered index.

Non-Clustered Index:
Non-Clustered Index does not alter the physical order of the table and maintains a logical order of the data. Each table can have many nonclustered indexes.

Q 5. Are NULL values the same as that of zero or a blank space? (VERITAS, TCS)
A NULL value is not at all the same as that of zero or a blank space. NULL value represents a value which is unavailable, unknown, assigned or not applicable whereas a zero is a number and blank space is a character

Q 6. What are the different types of a subquery? (D.E.SHAW)
There are two types of subquery namely, Correlated and Non-Correlated.

Correlated subquery: These are queries which select the data from a table referenced in the outer query. It is not considered as an independent query as it refers to another table and refers to the column in a table.

Non-Correlated subquery: This query is an independent query where the output of the subquery is substituted in the main query.

Q 7. Write a SQL query to find the names of employees that begin with 'A'? (ADOBE)
To display name of the employees that begin with 'A', type in the below command:
SELECT * FROM Table_name WHERE EmpName like 'A%';

Q 8. What is the need for group functions in SQL? (WIPRO)
Group functions work on the set of rows and return one result per group. Some of the commonly used group functions are: AVG, COUNT, MAX, MIN, SUM, VARIANCE.

Q 9. How can you insert NULL values in a column while inserting the data? (INTUIT)
NULL values in SQL can be inserted in the following ways:

- Implicitly by omitting column from column list.
- Explicitly by specifying NULL keyword in the VALUES clause

Q 10. What is the main difference between 'BETWEEN' and 'IN' condition operators? (GOOGLE)

BETWEEN operator is used to display rows based on a range of values in a row whereas the IN condition operator is used to check for values contained in a specific set of values.

Q 11. What is the difference between 'HAVING' CLAUSE and a 'WHERE' CLAUSE? (HSBC)

HAVING clause can be used only with a SELECT statement. It is usually used in a GROUP BY clause and whenever GROUP BY is not used, HAVING behaves like a WHERE clause. Having Clause is only used with the GROUP BY function in a query whereas WHERE Clause is applied to each row before they are a part of the GROUP BY function in a query.

Q 12. How can you select unique records from a table? (WALMART)

You can select unique records from a table by using the DISTINCT keyword.
Select DISTINCT studentID from Student
Using this command, it will print a unique student id from the table Student.

Q 13. Write a Query to display the number of employees working in each region? (MASTERCARD)

SELECT region, COUNT(gender) FROM employee GROUP BY region;

Q 14. How to use LIKE in SQL? (BITWISE)

The LIKE operator checks if an attribute value matches a given string pattern. Here is an example of LIKE operator
SELECT * FROM employees WHERE first_name like 'Steven';
With this command, we will be able to extract all the records where the first name is like "Steven".

Interview Questions

Q 1.If we drop a table, does it also drop related objects like constraints, indexes, columns, default, views and stored procedures? (ZS)

Yes, SQL Server drops all related objects, which exist inside a table like constraints, indexes, columns, defaults etc. But dropping a table will not drop views and stored procedures as they exist outside the table.

Q 2. What is a join? (MICROSOFT)

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

Q 3. What are the types of joining and explain each? (ADOBE)

There are various types of join that can be used to retrieve data and it depends on the relationship between tables.

- Inner Join.

Inner join returns rows when there is at least one match of rows between the tables.

- Right Join.

Right join returns rows that are common between the tables and all rows of the Right-hand side table. Simply, it returns all the rows from the right-hand side table even though there are no matches in the left-hand side table.

- Left Join.

Left join returns rows that are common between the tables and all rows of the Left-hand side table. Simply, it returns all the rows from the Left-hand side table even though there are no matches in the Right-hand side table.

Q 4. What is the difference between cross join and natural join? (SIEMENS)

The cross join produces the cross product or Cartesian product of two tables whereas the natural join is based on all the columns having the same name and data types in both the tables.

Q 5. What is Self-Join? (DELOITTE)

Self-join is set to be a query used to compare to itself. This is used to compare values in a column with other values in the same column in the same table. ALIAS ES can be used for the same table comparison.

Q 6. What is the difference between cross join and natural join? (INFOSYS)

The cross join produces the cross product or Cartesian product of two tables whereas the natural join is based on all the columns having the same name and data types in both the tables.

Interview Questions

Q 1. What is a Primary key? (L&T)

- A Primary key is a column (or collection of columns) or a set of columns that uniquely identifies each row in the table.
- Uniquely identifies a single row in the table
- Null values not allowed

Q 2. What is a Unique key? (Persistent)

- Uniquely identifies a single row in the table.
- Multiple values allowed per table.
- Null values allowed.

Q 3. What is a Foreign key in SQL? (Mastercard)

- Foreign key maintains referential integrity by enforcing a link between the data in two tables.
- The foreign key in the child table references the primary key in the parent table.
- The foreign key constraint prevents actions that would destroy links between the child and parent tables.

Q 4. What is a View? (BAJAJ)

A view is a virtual table that consists of a subset of data contained in a table. Since views are not present, it takes less space to store. Views can have data of one or more tables combined and it depends on the relationship.

Q 5. What are Views used for? (AMAZON)

A view refers to a logical snapshot based on a table or another view. It is used for the following reasons:

- Restricting access to data.
- Making complex queries simple.
- Ensuring data independence.
- Providing different views of the same data.

DDL

DDL (Data Definition language)

The data definition language (DDL) is used to specify the relation schemas as well as other information about the relations.
All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Commands:

Create : Create a new table, a view of the table or other objects in the database.
Alter : Modifies existing database objects such as tables.
Drop : Deletes an entire table,a view of table or other objects in the database.
Rename : Rename object.

CREATE COMMAND

Create is a DDL command used to create a table or a database.

```
Syntax:
CREATE database database-name;
```

Example for Creating Database,
CREATE database Test;

ALTER COMMAND

Alter command is used for alteration of tableDrop Command
There are various uses of alter command, such as:

- to add a column to the existing table .
- to rename any existing column.
- to change the datatype of any column or to modify its size.
- alter is also used to drop a column.

DROP COMMAND

Drop query completely removes a table from the database.
This command will also destroy the table structure.

```
Syntax:
DROP  table table-name;
```

For Example
DROP table Student;

RENAME COMMAND

Rename command is used to rename a table.

```
Syntax:
RENAME table old-table-name to new-table-name;
```

For Example Rename table Student to Student-record;
The above query will rename the Student table to Student-record.

DML

DML (Data Manipulation Language)

The DML statements used for managing data within schema objects which deal with data manipulation.

DML commands are not auto-committed. It means changes are not permanent to the database, they can be rolled back.

Commands:

Select : retrieve data from a database
Insert : Creates a record
Update : Modifies record,updates existing data within a table
Delete : Deletes a record

SELECT COMMAND

Select command is used to view the records from the table. To view all the columns and all the rows '*'can be specified with the select statement. The name of the table is required while specifying the select.

Syntax :- Select * from <tablename>;

INSERT COMMAND

Insert command is used to insert data into a table.

Syntax:
INSERT into table-name values(data1,data2);

For Example:
Consider a table Student with following fields.

S_id	S_Name	age

INSERT into Student values(101,'Adam',15);

The above command will insert a record into Student table

S_id	S_Name	age
101	Adam	15

UPDATE COMMAND

Update command is used to update a row of a table.

Syntax:

UPDATE table-name set column-name = value where condition;

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	chris	14

UPDATE Student set s_name='Abhi',age=17 where s_id=103;

The above command will update two columns of a record.

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

DELETE COMMAND

Delete command is used to delete data from a table. Delete command can also be used with conditions to delete a particular row.

Syntax:

DELETE from table-name ;

Example to Delete all Records from a Table:

DELETE from Student;

The above command will delete all the records from Student table.

Example to Delete a particular Record from a Table

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

DELETE from Student where s_id=103;

The above command will delete the record where s_id is 103 from the Student table.

S_id	S_Name	age
101	Adam	15
102	Alex	18

DCL

DCL (Data Control Language)

The DCL Commands is used to control user access in a database. Using DCL command, it allows or restricts the user from accessing data in database schema.

Commands:

Grant : Gives Privilege to users.
Revoke : Take back privileges granted from users.

GRANT COMMAND

It used to provide any user access privileges or other privileges for the database.

Syntax :
GRANT privileges ON object TO user;

Privileges: Any dml or ddl command

Object: The name of the database object that you are granting permissions for. In the case of granting privileges on a table, this would be the table name.

User: The name of the user that will be granted these privileges.

For example**,** if you wanted to grant SELECT, INSERT, UPDATE, and DELETE privileges on a table called employees to a user name smith, you would run the following GRANT statement:
GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO smith;

REVOKE COMMAND

The revoke command removes user access rights or privileges to the database objects.

The syntax for the REVOKE command is:
REVOKE privilege_name ON object_name FROM {User_name | PUBLIC | Role_name}

Example:
REVOKE SELECT ON employee FROM user1 This command will revoke a SELECT privilege on employee table from user1.

TCL

TCL (Transaction Query Language)

The TCL commands for specifying the beginning and ending of transactions.These are used to manage the changes made by DML statements.

Commands:

Commit : Permanently save any transaction into the database.
Rollback : Restores the database to the last committed state.

COMMIT COMMAND

COMMIT command saves all the work done.
It ends the current transaction and makes permanent changes during the transaction.

Syntax:
commit;

ROLLBACK COMMAND

ROLLBACK command restores database to original since the last COMMIT.
It is used to restore the database to the last committed state.

Syntax :
ROLLBACK TO SAVEPOINT <savepoint_name>;

Example:
ROLLBACK TO SAVEPOINT no_update;

INTERVIEW QUESTIONS

Q 1. What is DDL? (TCS)

DDL (Data Definition Language): It is used to define the database structure such as tables. It includes three statements such as CREATE, ALTER, and DROP.

Q 2.Difference between DDL, DML commands?(ADOBE)

Data Definition Language (DDL) statements are used to define the database structure or schema.

Data Manipulation Language (DML) statements are used for managing data within schema objects.

Q 3. Explain different commands in DML. (FLIPKART)

The DML statements used for managing data within schema objects which deal with data manipulation.

DML commands are not auto-committed. It means changes are not permanent to the database, they can be rolled back.

Q 4. Difference between commit and rollback. (MASTERCARD)

COMMIT validates the modifications made by the current transaction.

COMMIT occurs when the transaction gets executed successfully.

ROLLBACK erases the modifications made by the current transaction.

ROLLBACK occurs when the transaction is aborted in the middle of the execution.

Q 5. Explain Grant and Revoke. (MYNTRA)

Grant: It used to provide any user access privileges or other privileges for the database.

Revoke:The revoke command removes user access rights or privileges to the database objects.

Indexing

A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure.
It is a data structure that is added to a file to provide faster access to the data.
It reduces the number of blocks that the DBMS has to check.

How do DBMS access data?

The operations read, modify, update, and delete are used to access data from the database. DBMS must first transfer the data temporarily to a buffer in main memory. Data is then transferred between disk and main memory into units called blocks.

Two Types Of Indices

Ordered index (Primary index or clustering index) – which is used to access data sorted by order of values.

Hash index (secondary index or non-clustering index) - used to access data that is distributed uniformly across a range of buckets.

Ordered Index

In an ordered index, index entries are stored sorted on the search key value. Example, the author catalog in the library. The indices are usually sorted to make searching faster.

Example: Suppose we have a student table with thousands of records and each of which is 10 bytes long. If their IDs start with 1, 2, 3....and so on and we have to search students with ID-555.
In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 555*10=5550 bytes.

Primary index:

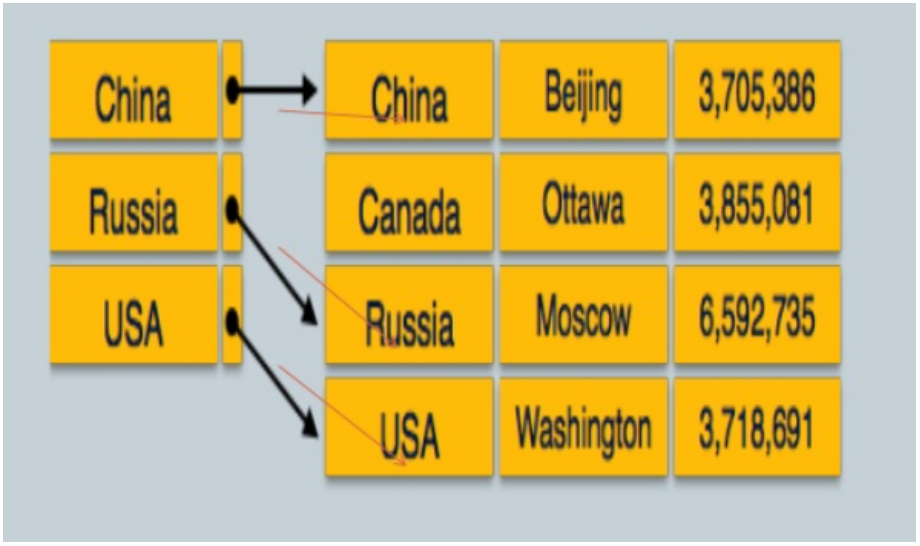
In a sequentially ordered file, the index whose search key specifies the sequential order of the file. Also called clustering index The search key of a primary index is usually but not necessarily the primary key. The primary index can be classified into two types: Dense index and Sparse index.

Dense index:

- An index record appears for **every** search key value in file.
- This record contains a search key value and a pointer to the actual record.
- It needs more space to store the index record itself. The index records have the search key and a pointer to the actual record on the disk.

Sparse Index:

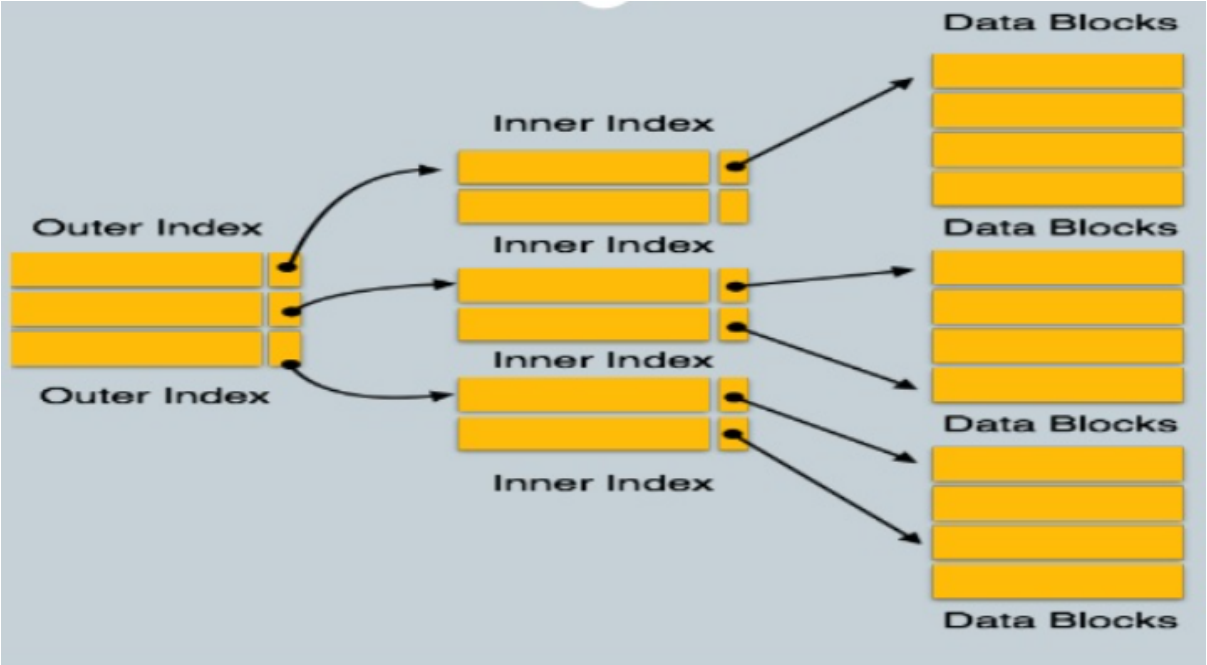
It contains index records for only some search-key values.
Applicable when records are sequentially ordered on search-key.
To locate a record with search-key value K we: Find index record with largest search-key value < K Search file sequentially starting at the record to which the index record points.



Compared to dense indices:
Less space and less maintenance overhead for insertions and deletions.
Generally slower than dense index for locating records.
Good tradeoff: sparse index with an index entry for every block in file, corresponding to least search-key value in the block.

Multilevel Indexing

In this method, we can see that index mapping growth is reduced to a considerable amount. But this method can also have the same problem as the table size increases. In order to overcome this, we can introduce multiple levels between primary memory and secondary memory. This method is also known as multilevel indexing. In this method, the number of secondary level indexes is two or more.



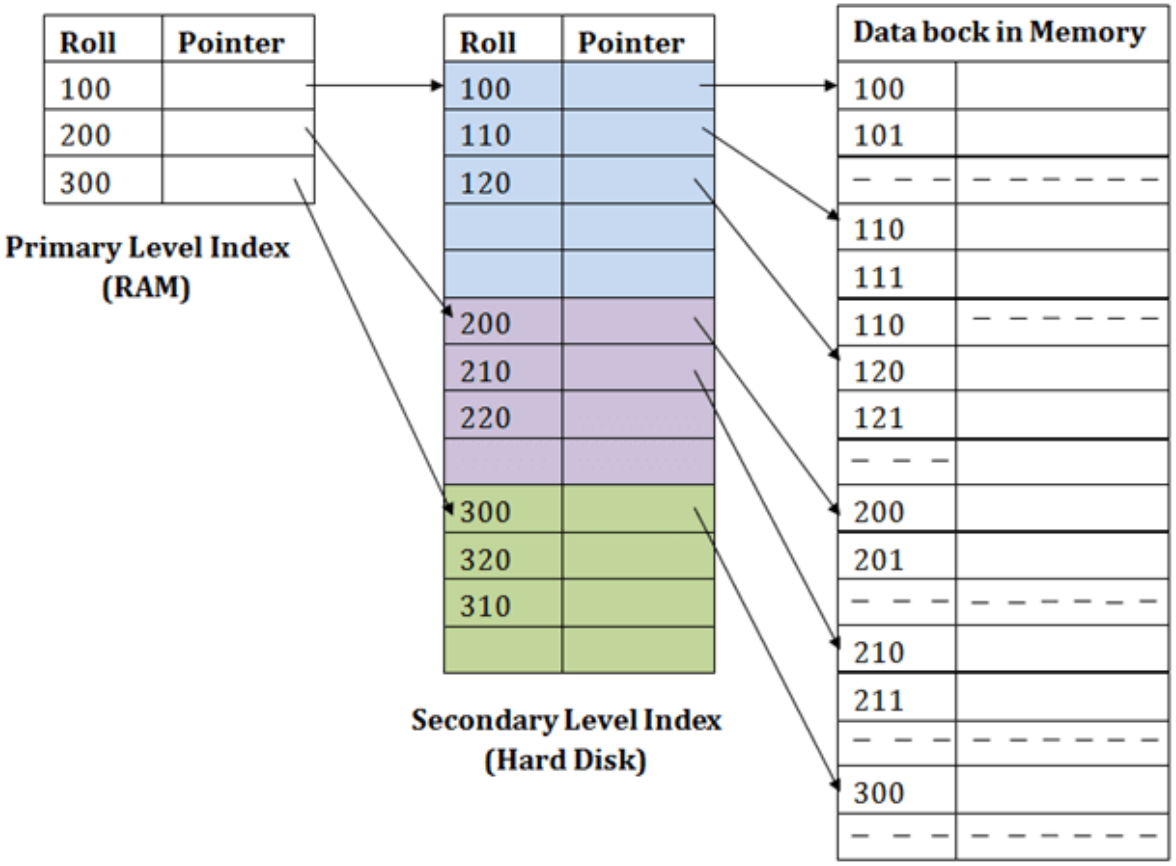
Secondary index/Non-cluster

An index whose search key specifies an order different from the sequential order of the file. Also called non-clustering index.

If the search key of a secondary index is not a candidate key, it is not enough to point to just the first record with each search-key value because the remaining records with the same search-key value could be anywhere in the file. Therefore, a secondary index must contain pointers to all the records.

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

Secondary indices improve the performance of queries on non-primary keys.



For example:

- If you want to find the record of roll 111 in the diagram, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
- Then in the second index level, again it does max (111) <= 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111.
- This is how a search is performed in this method. Inserting, updating or deleting is also done in the same manner.

Interview Questions

Q 1. What kind of data structure is an index? (Adobe)
In most of the cases, B-Tree data structure is used to store the indexes. This is because of the time efficiency of B-Trees. In other words, B-trees are traversed, searched, inserted, deleted and updated in logarithmic time.

Q 2. A table can have multiple non-clustered indexes, but only one clustered index. Why?(Amazon)
If a table has a clustered index, then its records are sorted based on the column on which the clustered index is created. Therefore, if we create multiple clustered indexes, we cannot sort the table record for all those columns. Only one time sorting based on one column for which clustering is done is valid.

Q 3.What is index and types of index?
Two main types of indexing methods are 1)Primary Indexing 2) Secondary Indexing.

Q 4.Is the primary key clustered index? (Goldman Sachs)
A primary key is a unique index that is clustered by default. By default means that when you create a primary key, if the table is not clustered yet, the primary key will be created as a clustered unique index.

Q 5.Why are B tree indexes popular? (Mastercard)
The B-tree enables the database to find a leaf node quickly. The tree traversal is a very efficient operation so efficient that I refer to it as the first power of indexing. It works almost instantly even on a huge data set.

Interview Questions

Q 1. What is meant by ACID properties in DBMS? (TCS)
ACID stands for Atomicity, Consistency, Isolation, and Durability in a DBMS; these are those properties that ensure a safe and secure way of sharing data among multiple users.

Q 2. What are conflict serializable schedules? (Adobe)
Ans: A schedule S of n transactions is serializable if it is equivalent to some serial schedule of the same n transactions.

Q 3. What are conflict equivalent schedules? (Amazon)
Ans: Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

Q 4 . What is view equivalence? (HSBC)
Ans: Two schedules S and S' are said to be view equivalent if the following three conditions hold :
1. Both S and S' contain the same set of transactions with the same operations in them.
2. If any read operation read(x) reads a value written by a write operation or the original value of x the same conditions must hold in the other schedule for the same read(x) operation.
3. If an operation write1(y) is the last operation to write the value of y in schedule S then the same operation must be the last operation in schedule S'.

Q 5. What is view serializable? (Myntra)
Ans: A schedule is said to be view serializable if it is view equivalent with some serial schedule.

Q 6. What is the result equivalent?
Ans: Two schedules are called result equivalent if they produce the same final state of the database.

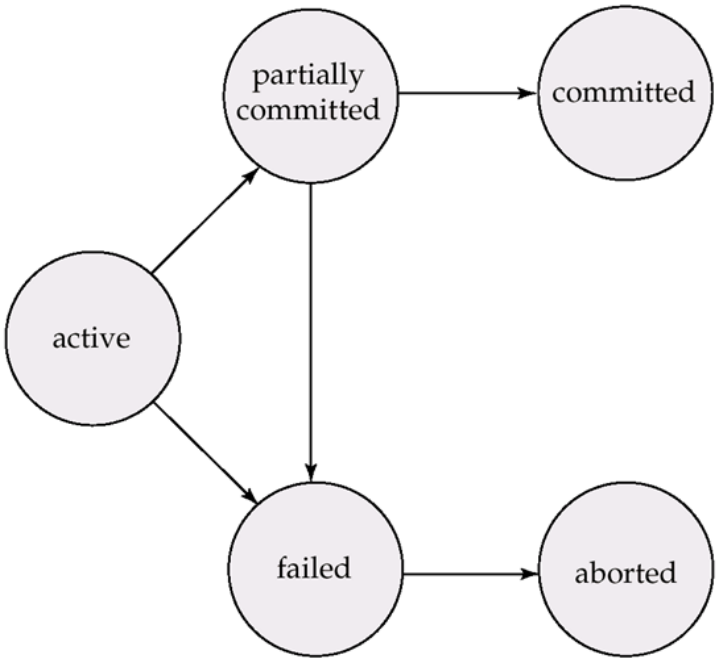
Transaction

What is Transaction?

A transaction is a unit of program execution that accesses and possibly updates one or more data items in the database.
A group of tasks where the task is a minimum processing unit which cannot be divided further.

State of Transaction

Active: The initial state, the transaction stays in this state while it is executing.
Partially committed: After the final statement has been executed.
Failed: After the discovery that normal execution can no longer proceed.
Aborted: After the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
Committed: After successful completion.



ACID Properties

Atomicity:
Atomicity means that transactions execute as a whole.
DBMS to guarantee that either all of the operations are performed or none of them.

Example: transfer of funds between banks: either withdraw+deposit both execute successfully or none of them, In case of failure Database remains unchanged.

Consistency:
Consistency means the database is in legal state when the transaction begins and when it ends.
Only valid data will be written in the database and transactions cannot break the rules of database e.g Integrity Constraints: Primary keys, foreign key.

Example of this is that Transaction cannot end with a duplicate primary key in the table.

Integrity:
Multiple transactions running at the same time do not impact each other's execution.
Transactions don't see other transaction's uncommitted changes.

Isolation level defines how deep transactions isolate from one another.

Isolation's example: if two or more people try to buy the last copy of a product, just one of them will succeed.

DURABILITY
Durability means if a transaction is committed it becomes persistent, cannot be lost or undone.
Ensured by use of database transaction logs.

Example of this is that after funds are transmitted and the committed power supply at the DB server is lost, Transactions stay persistent (No data is lost).

Concurrent Transaction

Concurrent execution means running side by side or parallely of transactions.

Advantages of Concurrent execution are:

Improved throughput & Resource utilization:i.e. no. of transactions executed increases in a given amount of time & the processor is utilized properly.

Reduced Waiting time:The unpredictable delays in running transactions as well as the average response time is reduced.

Schedules

Schedules:
Sequences that indicate the chronological order in which instructions of concurrent transactions are executed
A schedule can have many transactions in it, each consisting of a number of instructions/tasks.

Serial schedule:
A schedule S is serial if, for every transaction T participating in the schedule, all the operations of T are executed consecutively in the schedule. Otherwise, the schedule is called a non serial schedule.

Serializable schedule:
Serializable schedules are always considered to be correct when concurrent transactions are executed.
The main difference between the serial schedule and the serializable schedule is that in serial schedule, no concurrency is allowed whereas in serializable schedule, concurrency is allowed.

Example:
Let *T*1 transfer $50 from *A* to *B*, and *T*2 transfer 10% of the balance from *A* to *B.* The following is a serial schedule, in which *T*1 is followed by *T*2.

Schedule 1

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

Let *T*1 and *T*2 be the transactions defined previously. The following schedule 2 is not a serial schedule, but it is *equivalent* to Schedule 1.

Schedule 2

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

In both Schedule 1 and 2, the sum A + B is preserved.
The following concurrent schedule does not preserve the value of the sum *A* + *B*.

Schedule 3

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Serializability

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transaction are interleaved with some other transaction.

What is a conflict?

A pair of Operations in a schedule such that if their order is interchanged then the behavior of at least one of the transactions may change.

Operations are conflict, if they satisfy all three of the following conditions :

They belong to different transactions.

They access the same data item .

At least one of the operations is a write operation.

Conflict Equivalence

Schedules are conflict equivalent if they can be transformed one into another by a sequence of non conflicting interchanges adjacent actions.

Conflict Serializability

Instructions l_i and l_j of transactions T_i and T_j respectively, conflict if and only if there exists some item Q accessed by both l_i and l_j, and at least one of these instructions wrote Q.

1.l_i = read(Q), l_j = read(Q). l_i and l_j don't conflict.

2.l_i = read(Q), l_j = write(Q). They conflict.

3.l_i = write(Q), l_j = read(Q). They conflict

4.l_i = write(Q), l_j = write(Q). They conflict

Intuitively, a conflict between l_i and l_j forces a (logical) temporal order between them. If l_i and l_j are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

If a schedule S can be transformed into a schedule S´ by a series of swaps of non-conflicting instructions, we say that S and S´ are conflict equivalent.

We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

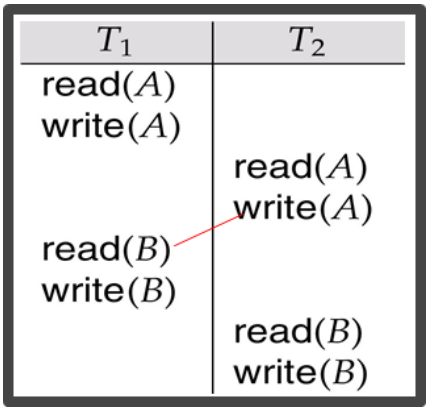
Example of a schedule that is not conflict serializable:

T3	**T4**
READ(Q)	

	WRITE(Q)
WRITE(Q)	

We are unable to swap instructions in the above schedule to obtain either the serial schedule < T_3, T_4 >, or the serial scheSchedule 4 below can be transformed into a serial schedule where T_2 follows T_1, by series of swaps of non-conflicting instructions.
Therefore Schedule 4 is conflict serializable.
dule < T_4, T_3 >.

Schedule 4



View Serializability

Let S and S´ be two schedules with the same set of transactions. S and S´ are view equivalent if the following three conditions are met, for each data item Q,

1.If in schedule S, transaction T_i reads the initial value of Q, then in schedule S' also transaction T_i must read the initial value of Q.
2.If in schedule S transaction T_i executes read(Q), and that value was produced by transaction T_j (if any), then in schedule S' also transaction T_i must read the value of Q that was produced by the same write(Q) operation of transaction T_j .
3.The transaction (if any) that performs the final write(Q) operation in schedule S must also perform the final write(Q) operation in schedule S'.

As can be seen, view equivalence is also based purely on reads and writes alone

A schedule S is viewed as serializable if it is view equivalent to a serial schedule.
Every conflict serializable schedule is also view serializable.
Below is a schedule which is view-serializable but not conflict serializable.

T3	T4	T6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

Concurrency control

The technique used to protect data when multiple users are accessing the same data concurrently (same time) is called concurrency control.

Types of concurrency control protocol
Concurrency control protocols can be broadly divided into two categories :

1. Lock-based protocols
2. Time-stamp based protocols

Lock-based Protocols :
Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it.

Data items can be locked in two modes :

Exclusive (X) mode: Data items can be both read as well as written. X-lock is requested using lock-X instruction.

Shared (S) mode: Data items can only be read. S-lock is requested using lock-S instruction.

Lock-compatibility matrix

	S	X
S	true	false
X	false	false

A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by another transaction.
Any number of transactions can hold shared locks on an item, But if any transaction holds an exclusive on the item no other transaction may hold any lock on the item.

If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted.

Example:

```
T1: lock-S(A); // Grant-S(A,T1)
read (A);
unlock(A);
lock-S(B); // Grant-S(B,T1)
read (B);
unlock(B);
display(A+B)
```

Timestamp based Protocol

Each transaction is issued a timestamp when it enters the system. If an old transaction Ti has time-stamp TS(Ti), a new transaction Tj is assigned time-stamp TS(Tj) such that TS(Ti) <TS(Tj).

The timestamp of transaction Tiis denoted as TS(Ti).
Read time-stamp of data-item X is denoted by R-timestamp(X).
Write time-stamp of data-item X is denoted by W-timestamp(X).

Two Phase Locking Protocol

This is a protocol which ensures conflict-serializable schedules.
Phase 1: Growing Phase
In this phase transactions may obtain locks but may not release locks.

Phase 2: Shrinking Phase
In this phase transaction may release locks,but may not obtain locks.

MERITS OF 2 PHASE LOCKING PROTOCOL

The two phase locking protocol ensures conflict serializability.
Consider any transaction, the point in the schedule where the transaction has obtained it's final lock is called the lock point of the transaction.
Now, transactions can be ordered according to their lock points. This ordering is a serializability ordering for the transactions.

Two-phase locking does not ensure freedom from deadlocks.

Cascading roll-back is possible under two-phase locking. To avoid this, follow a modified protocol called strict two-phase locking. Here a transaction must hold all its exclusive locks till it commits/aborts.

Rigorous two-phase locking is even stricter: here all locks are held till commit/abort. In this protocol transactions can be serialized in the order in which they commit.

Interview Question

Q 1. Describe concurrency control? (Morgan stanley)

Concurrency control is the process of managing simultaneous operations on a database so that database integrity is not compromised. There are two approaches to concurrency control.
They are Locking (controlling access to data items using locks) and Versioning (using Multi-Version Concurrency Control) respectively.

Q 2.What are different locks? (Adobe)

Shared Lock
Exclusive Lock

Q 3.What is a shared lock? (Infosys)

It is also known as Read-only lock. As the name suggests it can be shared between transactions because while holding this lock the transaction does not have the permission to update data on the data item. S-lock is requested using lock-S instruction.

Q 4.What is an exclusive lock? (Vmware)

Data items can be both read as well as written.This is Exclusive and cannot be held simultaneously on the same data item. X-lock is requested using lock-X instruction.

Q 5. What is starvation? (Myntra)

Starvation or Livelock is the situation when a transaction has to wait for an indefinite period of time to acquire a lock.