1. List the significant differences between DBMS and File System

Feature	DBMS (Database Management System)	File System
Data Management	Manages data using structured schemas, supports queries through SQL, and provides data integrity.	Manages data in files with no structured format.
Data Integrity	Enforces rules like primary keys, foreign keys, and constraints to maintain data consistency.	Limited to file-level integrity, no built-in checks for data consistency.
Security	provides user authentication, authorization and encryption.	Minimal security, relies on OS-level permissions.
Redundancy	uses normalization and relationships to minimize redundancy.	High reduandancy as the same data might be stored in multiple files.
Backup and Recovery	Automated, supports transaction logs for recovery.	Manual backup, recovery can be complex and incomplete

Example: In a DBMS, you can enforce a rule where no two employees have the same employee ID. In a file system, you would have to manually ensure no duplicates exist.

2. Write a short note on database languages

- DDL (Data Definition Language): Used to define, alter, and delete database objects like tables, indexes and views. For example, `CREATE TABLE Students (ID INT, Name VARCHAR(50));` creates a new table named 'students'.
- DML (Data Manipulation Language): Used for data manipulation within tables. Examples include INSERT, UPDATE, DELETE and SELECT.

For instance, INSERT INTO students VALUES (1, 'John'); adds a new record.

- DCL (Data Control Language): Manages permission within the database using commands like GRANT and REVOKE. For example, GRANT SELECT ON students TO user1; allows 'user1' to select data from the 'students' table.

- TCL (Transaction Control Language): Manages transactions to ensure data integrity, using commands such as COMMIT, ROLLBACK, and SAVEPOINT. For instance, COMMIT, saves all changes made in the current transaction.

3. What do you understand by ER diagram? What are the various constraints that can be added to an ER diagram?

- ER diagram (Entity-Relationship Diagram): A graphical representation of entities within a database. It helps in designing the database structure.

- Constraints in ER Diagrams:

  - primary key: Uniquely identifies each entity instance (Eg: Employee ID for an Employee entity)

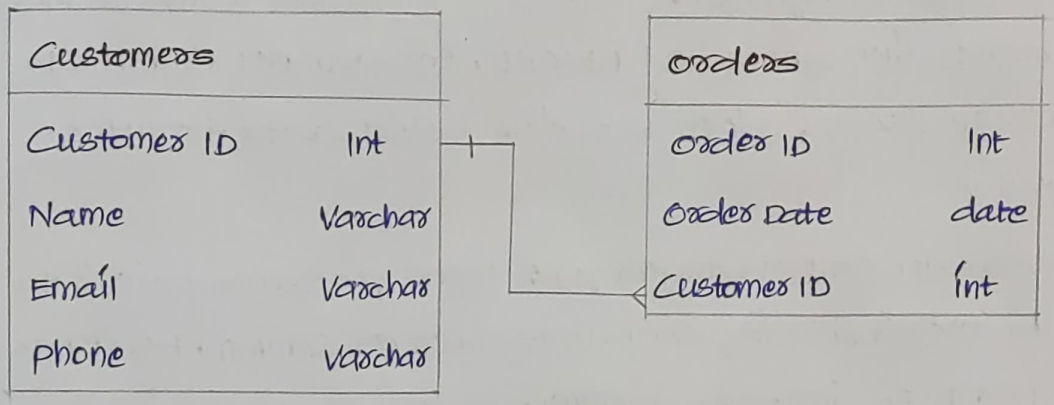
  - Foreign key: Represents relationships between entities, ensuring referential integrity (eg: Employee ID in the 'projects' entity that refers to the 'Employee' entity).

  - Unique: Ensures that certain attributes have unique values (e.g; Social security number)

  - Not Null: Ensures that certain attributes must have a value (e.g; an employee must have a name).

  - Relationship Constraints: Cardinality (one-to-one, one-to-many) and

participation (mandatory or optional)

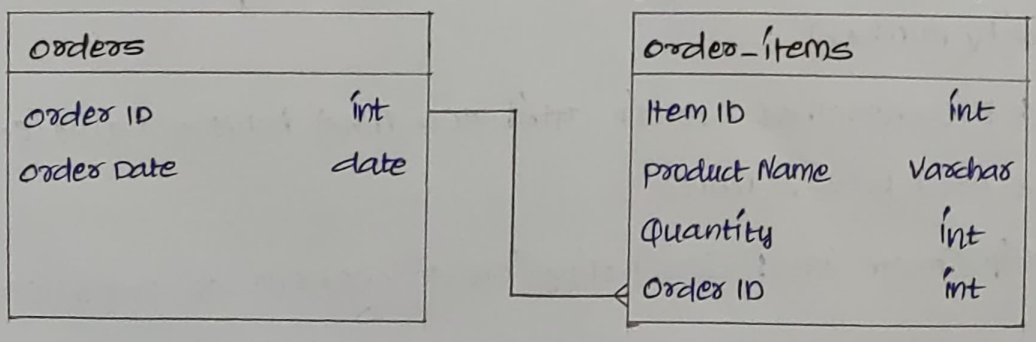


4. What is the importance of weak entity sets with the help of an example?

- Weak Entity sets: These are entities that cannot exist without being associated with another entity. They do not have a primary level of their own and rely on a strong entity for identification.
- Importance: They help model real-world situations where some entities depend on others. For example, in an 'order' and 'order Items' relationship, 'order items (weak entity) depend on 'order' (strong entity) because 'order items' do not uniquely exist without an associated 'Order'.

Example:

- order (strong Entity): order ID (primary key)
- order Items (weak Entity): Item ID (partial key), relies on order ID from 'order' to form a Composite key (order ID, Item ID)



5. Differentiate the following : Relational Model and Relational Algebra

Feature	Relational Model	Relational Algebra
Definition	A theoretical framework for organizing data using tables (relations).	A procedural query language that defines operations on tables.
purpose	To provide a structure and integrity for databases	To perform operations on the data within the tables.
Components	Tables, attributes, keys and relationships.	Operations like SELECT, PROJECT, JOIN, UNION
Example	A table of 'Students' with Colums ID, Name, Age.	SELECT Name FROM Students WHERE Age > 18, retrieves names of students older than 18.

6. What is RDBMS ? Is there any order of tuples ?

- RDBMS (Relational Database Management System) : A type of DBMS that organizes data into related tables using relations. It uses SQL for data manipulation and ensures integrity through constraints and relationships.
- Order of Tuples : In RDBMS, the tuples (rows) do not have a specific order, unlike arrays or lists in programming. Retrieval order is determined by queries, such a susing ORDER By.

EXAMPLE : A 'Customers' table where each row is a Customer record, and retrieval is not inherently ordered unless specified.

7. Explain various operations of relational algebra.

- SELECT (σ) : Filters rows based on a condition.

Example : $\sigma_{Age > 25}$ (Employees) retrieves employees older than 25.

- PROJECT (π) : Selects specific columns from a table.

Example : π name, salary (Employees) retrieves only the 'Name' and 'Salary' columns

- JOIN (⋈) : Combines rows from two or more tables based on a related column.

Example : Employees ⋈ Departments join employees with their respective departments.

- UNION (∪) : Combines results of two questions, removing duplicates.

Example : π name (customers) ∪ π name (suppliers) lists unique names from both Customers and Suppliers.

- INTERSECT (∩) : Returns common rows from two queries

Example : π Name (Customers) ∩ π Name (Suppliers) shows Names that are both Customers and Suppliers.

- DIFFERENCE (-) : Returns rows from one query that are not in another.

Example : π Name (Customers) - π Name (Suppliers) shows names only found in Customers.

8) Differentiate between primary key, Candidate key and Super key.

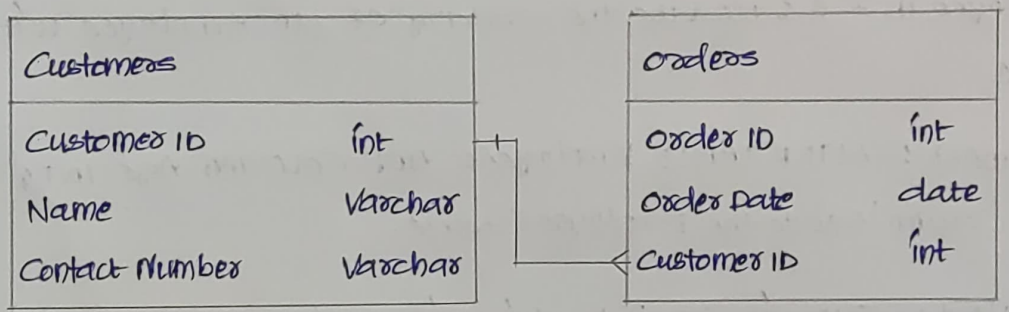
Key Type	Definition	Example
primary key	A unique identifier for each record in a table.	Employee ID in an 'Employees' table
Candidate key	A column, or set of columns, that can uniquely identify records and could be a primary key	Both Employee ID and SSN in 'Employees'

Super key	A set of Columns that includes the primary key and more; Can identify rows uniquely	{Employee ID, Name}, {Employee ID, Dept}.

9. What is meant by Referential integrity? How is it implemented as a foreign key? Illustrate using a real example.

- Referential integrity: Ensures that a foreign key value always refers to an existing primary key value in another table. This maintains consistency across related tables.
- Implementation: A foreign key in one table references a primary key in another table to establish a link.

Example:



- Customers Table:
  - Customer ID (primary key)
- Orders Table:
  - Order ID (primary key), Customer ID (foreign key referencing Customer ID in Customers Table)

This ensures every order references a valid Customer, preventing orphaned records.

10. With the help of an example, Compare DML and DDL.

Type	Example	purpose
DML	INSERT INTO Students VALUES (1, 'John');	Used to manipulate data within tables
DDL	CREATE TABLE Students (ID INT, Name VARCHAR(50));	Used to define and modify database structure

Example Explanation :

- DML Statement : Adds a new record into the 'students' table
- DDL Statement : Creates the structure of an 'Students' table.

11. Write a Sample Statement in DML and one in DDL.

- DML Statement : UPDATE Employees SET Salary = Salary + 500 WHERE Employee ID = 1 ; Update the salary of an employee with Employee ID 1.

- DDL Statement : ALTER TABLE Employees ADD COLOUMN Age INT; Adds a new Column 'Age' to the Employees' table.

12. What is Nested Query in SQL? Explain with the help of an example.

- Nested Query (Subquery) : A query within another SQL query, used to filter results based on the output of the inner query.

Example :

SELECT Name FROM Employees WHERE Department ID = (SELECT Department ID FROM Departments WHERE Department Name = 'Sales');

- Explanation : The inner query finds the Department ID of 'Sales' and the outer query retrieves names of employees in that department.

13. List aggregate functions of SQL.

- COUNT() : Returns the number of rows.

Example : SELECT COUNT(*) FROM orders ; Counts total orders.

- SUM() : Returns the sum of values

Example : SELECT SUM (salary) FROM Employees ; total all salaries

- AVG() : Returns the average value

Example : SELECT AVG (salary) FROM Employees ; Calculates the average salary.

- MIN() : Returns the smallest value

Example : SELECT MIN (salary) FROM Employees ; finds the lowest salary.

- MAX() : Returns the largest value

Example : SELECT MAX (salary) FROM Employees ; finds the highest salary.

14. What are the different types of indexes used in index sequential files?

- primary index : Built on the primary key of a table, providing fast access.

Example : Index on Employee ID in 'Employees' table.

- Secondary Index : Built on non-primary key columns, used for faster retrieval of non-primary data.

Example : Index on Last Name in 'Employees' for quicker searches by name.

- clustering Index : Groups similar data together, based on clustering fields that are not necessarily unique.

Example : Index on 'Department' in 'Employees' where many employees belong to the same department.

15. How is a B-Tree structurally different from a B^+-Tree ?

Feature	B-Tree	B+-Tree
Data Storage	Stores data in internal and leaf nodes.	Stores data only in leaf nodes; internal nodes only store keys.
Traversal	Traversal can stop at any node with data	Traversal must reach leaf nodes for data
Efficiency	Less Efficient for range queries as data is spread.	More efficient for range queries, data is linked in leaf nodes.

Example use case :

- B-Tree : Good for balanced lookups
- B^+-Tree : Better for range searches, such as finding all records between two values.

16. Distinguish between dense index and sparse index and give examples for each.

Feature	Dense Index	Sparse Index
Indexing	Every record has an index entry.	only some records have an index entry (usually the first record of each block).
Storage	Requires more storage space due to more entries.	Requires less storage space as not every record is indexed.
Example	Index on every row of a 'Customers' table	Index on the first row of each block in a 'Customers' table.

17. What are the desirable properties of transactions? Explain

- Atomicity : Ensures all operations in a transaction are completed successfully ; if not, none are.

Example : In a fund transfer, both debit and credit operations must occur ; otherwise, neither does.

- Consistency : Ensures a transaction brings the database from one valid state to another.

Example : Bank account totals should remain consistent after transactions.

- Isolation : Ensures transactions do not interfere with each other.

Example : Two users updating the same record should not see others partial changes.

- Durability : Ensures that once a transaction is committed, changes are permanent, even in case of a system failure.

Example : once a booking is confirmed, it remains even if the system crashes.

18. Discuss the four ACID properties and their importance.

property	Description	Importance
Atomicity	Ensures all operations within a transaction are completed	prevents incomplete operations from affecting data integrity
Consistency	Ensures the database remains in a valid state post-transaction.	Maintains data validity and integrating throughout
Isolation	Ensures transactions do not affect each other	provides accurate data and prevents conflicts.
Durability	Ensures committed transactions remain even after failures	Guarantees that completed transactions are permanent

19. What is TCL ? Explain

• TCL (Transaction Control Language) : Commands that manage transactions in SQL to ensure data integrity and proper management of database states.

• COMMIT : saves changes made during the transaction permanently.

Example : COMMIT; to save all changes in the transaction.

• ROLLBACK : Reverts changes made during the transaction to the last Committed state.

Example : ROLLBACK; to undo all changes if something goes wrong.

• SAVEPOINT : sets a Savepoint within a transaction for partial rollbacks.

Example : SAVEPOINT sp1; allows rollback to this specific point without affecting prior changes.

20. How Control structures like IF-THEN-ELSE, IF-THEN-ELSE IF, CASE, WHILE are implemented using PL/SQL ?

• PL/SQL Control Structures : Allow decision-making and iteration within SQL procedures.

- IF-THEN : Executes code if a condition is true

Example :

```
IF Salary > 5000 THEN
bonus := salary * 0.10;
END IF;
```

- IF-THEN-ELSE : provides an alternative if the condition is false.

Example :

```
IF Salary > 5000 THEN
  bonus := Salary * 0.10;
ELSE
  bonus := salary * 0.05;
END IF;
```

- ELSEIF : checks multiple conditions in sequence

Example :

```
IF Salary > 5000 THEN
bonus := salary * 0.10;
ELSE IF Salary > 3000 THEN
  bonus := Salary * 0.08;
ELSE
  bonus := salary * 0.05;
END IF;
```

- CASE similar to IF-THEN-ELSEIF, used for evaluating a condition against multiple possible values.

Example :

```
CASE grade
WHEN 'A' THEN result = 'Excellent';
WHEN 'B' THEN result = 'Good';
ELSE result := 'Needs Improvement';
 END CASE;
```

- WHILE : Repeats a block of code while a condition is true

Example :

```
WHILE Counter < 10 LOOP
Counter := Counter + 1 ;
END LOOP ;
```

21. Explain backup and restore, why is it needed?

- Backup : Creating a copy of data at a certain point of time to prevent loss. This can be done through full, incremental, or differential backups.
- Restore : The process of retrieving data from a backup to bring the database to a previous state. Used in cases of data loss, corruption, or system failure.
- Importance : Backups protect against unexpected data loss, hardware failures, accidental deletions and cyber attacks. Regular backups ensure that data can be restored to the most recent, accurate state, minimizing downtime and loss of critical information.

Example : A Company performs daily backups of its Customer database. In case of data breach, they can restore to the last clean backup, ensuring minimal data loss and business continuity