SQL & DBMS Notes

Introduction to DBMS

- **DBMS (Database Management System):** Software for creating, storing, and managing databases.
- **Examples:** MySQL, Oracle, PostgreSQL, SQL Server.
- **Benefits:** Data security, consistency, integrity, concurrent access, and recovery.

What is SQL

- **SQL:** Structured Query Language for interacting with relational databases.
- **Used for:** Creating structures, inserting, updating, deleting, and querying data.

MySQL Installation

- Download MySQL from dev.mysql.com
- Install MySQL Server, MySQL Workbench.
- Configure root password, connect via GUI (Workbench).

Sub Languages of SQL

1. **DDL (Data Definition Language):** Used for defining the database structure.
    - CREATE
    - ALTER
    - DROP
2. **DML (Data Manipulation Language):** Used for managing data within the database.
    - INSERT
    - UPDATE
    - DELETE
3. **DQL (Data Query Language):** Used for retrieving data from the database.
    - SELECT
4. **TCL (Transaction Control Language):** Used for managing transactions in the database.
    - COMMIT
    - ROLLBACK
5. **DCL (Data Control Language):** Used for managing permissions and access rights.
    - GRANT
    - REVOKE

Creating a Database

- CREATE DATABASE school;
- USE school;

Data Types in SQL Server

- **Numeric:** INT, FLOAT, DECIMAL
- **String:** CHAR(n), VARCHAR(n)
- **Date/Time:** DATE, DATETIME, TIME
- **Others:** BOOLEAN, TEXT, BLOB

Constraints for Data in Database

- Constraints are rules applied to columns to maintain data integrity.
- Common constraints include

NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, and DEFAULT**.**

- **Example:**

CREATE TABLE Employees ([cite: 91]

EmpID INT PRIMARY KEY, [cite: 93]

Name VARCHAR(50) NOT NULL, [cite: 94]

Age INT CHECK (Age >= 18), [cite: 95]

Email VARCHAR(100) UNIQUE, [cite: 95]

DepartmentID INT, [cite: 96]

FOREIGN KEY (DepartmentID) REFERENCES Departments (DeptID) [cite: 97]

);

Creating a Table

- **Example:**

SQL

CREATE TABLE students ([cite: 28]

student_id INT PRIMARY KEY, [cite: 29]

name VARCHAR(100), [cite: 30]

age INT, [cite: 31]

grade VARCHAR(10) [cite: 32]

);

Adding, Modifying, and Deleting Columns

- **Add:** ALTER TABLE students ADD email VARCHAR(100);

- **Modify:** ALTER TABLE students MODIFY age SMALLINT;
- **Delete:** ALTER TABLE students DROP COLUMN email;

Deleting a Table

- DROP TABLE students;

Inserting Data

- **Example:**

INSERT INTO students (student_id, name, age, grade) [cite: 44]

VALUES (1, 'John', 16, '10th'); [cite: 44]

Updating Data

- **Example:**

UPDATE students SET age = 17 WHERE student_id = 1; [cite: 46]

Deleting Data

- **Example:**

DELETE FROM students WHERE student_id = 1; [cite: 48]

Primary Key

- Unique identifier, not NULL, no duplicates.
- **Example:**

CREATE TABLE teachers ([cite: 52]

teacher_id INT PRIMARY KEY, [cite: 53]

name VARCHAR(100) [cite: 54]

);

Foreign Key

- References primary key of another table.
- **Example:**

CREATE TABLE classes ([cite: 59]

class_id INT PRIMARY KEY, [cite: 60]

teacher_id INT, [cite: 61]

FOREIGN KEY (teacher_id) REFERENCES teachers (teacher_id) [cite: 62]

);

SELECT Query

- SELECT * FROM students;

- SELECT name, age FROM students WHERE age > 15;

WHERE Clause in SQL

- The WHERE clause filters rows based on a condition.
- **Example:**

SELECT * FROM Employees WHERE Age > 30; [cite: 101]

Order By Clause in SQL

- The ORDER BY clause is used to sort the result-set of a SELECT query in ascending or descending order.
- By default, it sorts in ascending order (ASC).
- To sort in descending order, use the DESC keyword.
- **Example:** SELECT Name, Age FROM Employees ORDER BY Age DESC;

Aggregate Functions in SQL

- Used to perform calculations on multiple rows and return a single result.
- Common functions:

COUNT, SUM, AVG, MAX, MIN.

- **Example:**

SQL

SELECT AVG(Age) AS Avg_age FROM Employees; [cite: 106]

Group By Clause in SQL

- The GROUP BY clause groups rows that have the same values in specified columns into a summary row.
- It is often used with aggregate functions (COUNT, SUM, AVG, MAX, MIN) to perform calculations on each group.
- **Example:** SELECT DepartmentID, COUNT(EmpID) AS NumberOfEmployees FROM Employees GROUP BY DepartmentID;

Having Clause in SQL

- The HAVING clause is used to filter groups based on a specified condition.
- It is used after the GROUP BY clause and cannot be used without GROUP BY.
- WHERE filters rows *before* aggregation, while HAVING filters groups *after* aggregation.
- **Example:** SELECT DepartmentID, AVG(Age) AS AvgAge FROM Employees GROUP BY DepartmentID HAVING AVG(Age) > 30;

Rank Function in SQL

- Ranking functions assign a rank to each row within a partition of a result set.

- Common ranking functions include ROW_NUMBER(), RANK(), DENSE_RANK(), and NTILE().
- ROW_NUMBER() assigns a unique sequential integer to each row.
- RANK() assigns a rank with gaps for ties.
- DENSE_RANK() assigns a rank without gaps for ties.
- **Example:** SELECT Name, DepartmentID, Age, RANK() OVER (PARTITION BY DepartmentID ORDER BY Age DESC) AS RankWithinDept FROM Employees;

Joins in SQL

- Used to combine rows from two or more tables based on a related column.
- **Example Tables:** Employees and Departments

Employees Table:

| EmpID | Name | DeptID | |-------|-------|--------| | 1 | Alice | 101 | | 2 | Bob | 102 | | 3 | Carol | 103 |

Departments Table:

| DeptID | DeptName | |--------|----------| | 101 | HR | | 102 | IT | | 104 | Finance |

4.1 INNER JOIN

- Returns rows with matching values in both tables.

SELECT e.Name, d.DeptName [cite: 116]

FROM Employees e [cite: 117]

INNER JOIN Departments d ON e.DeptID = d.DeptID; [cite: 118]

4.2 LEFT JOIN

- Returns all rows from the left table and matched rows from the right table or

NULL for no match.

SELECT e.Name, d.DeptName [cite: 122]

FROM Employees e [cite: 123]

LEFT JOIN Departments d ON e.DeptID = d.DeptID; [cite: 124]

4**.3 RIGHT JOIN**

- Returns all rows from the right table and matched rows from the left table or

NULL for no match.

- **Example:**

SQL

SELECT e.Name, d.DeptName [cite: 128]

FROM Employees e [cite: 129]

RIGHT JOIN Departments d ON e.DeptID = d.DeptID; [cite: 130]

4.4 FULL OUTER JOIN

- Returns all rows when there's a match in one of the tables (returns all rows from both tables, with NULLs for non-matches).
- **Example:**

SELECT e.Name, d.Dept.Name [cite: 134]

FROM Employees e [cite: 135]

FULL OUTER JOIN Departments d ON e.DeptID = d.DeptID; [cite: 136]

4.5 SELF JOIN

- A table is joined with itself.
- **Example:**

SQL

SELECT e.Name AS Employee, m.Name AS Manager [cite: 140]

FROM Employees e [cite: 142]

LEFT JOIN Employees m ON e.ManagerID = m.EmpID; [cite: 142]

Union and Intersection in SQL

- **UNION**: Combines the result-set of two or more SELECT statements.
    - Each SELECT statement must have the same number of columns, similar data types, and columns in the same order.
    - UNION removes duplicate rows by default (UNION ALL includes duplicates).
- **INTERSECT**: Returns only the rows that are common to all SELECT statements. (Note: INTERSECT is not supported in all SQL databases like MySQL).
- **Example (UNION):**

SQL

SELECT Name FROM Employees

UNION

SELECT Name FROM Students;

- **Example (INTERSECT - conceptual):**

SQL

SELECT Name FROM TableA

INTERSECT

SELECT Name FROM TableB;

Subqueries in SQL

- A subquery (or inner query) is a query nested inside another SQL query (outer query).
- Can be used with SELECT, INSERT, UPDATE, and DELETE statements.
- Can return a single value, a single row, a single column, or a table.
- **Example (Scalar Subquery):** SELECT Name, (SELECT DeptName FROM Departments WHERE DeptID = Employees.DepartmentID) AS Department FROM Employees;
- **Example (Subquery with IN):** SELECT Name FROM Employees WHERE DepartmentID IN (SELECT DeptID FROM Departments WHERE DeptName = 'IT');

Co-related Subqueries

- A co-related (or correlated) subquery depends on the outer query for its values and is executed once for each row processed by the outer query.
- Cannot be run independently of the outer query.
- Often used for row-by-row processing.
- **Example:** SELECT e.Name, e.DepartmentID FROM Employees e WHERE e.Salary > (SELECT AVG(Salary) FROM Employees WHERE DepartmentID = e.DepartmentID);

Normalization in SQL

- Normalization is the process of organizing the columns and tables of a relational database to reduce data redundancy and improve data integrity.
- Primary goals are to eliminate redundant data and ensure data dependencies make sense.
- Involves breaking down a large table into smaller, more manageable tables and defining relationships between them.

2NF (Second Normal Form)

- A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.
- No non-key attribute is dependent on only a part of a composite primary key.
- To achieve 2NF, remove subsets of data that apply to multiple rows of a table and place them in a new table.

3NF (Third Normal Form)

- A table is in 3NF if it is in 2NF and there are no transitive dependencies.
- A transitive dependency exists when a non-key attribute is dependent on another non-key attribute.
- To achieve 3NF, remove columns that are not directly dependent on the primary key.

BCNF (Boyce-Codd Normal Form)

- BCNF is a stricter form of 3NF.

- A table is in BCNF if and only if for every non-trivial functional dependency X -> Y, X is a superkey.
- Addresses some anomalies that 3NF might miss, especially with multiple overlapping candidate keys.

Views in SQL

- A view is a virtual table based on the result-set of a SQL query.
- It does not store data itself; data is derived from base tables.
- **Benefits:** Simplifies complex queries, enhances security, and provides data independence.
- **Example:** CREATE VIEW EmployeeHR AS SELECT EmpID, Name, Email FROM Employees WHERE DepartmentID = (SELECT DeptID FROM Departments WHERE DeptName = 'HR');

Stored Procedures in SQL

- A stored procedure is a prepared SQL code that you can save and reuse.
- It is a set of SQL statements that perform a specific task.
- **Benefits:** Improved performance, reduced network traffic, better security, and reusability.
- **Example:**

```
DELIMITER //

CREATE PROCEDURE GetEmployeesByDept (IN deptName VARCHAR(50))

BEGIN

   SELECT E.Name, D.DeptName

   FROM Employees E

   JOIN Departments D ON E.DepartmentID = D.DeptID

   WHERE D.DeptName = deptName;

END //

DELIMITER ;

CALL GetEmployeesByDept('IT');
```

Indexes in SQL

- An index is a special lookup table that the database search engine can use to speed up data retrieval.
- Works similarly to a book index.
- Can be created on one or more columns.
- **Types:** Clustered (determines physical order, one per table) and Non-clustered (separate structure, multiple per table).
- **Benefits:** Faster query performance for SELECT statements.

- **Drawbacks:** Slower INSERT, UPDATE, and DELETE operations, consumes disk space.
- **Example:** CREATE INDEX idx_employee_name ON Employees (Name);

Practice Problems

Practice Problem - 1

- **Problem:** Retrieve the names of all employees who work in the 'HR' department.
- **Solution Approach:** Join Employees and Departments tables and filter by DeptName.
- **SQL Query:**

SELECT E.Name

FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DeptID

WHERE D.DeptName = 'HR';

Practice Problem - 2

- **Problem:** Find the total number of employees in each department.
- **Solution Approach:** Use GROUP BY on DepartmentID and COUNT aggregate function.
- **SQL Query:**

SQL

SELECT DepartmentID, COUNT(EmpID) AS TotalEmployees

FROM Employees

GROUP BY DepartmentID;

Practice Problem - 3

- **Problem:** List departments that have more than 2 employees.
- **Solution Approach:** Use GROUP BY and HAVING clause with COUNT.
- **SQL Query:**

SQL

SELECT DepartmentID, COUNT(EmpID) AS NumberOfEmployees

FROM Employees

GROUP BY DepartmentID

HAVING COUNT(EmpID) > 2;

Practice Problem - 4

- **Problem:** Get the names of employees and their respective department names, including employees who might not have a department assigned, and departments that might not have employees.
- **Solution Approach:** Use a FULL OUTER JOIN.
- **SQL Query:**

SQL

```
SELECT E.Name, D.DeptName
FROM Employees E
FULL OUTER JOIN Departments D ON E.DepartmentID = D.DeptID;
```

Practice Problem - 5

- **Problem:** Find the employee(s) who are the oldest in each department.
- **Solution Approach:** Use a subquery or a ranking function (RANK() or DENSE_RANK()) partitioned by department.
- **SQL Query (using DENSE_RANK):**

SQL

```
WITH RankedEmployees AS (
  SELECT
    Name,
    DepartmentID,
    Age,
    DENSE_RANK() OVER (PARTITION BY DepartmentID ORDER BY Age DESC) as rnk
  FROM Employees
)
SELECT Name, DepartmentID, Age
FROM RankedEmployees
WHERE rnk = 1;
```

Interview Questions and Answers

- **Q: What is DBMS?**
    - **A:** DBMS is software for managing data in a structured format, allowing operations like insert, update, and query.
- **Q: What is SQL?**

    - **A:** SQL is the language used to interact with relational databases using commands like SELECT, INSERT, UPDATE.

- **Q: Types of SQL Commands**
    - **A:** DDL, DML, DQL, TCL, DCL.
- **Q: Difference between DELETE, TRUNCATE, DROP**
    - **A:** DELETE: removes rows (with WHERE), TRUNCATE: removes all rows (faster), DROP: deletes the table.
- **Q: What is a Primary Key?**
    - **A:** A unique column that identifies rows in a table.
- **Q: What is a Foreign Key?**
    - **A:** A column that creates a relationship with another table's primary key.
- **Q: How to modify a column?**
    - **A:** ALTER TABLE students MODIFY age INT;
- **Q: Safe update mode**
    - **A:** Prevents updates/deletes without key-based WHERE clause. Can disable via

SET SQL_SAFE_UPDATES = 0;

- **Q: WHERE vs HAVING**
    - **A:** WHERE filters rows before aggregation.

HAVING filters after aggregation.

- **Q: What is normalization?**
    - **A:** Process of organizing data to reduce redundancy and improve integrity.
- **Q: What is the purpose of constraints in SQL?**
    - **A:** They ensure data integrity by enforcing rules on table columns (e.g., NOT NULL, UNIQUE).
- **Q: How does the WHERE clause work?**
    - **A:** It filters rows based on a specified condition before they are returned.
- **Q: What is an aggregate function?**
    - **A:** It performs a calculation on a set of values and returns a single result (e.g., COUNT, SUM).
- **Q: What is the difference between INNER JOIN and LEFT JOIN?**
    - **A:** INNER JOIN returns only matching rows;

LEFT JOIN includes all rows from the left table.

- **Q: When do you use a SELF JOIN?**
    - **A:** To relate rows within the same table, such as showing employees and their managers.