

Question Bank Answers (DBMS – IAE1)

CONTENT WARNING: READING THIS DOCUMENT MAY CAUSE SUDDEN BURSTS OF INTELLIGENCE.
PROCEED WITH CAUTION.

1. Define DBMS and state its 5 application in real life.

Ans.

DBMS Definition :

A Database Management System (DBMS) is software designed to store, retrieve, define, and manage data in a database. DBMS software primarily functions as an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.

Application :

1. Transaction Management
2. Storage Management
3. Database Administrator
4. Database Users
5. Overall System Structure

Q2. State and explain ACID property of DBMS.

Ans.

ACID Properties in DBMS :

Atomicity :

The entire transaction takes place at once or doesn't happen at **all**.

Consistency :

The database must be consistent before and after the transaction.

Isolation :

Multiple Transactions occur independently without interference.

Durability :

The changes of a successful transaction occurs even if the system failure occurs.

Q3. Explain Types of DBMS users & role of DBA in detail.

Ans.

Types of DBMS Users:

1. End Users: These users interact directly with the database through applications. They are categorized into:

- **Casual Users:** Use the database occasionally with little understanding of its structure.
- **Naive Users:** Perform routine operations using predefined queries and forms.
- **Sophisticated Users:** Write complex queries to fulfill their needs, typically business analysts.
- **Standalone Users:** Work with personal databases using tools like MS Access.

2. Application Programmers: They write application programs that interact with the database, using languages like Java or SQL to retrieve, insert, update, or delete data.

3. Database Designers: These users design the database schema and structure, defining entities, relationships, and constraints to ensure the database meets the organization's requirements.

4. System Analysts: They analyze user requirements and translate them into a database design.

5. Database Administrators (DBA): DBAs manage the entire database system. They play a crucial role in the smooth operation and maintenance of the database.

Role of a DBA:

- **Database Security:** Managing user access and permissions to ensure data is protected against unauthorized access.

- **Performance Tuning:** Monitoring and optimizing database performance for efficient data retrieval and processing.

- **Backup and Recovery:** Setting up regular backups and ensuring the database can be restored in case of data loss.

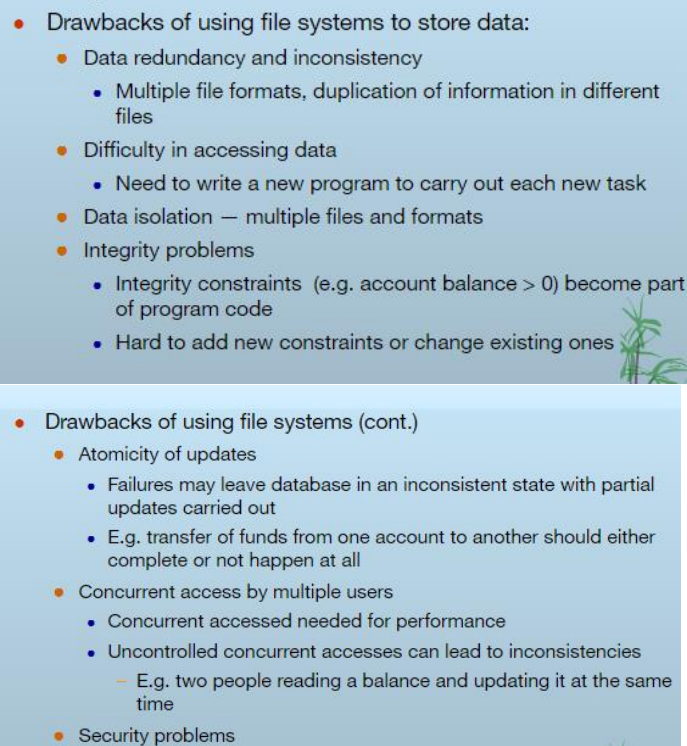
- **Data Integrity:** Ensuring data is accurate and consistent by managing constraints and resolving inconsistencies.

- **Maintenance:** Regularly updating and maintaining the database, including applying patches and upgrades.

Q4 Give the advantage of DBMS over file system

Ans.

Advantages of DBMS Over File System :

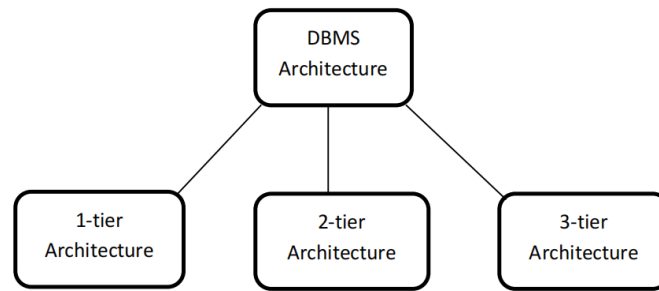
- 
- Drawbacks of using file systems to store data:
 - Data redundancy and inconsistency
 - Multiple file formats, duplication of information in different files
 - Difficulty in accessing data
 - Need to write a new program to carry out each new task
 - Data isolation — multiple files and formats
 - Integrity problems
 - Integrity constraints (e.g. account balance > 0) become part of program code
 - Hard to add new constraints or change existing ones
 - Drawbacks of using file systems (cont.)
 - Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - E.g. transfer of funds from one account to another should either complete or not happen at all
 - Concurrent access by multiple users
 - Concurrent accesses needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - E.g. two people reading a balance and updating it at the same time
 - Security problems

Q5. Describe the overall architecture of DBMS with the diagram

Ans.

DBMS Architecture :

DBMS architecture defines how users interact with the database, and it can be **single-tier** or **multi-tier**. Logically, there are two main types: **2-tier** and **3-tier** architecture.



1-Tier Architecture:

1. The database is directly accessible to the user.
2. Used mainly in **development environments** where programmers interact directly with the DBMS.
3. No intermediate layer or security abstraction.

2-Tier Architecture:

1. Involves a client-server model.
2. The client (user interface) interacts with the server (DBMS) to process queries.
3. Common for small networks, like desktop applications interacting with remote databases.

3-Tier Architecture:

- a. Includes an additional layer called the application server (middleware).
- b. Separates the client, application logic, and database.
- c. Improves scalability, security, and maintenance.
- d. Common in web-based applications, where the client (browser) communicates with a web server, which interacts with the database server

Q6. What is a view?How it is created and stored.

Ans.

- A database view is a Logical or virtual table based on a query.
- It is useful to think of a view as a stored query.
- Views are created through use of a CREATE VIEW command that incorporates use of the SELECT statement.
- Views are queried just like tables.

Syntax :

```
CREATE VIEW employee_parking (parking_space, last_name, first_name, ssn)
ASSELECT emp_parking_space, emp_last_name, emp_first_name, emp_ssn
FROM employee
ORDER BY emp
parking_space;
```

Example :

```
SELECT * FROM employee_parking;
```

```
PARKING_SPACE LAST_NAME FIRST_NAME NUMBER
```

1	Bardoloi	Bi joy	999666666
3	Joyner	Suzanne	999555555
32	Zhu	Waiman	999444444

Q7.Explain various data types used in SQL

Ans.

- ☐ String data types/Character data types

Data type syntax	Explanation
char(Size)	Where size is the number of character to store.
Long	Fixed length string. Space padded
Raw	Variable-length string(Backward Compatible)
	Variable-length binary string

☐ Numeric datatypes

Data type syntax	Explanation
Float	A floating point value
Integer	An integer number

☐ Date/Time date types

Data type syntax	Explanation
Date	Includes year,month,day
Timestamp(fractional seconds precision)	Includes year,month,day,hour,minute & second

Q8. Write and explain GRANT & REVOKE commands

Ans.

Grant : The GRANT command is used to give privileges (permissions) to users or roles on database objects. These privileges determine what actions the users can perform on those objects.

Syntax :

GRANT<privilege_type> ON <object_name> TO USER;

Example :

GRANT SELECT, INSERT ON Employees TO John;

Revoke : The REVOKE command is used to take back privileges that were previously granted to users or roles.

Syntax :

REVOKE<privilege_type> ON <object_name> FROM USER[cascade];

Example:

REVOKE ALL PRIVILEGES ON Employees FROM John;

Q9. How to create a role and assign password and privileges.

Ans.

Syntax :

role<role_name>,<password>;

Example :

Create role 'role_test', 'best123';

Q10. List & explain SET operators.

Ans.

1. UNION: Combines the results of two queries and removes duplicate rows.

Example :

```
SELECT column1 FROM
table1 UNION
SELECT column1 FROM table2;
```

2. UNION ALL: Combines the results of two queries but includes all duplicate rows.

Example :

```
SELECT column1 FROM
table1 UNION ALL
SELECT column1 FROM table2;
```

3. INTERSECT: Returns only the rows that are present in both queries.

Example :

```
SELECT column1 FROM  
table1 INTERSECT  
SELECT column1 FROM table2;
```

4. EXCEPT (or MINUS in some SQL databases): Returns rows from the first query that are not present in the second query. It removes duplicates, similar to INTERSECT.

Example :

```
SELECT column1 FROM  
table1 EXCEPT  
SELECT column1 FROM table2;
```

Q11. Explain Group by and having clause.

Ans.

Group by clause:

- The function to divide the tuples into groups and returns an aggregate for each group.
- Usually, it is an aggregate function's companion.

Syntax :

```
SELECT food, sum (sold) as total Sold  
FROM Food Cart group by food;
```

Having clause:

- The substitute of WHERE for aggregate functions
- Usually, it is an aggregate function's companion

Syntax :

```
SELECT food, sum (sold) as totalSold  
FROM Food Cart group by food  
having sum (sold) > 450;
```

Q12. What do you mean by super key and candidate key.

Ans.

1. Super Key:

- A super key is a set of one or more columns (attributes) that uniquely identifies a row in a table.
- It can contain additional columns beyond what is necessary for uniqueness. Essentially, any combination of columns that can uniquely identify rows qualifies as a super key.
- For example, in a table with columns A, B, and C, both (A, B) and (A, B, C) could be super keys if each combination can uniquely identify a row.

2. Candidate Key:

- A candidate key is a minimal super key, meaning it is a super key with no redundant columns. It is a set of columns that uniquely identifies rows and cannot be reduced further without losing uniqueness.
- There can be multiple candidate keys for a table, and one of them is chosen as the primary key.
- For instance, if (A, B) is a super key and removing either A or B makes it no longer unique, then (A, B) is a candidate key. If (C) alone also uniquely identifies rows, then (C) could be another candidate key.

Q13. Explain different types of constraints in SQL

Ans.

1. Primary Key Constraint:

- Ensures that each row in a table is unique and not null.
- A table can have only one primary key, which can consist of one or more columns.
- **Example:**

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY
    KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```

2. Foreign Key Constraint:

- Establishes a relationship between two tables by linking the primary key of one table to a column in another table.
- It ensures that the value in the foreign key column matches a value in the primary key column of the referenced table.

- Example:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY
    KEY,
    EmployeeID INT,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

3. Unique Constraint:

- Ensures that all values in a column or a group of columns are unique across the table.
- Unlike a primary key, a table can have multiple unique constraints.

- Example:

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY
    KEY, Email VARCHAR(100)
    UNIQUE
);
```

4. Not Null Constraint:

- Prevents null values from being inserted into a column, ensuring that the column always contains a value.

- Example:

```
CREATE TABLE Products (
    ProductID INT PRIMARY
    KEY,
    ProductName VARCHAR(100) NOT NULL
);
```

5. Check Constraint:

- Ensures that all values in a column satisfy a specific condition or criteria.

- Example:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY
    KEY, Age INT CHECK (Age
    >= 18)
);
```

6. Default Constraint:

- Assigns a default value to a column if no value is provided during an insert operation.

- Example:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY
    KEY,
    OrderDate DATE DEFAULT GETDATE()
);
```

Q14. Explain primary key constraint with syntax.

Ans.

Primary Key Constraint:

- Ensures that each row in a table is unique and not null.
- A table can have only one primary key, which can consist of one or more columns.

- Example:

```
CREATE TABLE Employees (  
    EmployeeID INT PRIMARY  
    KEY,  
    FirstName  
    VARCHAR(50),  
    LastName  
    VARCHAR(50)  
);
```

Q15. Explain foreign key constraint with syntax.

Ans.

Foreign Key Constraint:

- Establishes a relationship between two tables by linking the primary key of one table to a column in another table.
- It ensures that the value in the foreign key column matches a value in the primary key column of the referenced table.

- Example:

```
CREATE TABLE Orders (  
    OrderID INT PRIMARY  
    KEY,  
    EmployeeID INT,  
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)  
);
```

Q16. Consider the following relation and answer the given queries.

Author (Author-Name, country)

Books (ISBN, title, Pub-year, unit-price, Author-name, Publisher - name)

Ans.

1. Get the author country details for the book 'DBMS':

```
SELECT  
A.country FROM  
Author A  
JOIN Books B ON A.Author-Name = B.Author-  
Name WHERE B.title = 'DBMS';
```

2. Get the details of book title starting with 'D':

```
SELECT *  
FROM Books  
WHERE title LIKE 'D%';
```

3. Get the author details whose name has 'A' in the second place:

```
SELECT *  
FROM Author  
WHERE Author-Name LIKE '_A%';
```

4. Define primary key AUTHOR NAME in the author table and foreign key on Author-name of books table:

-- Define Primary Key in Author table

```
ALTER TABLE Author  
ADD PRIMARY KEY (Author-Name);
```

-- Define Foreign Key in Books table

```
ALTER TABLE Books  
ADD CONSTRAINT fk_author  
FOREIGN KEY (Author-Name) REFERENCES Author(Author-Name);
```

5. Get the sum of book price published in each year:

```
SELECT Pub-year, SUM(unit-price) AS
Total_PriceFROM Books
GROUP BY Pub-year;
```

6. Get the average price of book published in each year:

```
SELECT Pub-year, AVG(unit-price) AS
Average_PriceFROM Books
GROUP BY Pub-year;
```

7. Get the count of books published in each year where the count should be more than 10:

```
SELECT Pub-year, COUNT(*) AS Book_Count
FROM
Books
GROUP BY Pub-year
HAVING COUNT(*) >
10;
```

8. Get the maximum price of books published each year and arrange the output in increasing order of price:

```
SELECT Pub-year, MAX(unit-price) AS
Max_PriceFROM Books
GROUP BY Pub-year
ORDER BY Max_Price
ASC;
```

9. Get the minimum price of books published each year and sort the output in decreasing order of year:

```
SELECT Pub-year, MIN(unit-price) AS
Min_PriceFROM Books
GROUP BY Pub-year
ORDER BY Pub-year
DESC;
```

10. Insert a new row for "C programming" by Yashwant Kanetkar costing 250 in 2012 published by McGrawhills:

```
INSERT INTO Books (ISBN, title, Pub-year, unit-price, Author-Name, Publisher-name)
VALUES ('ISBN12345', 'C programming', 2012, 250, 'Yashwant Kanetkar', 'McGrawhills');
```

11. Delete the book titled 'Operating system':

```
DELETE FROM Books
WHERE title = 'Operating system';
```

12. Add a column of gender for the author:

```
ALTER TABLE Author
ADD COLUMN gender VARCHAR(10);
```

Q17. Define: Entity set, Attribute, Relation.

Ans.

1. Entity Set : An entity set is a collection of similar entities that share the same properties or characteristics. In the context of a database, an entity is an object or thing in the real world that is distinguishable from other objects. For example, a "Students" entity set would consist of all the students in a database.

2. Attribute : An attribute is a property or characteristic of an entity. Attributes provide more details about the entities within an entity set. In a relational database, attributes are represented as columns in a table.

3. Relation : A relation is a table in a relational database. It represents a set of tuples (rows) that have

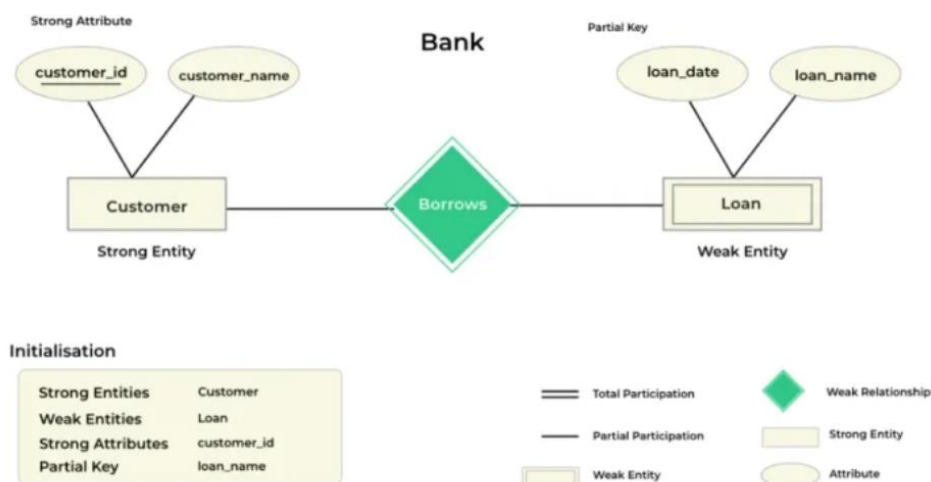
the same attributes (columns). In SQL, a relation is essentially a table that stores data about the entities. The term "relation" also refers to the logical connection between tables through keys (e.g., primary key and foreign key).

Q18. Explain Weak & strong entity set with example.

Ans.

1. Strong Entity Set

- Strong entity set always has a primary key.
- It is represented by a rectangle symbol.
- It contains a Primary key represented by the underline symbol.
- The member of a strong entity set is called as dominant entity set.
- Primary Key is one of its attributes which helps to identify its member.
- In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.
- The connecting line of the strong entity set with the relationship is single.



2. Weak Entity Set

- It does not have enough attributes to build a primary key.
- It is represented by a double rectangle symbol.
- It contains a Partial Key which is represented by a dashed underline symbol.
- The member of a weak entity set called as a subordinate entity set.
- In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
- The relationship between one strong and a weak entity set shown by using the double diamond symbol.
- The line connecting the weak entity set for identifying relationship is double.

Q19. Explain Type of attribute with example.

Ans.

1. Simple Attribute:

- **Definition:** A simple attribute is an attribute that cannot be divided into smaller components. It is atomic, meaning it holds a single value for an entity.

- **Example:** In a "Students" table, StudentID or FirstName would be a simple attribute because they represent single, indivisible values.

2. Composite Attribute:

- **Definition:** A composite attribute is an attribute that can be divided into smaller subparts, which represent more basic attributes with independent meanings.

- **Example:** An attribute like FullName could be considered composite, as it can be broken down into FirstName and LastName.

3. Single-Valued Attribute:

- **Definition:** A single-valued attribute is an attribute that holds only one value for each entity in the entity set.

- **Example:** The DateOfBirth attribute in a "Students" table would be single-valued, as each student has only one date of birth.

4. Multi-Valued Attribute:

- **Definition:** A multi-valued attribute can hold multiple values for a single entity.

- **Example:** An attribute like PhoneNumbers in a "Contacts" table might store multiple phone numbers for one contact. In SQL, this is usually handled by creating a separate related table to store each phone number.

5. Derived Attribute:

- **Definition:** A derived attribute is an attribute whose value is calculated or derived from other attributes.

- **Example:** An attribute like Age could be derived from DateOfBirth. Instead of storing Age in the database, it can be calculated when needed using the DateOfBirth attribute.

6. Key Attribute:

- **Definition:** A key attribute is an attribute that uniquely identifies each entity in an entity set. It's a unique identifier.

- **Example:** The StudentID in a "Students" table would be a key attribute since it uniquely identifies each student in the database.

7. Null Attribute

- **Definition:** A null attribute is one that doesn't have a value for a particular entity. This happens when an attribute's value is either unknown or not applicable.

- **Example:** An attribute like MiddleName in a "Students" table could be null if some students don't have a middle name.

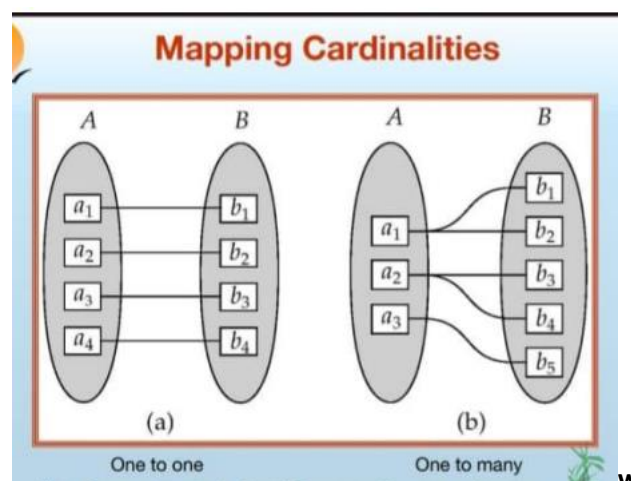
8. Stored Attribute:

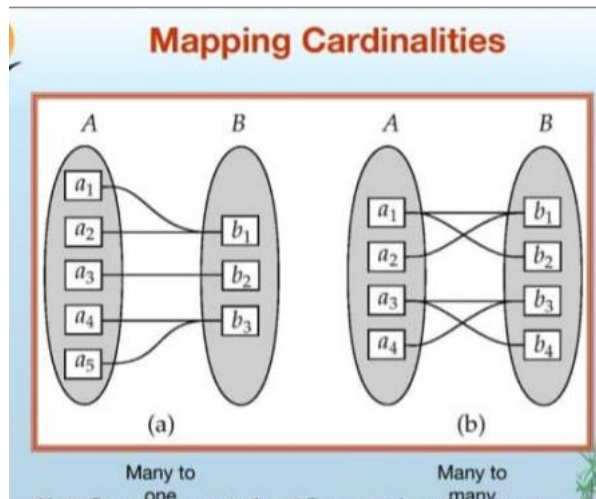
- **Definition:** A stored attribute is one that is physically stored in the database. It contrasts with derived attributes, which are calculated from other data.

- **Example:** FirstName and LastName in a "Students" table are stored attributes.

Q20. What is meant by mapping cardinality? Explain with example.

Ans.

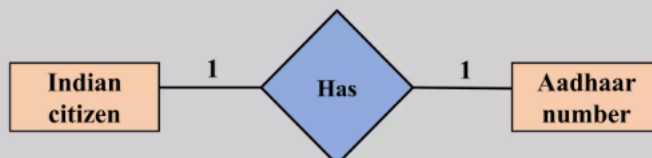




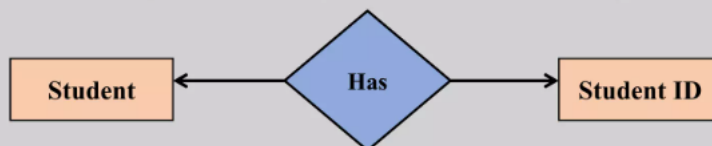
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
 - **One to one**
 - **One to many**
 - **Many to one**
 - **Many to many**

One-to-one Examples

One citizen has one Aadhar number, and the Aadhar number can only be used by one citizen.

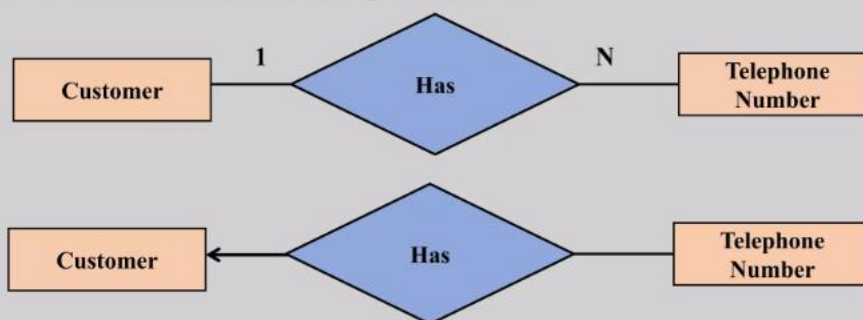


One student has only one student ID, and each student ID is assigned to only one student.



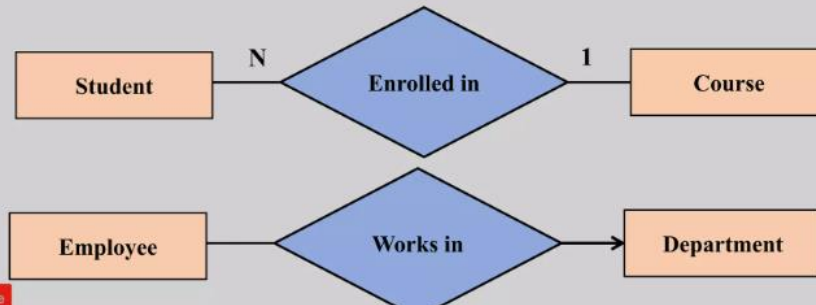
One-to-many Example

Each customer may have more than one telephone number, but each telephone number will be associated to only one customer.



Many-to-one Example

A student can take only one course, but one course can be taken by many students. So, the cardinality will be N to 1. It means that for one course there can be N students but for one student, there will be only one course.

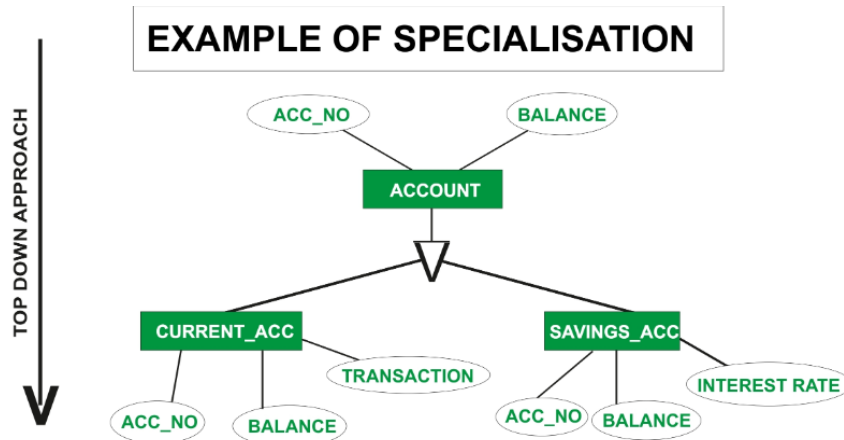


Q21. Write short note on: Specialization, Generalization, Aggregation.

Ans.

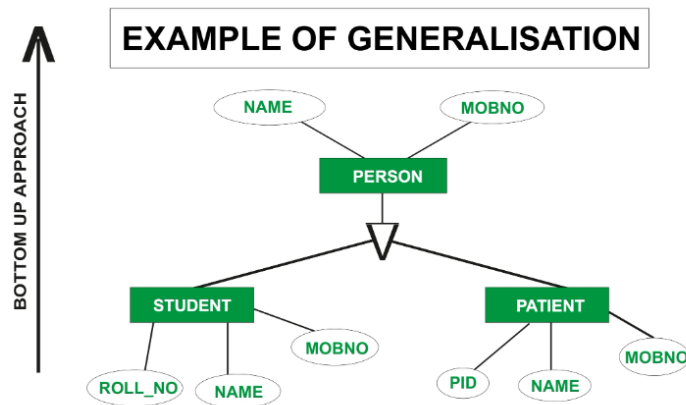
Specialization :

- Specialization is the process of defining one or more sub-entities of a higher-level entity. It allows for the creation of more specific entities based on the attributes of a general entity.
- It helps to model hierarchical relationships and capture the nuances of specific subtypes within a broader category.



Generalization :

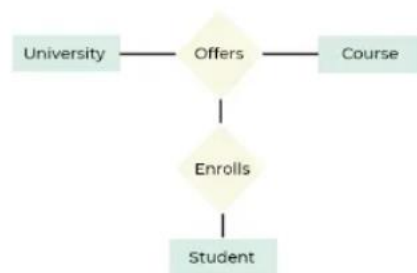
- Generalization is the reverse of specialization. It involves combining multiple specialized entities into a higher-level general entity, capturing the common attributes and relationships of the sub-entities.
- It helps simplify the model by reducing redundancy and focusing on shared characteristics.



Aggregation :

- Aggregation is a concept that allows us to treat a relationship set as a higher-level entity. It combines entities and their relationships into a single abstraction.
- It is useful for modeling complex relationships and situations where an entity is composed of other entities or relationships, thereby simplifying the overall model.

Aggregation in DBMS



Q22. Explain Like operator, between operator, IN operator and wild card characters in SQL.
Ans.

- **Like Operator :**

The Like Operator is used to search for specified pattern in a column

Syntax :

Select column_name (s)

From table_name

Where column_name Like pattern

- **In Operator :**

The In Operator allows you to specify multiple values in a where clause

Syntax :

Select column_name (s)

From table_name

Where column_name In (value1,value2,...)

- **Between Operator :**

The Between Operator selects a range of data between two values. The value can be number, text, dates

Syntax :

Select column_name (s)

From table_name Where

column_name

Between value1 and value2

- **Wildcards :**

SQL Wildcards can substitute for one or more characters when searching for data in a database.

NOTE : SQL Wildcards must be used with LIKE Operator.

With SQL the following wildcards can be used :

Wildcards

%

_

[charlist]

[^charlist] or [!charlist]

Description

A substitute for zero or more characters

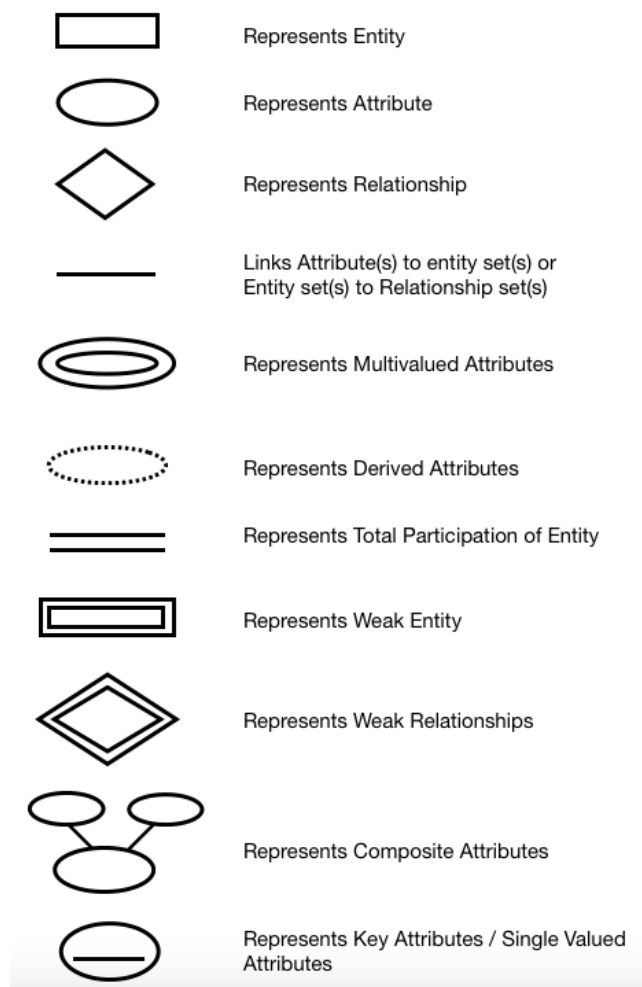
A substitute for exactly one character

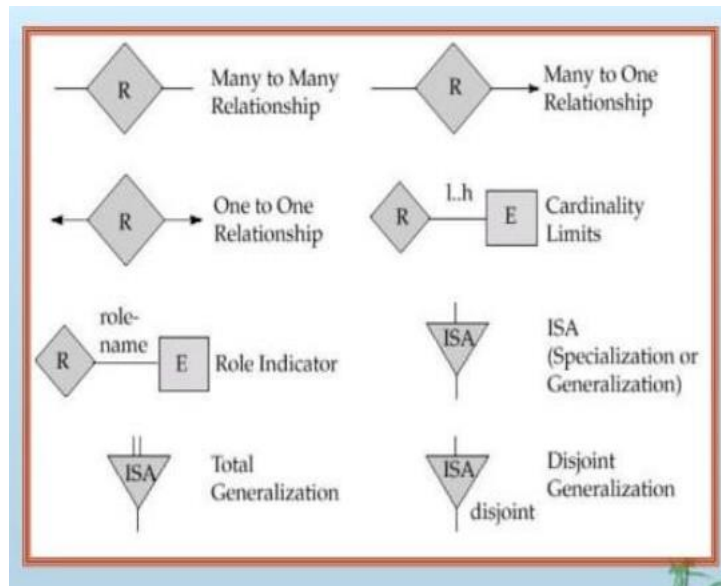
Any single character in charlist

Any single character not in charlist

Q23. Draw the notation used in ER model with their purpose

Ans





Q24. Draw the ER model for bank, hospital systems stepwise.

Ans

- **ER Model for a Bank System :**

Step 1: Identify Entities

Customer

Account

Loan

Branch

Step 2: Identify Relationships

Customer opens Account (1:N)

Customer borrows Loan (1:N)

Branch has Account (1:N)

Branch offers Loan (1:N)

Step 3: Identify Attributes

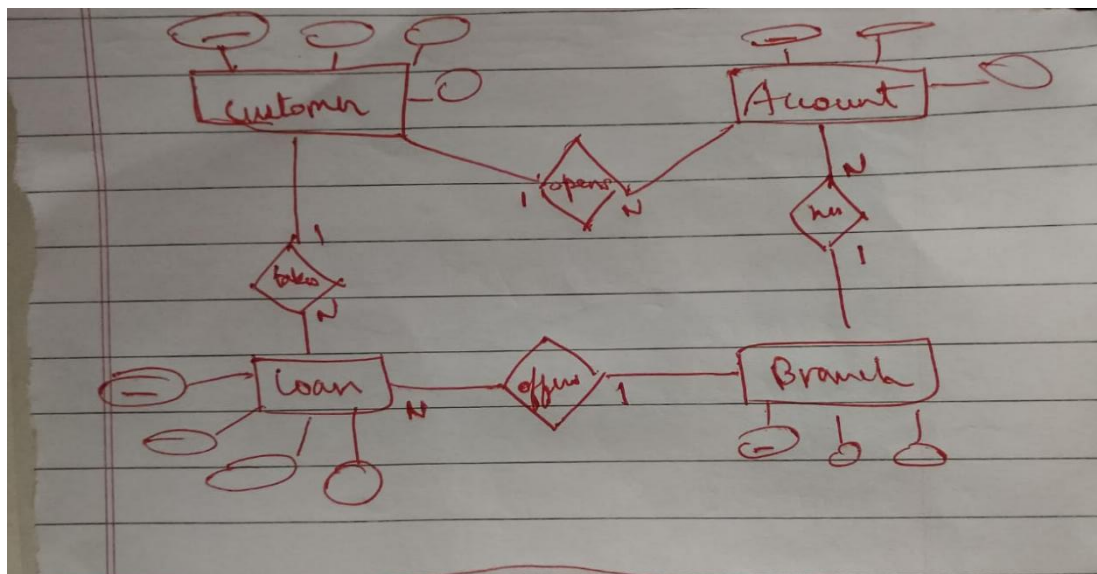
Customer: Customer ID (Key), Name, Address, Phone Number

Account: Account Number (Key), Account Type, Balance

Loan: Loan Number (Key), Loan Amount, Interest Rate

Branch: Branch ID (Key), Branch Name, Location

Step 4: Draw the ER Diagram



- **ER Model for a Hospital System :**

Step 1: Identify Entities

Patient
 Doctor
 Treatment
 Appointment
 Department

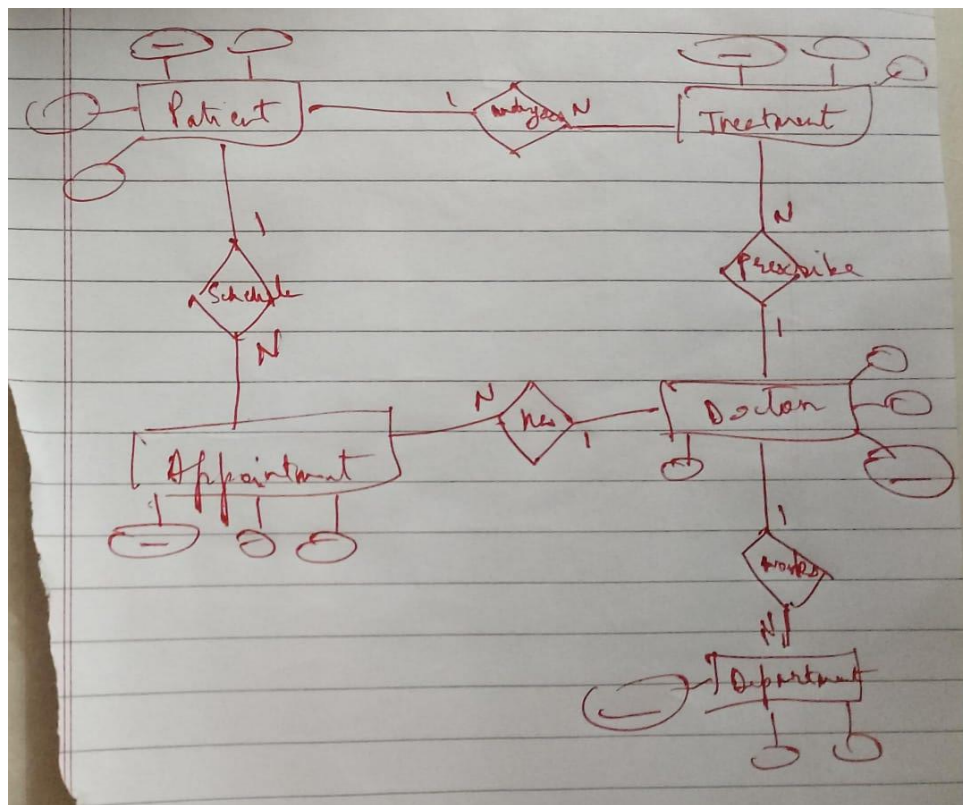
Step 2: Identify Relationships

Patient undergoes Treatment (1:N)
 Doctor prescribes Treatment (1:N)
 Patient schedules Appointment (1:N)
 Doctor has Appointment (1:N)
 Doctor works in Department (N:1)

Step 3: Identify Attributes

Patient: Patient ID (Key), Name, Address, Phone Number
 Doctor: DoctorID (Key), Name, Specialization, Phone Number
 Treatment: Treatment ID (Key), Treatment Description, Cost
 Appointment: Appointment ID (Key), Date, Time
 Department: Department ID (Key), Department Name, Location

Step 4: Draw the ER Diagram



Q25. Consider the following relations:

Employee (E-name, street, city)

WORK'S (E-name, company name, salary)

COMPANY (Company name, city)

Find the names of all employees who works for city Bank

Find the employee name and company names of all employee sorted in ascending order of company name & descending order of employee names of that company

Find employee who live in the same city as their company city.

Change the city of city bank to Delhi.

Ans.

1. Find the names of all employees who work for 'City Bank':

```
SELECT E-n
ameFROM
WORK'S
WHERE company name = 'City Bank';
```


2. Find the employee name and company names of all employees sorted in ascending order of company name & descending order of employee names of that company:

```
SELECT E-name, company  
name FROM WORK'S  
ORDER BY company name ASC, E-name DESC;
```

3. Find employees who live in the same city as their company city:

```
SELECT E.E-name  
FROM Employee E  
JOIN WORK'S W ON E.E-name = W.E-name  
JOIN COMPANY C ON W.company name = C.company name  
WHERE E.city = C.city;
```

4. Change the city of 'City Bank' to Delhi:

```
UPDATE COMPANY  
SET city = 'Delhi'  
WHERE company name = 'City Bank';
```

Q26. EXPLAIN DIFFERENT TYPES OF JOIN.

Ans.

1. INNER JOIN : The INNER JOIN keyword selects records that have matching values in both tables. It returns only the rows where there is a match between the tables.

Example:

```
SELECT Orders.OrderID,  
Customers.CustomerName FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

2. LEFT JOIN (or LEFT OUTER JOIN) : The LEFT JOIN returns all records from the left table (Table A), and the matched records from the right table (Table B). If there is no match, the result is NULL on the side of the right table.

Example :

```
SELECT Customers.CustomerName,  
Orders.OrderID FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

3. RIGHT JOIN (or RIGHT OUTER JOIN) : The RIGHT JOIN returns all records from the right table (Table B), and the matched records from the left table (Table A). If there is no match, the result is NULL on the side of the left table.

Example:

```
SELECT Orders.OrderID,  
Customers.CustomerNameFROM Orders  
RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

4. FULL JOIN (or FULL OUTER JOIN : The FULL JOIN returns all records when there is a match in either left (Table A) or right (Table B) table. If there is no match, the result is NULL from the table without a match.

Example :

```
SELECT Customers.CustomerName,  
Orders.OrderIDFROM Customers  
FULL JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

5. CROSS JOIN : The CROSS JOIN returns the Cartesian product of the two tables, i.e., it combines all rows from the first table with all rows of the second table.

Example :

```
SELECT Customers.CustomerName,  
Orders.OrderIDFROM Customers  
CROSS JOIN Orders;
```

6. SELF JOIN : A SELF JOIN is a regular join, but the table is joined with itself.

Example:

```
SELECT A.EmployeeName AS Employee, B.EmployeeName AS  
ManagerFROM Employees A, Employees B  
WHERE A.ManagerID = B.EmployeeID;
```