SQL : Comprehensive Course Notes

Table of Contents

1. Introduction to SQL

SQL (Structured Query Language) is a standardized language used for managing and manipulating relational databases. It allows you to create, retrieve, update, and delete data in a structured manner.

Key points:

- SQL is used with relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, SQL Server, etc.
- It's declarative, meaning you specify what you want, not how to get it.
- SQL is divided into several sub-languages: DDL, DML, DCL, and TCL.

2. Basic SQL Commands

The most fundamental SQL commands include:

- SELECT: Retrieve data from one or more tables
- INSERT: Add new data into a table
- UPDATE: Modify existing data in a table
- DELETE: Remove data from a table
- CREATE: Create new database objects (tables, views, etc.)
- ALTER: Modify the structure of database objects
- DROP: Delete database objects

Example:

```
SELECT * FROM employees;
INSERT INTO employees (name, position) VALUES ('John Doe',
'Manager');
UPDATE employees SET salary = 50000 WHERE id = 1;
DELETE FROM employees WHERE id = 2;
```

3. Data Types

SQL supports various data types to store different kinds of information:

- Numeric types: INT, BIGINT, FLOAT, DECIMAL
- String types: CHAR, VARCHAR, TEXT
- Date and time types: DATE, TIME, DATETIME, TIMESTAMP
- Boolean type: BOOLEAN

- Binary types: BLOB, BINARY

Example:

```
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2),
    description TEXT,
    created_at TIMESTAMP
);
```

4. Creating and Modifying Tables

Use DDL (Data Definition Language) commands to create and modify database structures:

- CREATE TABLE: Create a new table
- ALTER TABLE: Modify an existing table's structure
- DROP TABLE: Delete a table

Example:

```
CREATE TABLE customers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);

ALTER TABLE customers ADD COLUMN phone VARCHAR(20);

DROP TABLE old_customers;
```

5. Querying Data

The SELECT statement is used to query data from tables:

```
SELECT column1, column2 FROM table_name WHERE condition;
```

Key clauses:

- FROM: Specifies the table(s) to query
- WHERE: Filters rows based on a condition
- ORDER BY: Sorts the result set
- LIMIT: Restricts the number of rows returned

Example:

```
SELECT name, email FROM customers WHERE city = 'New York' O
RDER BY name LIMIT 10;
```

6. Filtering and Sorting Data

Use various operators and clauses to filter and sort data:

- Comparison operators: =, <>, <, >, <=, >=
- Logical operators: AND, OR, NOT
- LIKE: Pattern matching with wildcards (% and _)
- IN: Check if a value matches any value in a list
- BETWEEN: Check if a value is within a range
- IS NULL / IS NOT NULL: Check for null values

Example:

```
SELECT * FROM products
WHERE category = 'Electronics'
  AND price BETWEEN 100 AND 500
  AND name LIKE '%phone%'
ORDER BY price DESC;
```

7. Joins

Joins are used to combine rows from two or more tables based on a related column between them:

- INNER JOIN: Returns records that have matching values in both tables
- LEFT JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
- FULL OUTER JOIN: Returns all records when there is a match in either left or right table

Example:

```
SELECT orders.id, customers.name, orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.id;
```

8. Aggregation and Grouping

Aggregate functions perform calculations on a set of values and return a single result:

- COUNT(): Counts the number of rows
- SUM(): Calculates the sum of a set of values
- AVG(): Calculates the average of a set of values
- MAX(): Returns the maximum value
- MIN(): Returns the minimum value

The GROUP BY clause is used with aggregate functions to group the result-set by one or more columns.

Example:

```
SELECT category, COUNT(*) as product_count, AVG(price) as a
vg_price
FROM products
```

```
GROUP BY category
HAVING COUNT(*) > 10;
```

9. Subqueries

A subquery is a query nested inside another query. It can be used in various parts of an SQL statement:

- In the WHERE clause
- In the FROM clause (derived tables)
- In the SELECT clause (scalar subqueries)

Example:

```
SELECT name, price
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

10. Views

A view is a virtual table based on the result-set of an SQL statement:

```
CREATE VIEW high_value_customers AS
SELECT id, name, email
FROM customers
WHERE lifetime_value > 10000;
```

Views can simplify complex queries and provide an additional layer of security.

11. Indexes

Indexes are used to speed up data retrieval operations on database tables:

```
CREATE INDEX idx_last_name ON employees(last_name);
```

Types of indexes:

- Single-column indexes
- Multi-column indexes

- Unique indexes
- Full-text indexes

12. Transactions

Transactions ensure the integrity of data in a database. They have four key properties (ACID):

- Atomicity: All operations in a transaction succeed or they all fail
- Consistency: The database remains in a consistent state before and after the transaction
- Isolation: Concurrent transactions do not interfere with each other
- Durability: Once a transaction is committed, it remains so

Example:

```
BEGIN TRANSACTION;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
COMMIT;
```

13. Stored Procedures and Functions

Stored procedures and functions are SQL code that can be saved and reused:

```
DELIMITER //
CREATE PROCEDURE get_employee_by_id(IN emp_id INT)
BEGIN
    SELECT * FROM employees WHERE id = emp_id;
END //
DELIMITER ;

CALL get_employee_by_id(123);
```

14. Triggers

Triggers are SQL procedures that are automatically executed in response to certain events on a particular table:

```
CREATE TRIGGER update_modified_date
BEFORE UPDATE ON products
FOR EACH ROW
SET NEW.modified_date = CURRENT_TIMESTAMP;
```

15. Database Design and Normalization

Database design principles:

- Identify entities and relationships
- Define primary and foreign keys
- Apply normalization rules (1NF, 2NF, 3NF, BCNF)

Normalization reduces data redundancy and improves data integrity.

16. Performance Tuning

Techniques to improve SQL performance:

- Proper indexing
- Query optimization
- Avoiding correlated subqueries
- Using EXPLAIN to analyze query execution plans
- Partitioning large tables
- Caching frequently accessed data

17. Security and Access Control

SQL provides commands to manage database security:

- CREATE USER: Create a new database user
- GRANT: Give specific privileges to a user

- REVOKE: Remove specific privileges from a user

Example:

```
CREATE USER 'john'@'localhost' IDENTIFIED BY 'password123';
GRANT SELECT, INSERT ON database_name.* TO 'john'@'localhos
t';
```

18. Advanced SQL Concepts

Some advanced SQL topics include:

- Window functions (ROW_NUMBER, RANK, etc.)
- Common Table Expressions (CTEs)
- Pivot and unpivot operations
- Full-text search
- JSON and XML data handling
- Recursive queries

Example of a window function:

```
SELECT
    name,
    department,
    salary,
    ROW_NUMBER() OVER (PARTITION BY department ORDER BY sal
ary DESC) as salary_rank
FROM employees;
```

This comprehensive guide covers the main aspects of SQL from basic to advanced concepts. Remember that practice and hands-on experience are crucial for mastering SQL. Good luck with your SQL course!