September 19, 2025

SQL Topics Overview

Introduction to Database and DBMS

What is a Database?

A database is an organized collection of data that can be easily accessed, managed, and updated. It stores information in a structured format, enabling efficient retrieval and manipulation.

Introduction to DBMS

A Database Management System (DBMS) is software that interacts with databases. It helps users create, read, update, and delete data while managing database structure and access.

Key Characteristics of DBMS

DBMS provides data independence, efficient data access, data integrity, security, concurrency control, and backup/recovery mechanisms to ensure reliable and secure data management.

DBMS Models and Relational DBMS

Types of DBMS Models

- Hierarchical Model organizes data in a tree-like structure with parent-child relationships.
- Network Model uses graph structures allowing many-to-many relationships between entities.
- Relational Model stores data in tables (relations) with rows and columns, supporting SQL.
- Object-Oriented Model integrates database capabilities with object programming language features.
- Each model suits different application needs based on complexity and data relationships.

Relational DBMS Highlights

- Data is stored in tables with unique keys ensuring entity integrity.
- Supports powerful SQL queries for data manipulation and retrieval.
- Enforces data integrity through primary, foreign keys, and constraints.
- Highly flexible, scalable, and widely supported by major database vendors.
- Examples include Oracle, MySQL, SQL Server, and PostgreSQL.

Data Integrity and Security in Databases

Data Integrity

- Ensures accuracy, consistency, and reliability of data over its lifecycle.
- Includes constraints like primary keys, foreign keys, and unique constraints.
- Prevents data corruption and enforces valid data entry rules.
- Supports dependable decision-making and reporting.

Database Security

- Protects data from unauthorized access and malicious threats.
- Involves authentication, authorization, and encryption techniques.
- Implements roles and privileges to control user access.
- Audits and monitors database activities for suspicious behavior.

Normalization & Codd's Rules

Normalization Explained

Normalization is a systematic process of organizing data to minimize redundancy and dependency by dividing tables and defining relationships. It improves data integrity and efficiency in database design.

Codd's 12 Rules Overview

Dr. E.F. Codd defined 12 rules to ensure a database is truly relational and fully functional. These include guaranteed access, systematic treatment of nulls, data independence, and comprehensive integrity rules.

Importance for Functional Systems

Adhering to normalization and Codd's rules ensures the database supports consistent data handling, reliable query results, and scalability, making it robust for complex applications and enterprise systems.

Normal Forms: 1NF, 2NF, 3NF

First Normal Form (1NF)	1NF requires tables to have only atomic values with no repeating groups. Each column must contain unique, indivisible values. Example: One phone number per row, not multiple numbers in one field.
Second Normal Form (2NF)	2NF requires all non-key attributes to depend fully on the entire primary key, avoiding partial dependency. Example: In a composite key table, course details must depend on both StudentID & CourseID.
Third Normal Form (3NF)	3NF eliminates transitive dependency, ensuring non-key attributes depend only on the primary key. Example: AdvisorOffice depends on AdvisorName, so move it to a separate table or link directly to StudentID.

Introduction to SQL and SQL*Plus

What is SQL?	SQL (Structured Query Language) is the standard programming language used to manage and manipulate relational databases. It allows users to create, read, update, and delete data efficiently.
Overview of SQL*Plus	SQL*Plus is an Oracle command-line interface tool that allows users to execute SQL commands, scripts, and PL/SQL blocks interactively or in batch mode for database management.
Interacting with SQL*Plus	Users can connect to the Oracle database using SQL*Plus, input SQL statements, view query results, and execute scripts. It supports command history, formatting, and scripting capabilities.

SQL Basics and Statement Rules

SQL Statement Rules	SQL statements must end with a semicolon (;). Keywords are case-insensitive but typically written in uppercase. Statements must follow proper syntax to avoid errors.
Standard SQL Statement Groups	SQL is divided into groups: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language), and TCL (Transaction Control Language), each serving distinct functions.
Basic Data Types	Common SQL data types include VARCHAR2 for text, NUMBER for numeric values, DATE for date/time, and CLOB for large text. Oracle SQL supports additional specialized types.
Naming Rules for SQL Statements	Table and column names must start with a letter, avoid reserved keywords, and be unique within a schema. Use underscores (_) instead of spaces to improve readability.

Tables and Integrity Constraints

Rules for Naming Tables

- Use meaningful, descriptive names reflecting table content or purpose.
- Avoid using spaces and special characters; prefer underscores (_) if needed.
- Start table names with a letter, not a number or special symbol.
- Follow consistent naming conventions (e.g., singular nouns like 'Employee').
- Keep names concise but clear for easy understanding and maintenance.

Specifying Integrity Constraints

- Primary Key: Uniquely identifies each row; must be unique and not null.
- Foreign Key: Maintains referential integrity by linking to primary keys in other tables.
- Unique Constraint: Ensures all values in a column or group of columns are unique.
- Check Constraint: Validates data based on a specified condition or rule.
- Not Null Constraint: Ensures a column cannot have null values, enforcing required data.

DDL Statements and Table Types

DDL Statements Overview	Data Definition Language (DDL) includes commands like CREATE, ALTER, DROP, and TRUNCATE used to define and modify database structures such as tables and indexes.
Core DDL Commands	CREATE builds new tables or objects; ALTER modifies existing ones; DROP deletes tables or objects permanently; TRUNCATE quickly removes all rows from a table without logging individual row deletions.
Regular vs Temporary Tables	Regular tables store persistent data in the database. Temporary tables hold data temporarily during a session or transaction and are automatically dropped afterward.

Data Manipulation Language (DML)

Inserting Rows

The INSERT statement adds new rows to a table. In Oracle SQL, you specify the table and columns, then provide values. Example: INSERT INTO employees (id, name, salary) VALUES (101, 'John Doe', 50000);

Deleting Rows

The DELETE statement removes rows from a table based on a condition. Use WHERE clause to specify which rows to delete. Example: DELETE FROM employees WHERE id = 101;

Updating Rows

The UPDATE statement modifies existing rows. Specify the table, columns to update, new values, and a condition. Example: UPDATE employees SET salary = 55000 WHERE id = 101;

Data Control Language (DCL)

Grant Statement: Assigning Permissions

- Used to give users specific privileges on database objects.
- Privileges include SELECT, INSERT, UPDATE, DELETE, EXECUTE, and more.
- Syntax example: GRANT SELECT ON employees TO user1;
- Enables controlled access to tables, views, and procedures.

Revoke Statement: Removing Permissions

- Used to remove previously granted privileges from users.
- Syntax example: REVOKE SELECT ON employees FROM user1;
- Helps maintain security by limiting user capabilities as needed.
- Prevents unauthorized data manipulation and access.

Database Objects Overview

Index	An index improves query performance by allowing faster data retrieval through a sorted structure of key values.
Synonym	A synonym is an alias for a database object, simplifying access by allowing users to reference objects with alternative names.
Sequence and View	Sequences generate unique numeric values for keys; Views present virtual tables representing results of stored queries for simplified data access.

Data Query Language: Select Statement

SELECT Statement Basics	The SELECT statement is used to retrieve data from one or more tables. Syntax: SELECT column1, column2 FROM table_name; Use * to select all columns.
DISTINCT Clause	DISTINCT removes duplicate rows from the result set, returning only unique records. Example: SELECT DISTINCT city FROM customers; retrieves unique cities.
SQL Operators	SQL operators like comparison (=, <, >), arithmetic (+, -, *), and logical (AND, OR, NOT) enable filtering and manipulation of data within queries.

ORDER BY Clause and Tips

- The ORDER BY clause sorts query results by one or more columns in ascending (ASC) or descending (DESC) order.
- Default sorting is ascending if ASC or DESC is not specified explicitly.
- Multiple columns can be used to sort data hierarchically, e.g., ORDER BY department ASC, salary DESC.
- Use ORDER BY with SELECT statements to present data in a meaningful sequence for reports and analysis.
- Be mindful that sorting large datasets can impact query performance; use indexes to optimize ORDER BY operations.

Aggregate Functions and Grouping

Aggregate Functions

Functions like COUNT, SUM, AVG, MIN, and MAX perform calculations on multiple rows to return a single summarized value.

GROUP BY Clause

Used to group rows sharing the same values in specified columns, enabling aggregate functions on each group separately.

HAVING Clause

Filters groups created by GROUP BY based on aggregate conditions, similar to WHERE but for grouped data.

ROLLUP Operation

Extends GROUP BY to produce subtotals and grand totals for hierarchical levels of data aggregation.

CUBE Operation

Generates all possible combinations of subtotals across multiple dimensions, providing comprehensive multi-dimensional analysis.

Oracle SQL Tips

Use aliases for aggregated columns and combine ROLLUP or CUBE with ORDER BY for organized results. Example: SELECT dept, SUM(salary) FROM emp GROUP BY ROLLUP(dept);

Transactions and Commands

Transaction Definition	A transaction is a sequence of SQL operations executed as a single logical unit to ensure data integrity and consistency in the database.
Commit Command	The COMMIT command saves all changes made during the current transaction permanently to the database, making them visible to other users.
Rollback and Savepoints	ROLLBACK undoes changes made in the current transaction, while SAVEPOINTS allow partial rollback to defined points without aborting the entire transaction.
Transaction Properties (ACID)	Transactions follow ACID properties: Atomicity, Consistency, Isolation, and Durability to guarantee reliable database operations.

Joins and Subqueries

Inner/Equi Join

Combines rows from two tables where the join condition matches. Equi Join is a type of Inner Join using equality (=) operator, returning only matching rows.

Outer Join

Includes matched rows plus unmatched rows from one or both tables. Types: Left Outer Join (all from left), Right Outer Join (all from right), Full Outer Join (all from both).

Self Join

A table joins with itself to compare rows within the same table, often using table aliases to differentiate instances in the query.

Subqueries

A query nested inside another SQL query, used to return data that will be used in the outer query, can be in SELECT, WHERE, or FROM clauses.

Correlated Subqueries

A type of subquery that references columns from the outer query, executed row-by-row, used for complex filtering and comparisons.

Oracle SQL Example

Example: Find employees who earn more than the average salary in their department.

```
SELECT e.emp_name FROM employees e WHERE
e.salary > (SELECT AVG(salary) FROM employees
WHERE dept_id = e.dept_id);
```

Set Operations in SQL

UNION Operator

Combines the results of two SELECT queries and returns distinct rows from both. Example: SELECT column_name FROM table1 UNION SELECT column_name FROM table2; This eliminates duplicates by default.

INTERSECT Operator

Returns only the rows that appear in both SELECT query results. Example: SELECT column_name FROM table1 INTERSECT SELECT column_name FROM table2; Useful for finding common data between tables.

MINUS Operator

Returns rows from the first SELECT query that do not appear in the second. Example: SELECT column_name FROM table1 MINUS SELECT column_name FROM table2; Helps identify differences between datasets.

SQL Tips and Tricks with Oracle Examples

- Use bind variables to improve performance and prevent SQL injection: e.g., SELECT * FROM employees WHERE department_id = :dept_id;
- Leverage Oracle's hierarchical queries using CONNECT BY for tree-structured data retrieval: e.g., SELECT employee_id, manager_id FROM employees CONNECT BY PRIOR employee_id = manager_id;
- Optimize queries with hints like /*+ INDEX(table_name index_name) */ to guide the optimizer.
- Employ MERGE statement for efficient UPSERT operations combining INSERT and UPDATE.
- Utilize ROLLUP and CUBE for advanced aggregate reporting: e.g., SELECT department_id, job_id, SUM(salary) FROM employees GROUP BY ROLLUP(department_id, job_id);

Thank you