Database : Oracle SQL

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

1. Introduction to Database

What is a Database?

A database is an organized collection of structured information or data, typically stored electronically in a computer system. Think of it as a digital filing cabinet where you can store, organize, and retrieve information efficiently.

Key Characteristics:

- **Persistent Storage**: Data remains even after application closes
- **Shared Access**: Multiple users can access simultaneously
- **Data Integrity**: Maintains accuracy and consistency
- **Security**: Controls who can access what data

2. Different Types of Databases

1. Relational Databases (RDBMS)

- Organize data into tables with rows and columns
- Examples: Oracle, MySQL, PostgreSQL, SQL Server

2. NoSQL Databases

- **Document Databases**: Store data in JSON-like documents (MongoDB)
- **Key-Value Stores**: Simple key-value pairs (Redis)
- **Column-family Stores**: Optimized for queries over large datasets (Cassandra)
- **Graph Databases**: Store entities and relationships (Neo4j)

3. Examples of Popular Databases

Relational:

- **Oracle Database** - Enterprise-level, robust
- **MySQL** - Open-source, widely used
- **PostgreSQL** - Advanced open-source
- **Microsoft SQL Server** - Windows ecosystem

NoSQL:

- **MongoDB** - Document database
- **Cassandra** - Distributed column-store
- **Redis** - In-memory key-value store

4. What is RDBMS?

RDBMS = Relational Database Management System

Key Features:

- Data stored in tables (relations)
- Tables connected through relationships
- Uses SQL (Structured Query Language)
- Supports ACID properties
- Enforces data integrity

5. RDBMS Concepts

Tables

```
-- Example table structure
EMPLOYEES table:
+--------+----------+-----------+---------+
| EMP_ID | NAME     | DEPARTMENT| SALARY  |
+--------+----------+-----------+---------+
| 101    | John     | IT        | 50000   |
| 102    | Jane     | HR        | 45000   |
+--------+----------+-----------+---------+
```

Rows (Records)

- Horizontal entries in a table
- Each row represents a complete set of information

Keys

- **Primary Key**: Unique identifier for each row (EMP_ID)
- **Foreign Key**: Links to primary key in another table
- **Composite Key**: Combination of multiple columns as primary key

6. Data Types, Objects, Constraints

Common Data Types in Oracle:

```
-- Numeric
NUMBER, INTEGER, FLOAT

-- Character
VARCHAR2(50), CHAR(10)

-- Date/Time
DATE, TIMESTAMP

-- Binary
BLOB, RAW
```

Database Objects:

- **Tables**: Store data
- **Views**: Virtual tables
- **Indexes**: Improve query performance
- **Sequences**: Generate unique numbers
- **Synonyms**: Alternative names for objects

Constraints:

```
-- Types of constraints
NOT NULL          -- Column cannot contain NULL
UNIQUE            -- All values must be unique
PRIMARY KEY       -- Uniquely identifies each row
FOREIGN KEY       -- Links to primary key in another table
CHECK             -- Validates data against condition
DEFAULT           -- Sets default value
```

7. How to Handle NULLs

NULL represents:

- Missing information
- Unknown value
- Not applicable data

Handling NULLs:

```sql
-- Check for NULL
SELECT * FROM employees WHERE department IS NULL;

-- Check for NOT NULL
SELECT * FROM employees WHERE department IS NOT NULL;

-- Using NVL (replace NULL with value)
SELECT name, NVL(commission, 0) FROM employees;

-- Using COALESCE (returns first non-NULL value)
SELECT name, COALESCE(commission, bonus, 0) FROM employees;

-- NULL in calculations
SELECT salary + NVL(commission, 0) as total_comp FROM employees;
```

8. ACID Properties

Atomicity

- Transactions are all-or-nothing
- Either all operations complete or none do

Consistency

- Database moves from one valid state to another
- All constraints are maintained

Isolation

- Concurrent transactions don't interfere
- Each transaction appears to run alone

Durability

- Once committed, changes are permanent
- Survives system failures

9. Working with DDL (Data Definition Language)

CREATE Command

```
-- Create a table
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    department VARCHAR2(50),
    salary NUMBER(10,2),
    hire_date DATE DEFAULT SYSDATE
);

-- Create table with constraints
CREATE TABLE departments (
    dept_id NUMBER PRIMARY KEY,
    dept_name VARCHAR2(50) UNIQUE NOT NULL
);
```

```sql
-- Create table with foreign key
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    dept_id NUMBER,
    CONSTRAINT fk_dept FOREIGN KEY (dept_id)
    REFERENCES departments(dept_id)
);
```

ALTER Command

```sql
-- Add a column
ALTER TABLE employees ADD email VARCHAR2(100);

-- Modify a column
ALTER TABLE employees MODIFY email VARCHAR2(150);

-- Drop a column
ALTER TABLE employees DROP COLUMN email;

-- Add constraint
ALTER TABLE employees ADD CONSTRAINT uk_email UNIQUE(email);

-- Drop constraint
ALTER TABLE employees DROP CONSTRAINT uk_email;

-- Rename table
ALTER TABLE employees RENAME TO staff;
```

DROP Command

```
-- Drop table
DROP TABLE employees;

-- Drop table with check
DROP TABLE employees CASCADE CONSTRAINTS;
```

10. Examples on DDL

Complete Example:

```
-- Create departments table
CREATE TABLE departments (
    dept_id NUMBER PRIMARY KEY,
    dept_name VARCHAR2(50) NOT NULL,
    location VARCHAR2(100)
);
```

```sql
-- Create employees table with relationships
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE,
    phone VARCHAR2(20),
    hire_date DATE DEFAULT SYSDATE,
    salary NUMBER(10,2) CHECK (salary > 0),
    dept_id NUMBER,
    manager_id NUMBER,
    CONSTRAINT fk_dept FOREIGN KEY (dept_id)
        REFERENCES departments(dept_id)
);
```

11. Working with DML (Data Manipulation Language)

INSERT Command

```
-- Basic insert
INSERT INTO employees (emp_id, name, department, salary)
VALUES (1, 'John Doe', 'IT', 50000);

-- Insert with sequence
INSERT INTO employees (emp_id, name, department)
VALUES (2, 'Jane Smith', 'HR');

-- Insert multiple rows
INSERT ALL
    INTO employees VALUES (3, 'Bob Wilson', 'Finance', 60000)
    INTO employees VALUES (4, 'Alice Brown', 'IT', 55000)
```

UPDATE Command

```sql
-- Update single row
UPDATE employees
SET salary = 55000
WHERE emp_id = 1;

-- Update multiple columns
UPDATE employees
SET salary = salary * 1.1,
    department = 'Senior IT'
WHERE department = 'IT';

-- Update with subquery
UPDATE employees
SET salary = (
    SELECT AVG(salary)
    FROM employees
    WHERE department = 'IT'
)  WHERE emp_id = 2;
```

DELETE Command

```sql
-- Delete specific rows
DELETE FROM employees
WHERE department = 'HR';

-- Delete all rows
DELETE FROM employees;

-- Delete with subquery
DELETE FROM employees
WHERE salary < (
    SELECT AVG(salary)
    FROM employees
);
```

12. Examples on DML

Complete DML Example:

```
-- Insert sample data
INSERT INTO departments VALUES (10, 'IT', 'New York');
INSERT INTO departments VALUES (20, 'HR', 'Boston');
INSERT INTO departments VALUES (30, 'Finance', 'Chicago');

INSERT INTO employees VALUES (
    100, 'John', 'Doe', 'john.doe@company.com',
    '123-456-7890', DATE '2020-01-15', 75000, 10, NULL
);

INSERT INTO employees VALUES (
    101, 'Jane', 'Smith', 'jane.smith@company.com',
    '123-456-7891', DATE '2019-03-20', 65000, 20, 100
);
```

```sql
-- Update operations
UPDATE employees
SET salary = salary * 1.05
WHERE department = 'IT';

-- Delete operation
DELETE FROM employees
WHERE hire_date < DATE '2020-01-01';

-- Commit changes
COMMIT;
```

13. Working with DQL (Data Query Language)

SELECT Command

```sql
-- Select all columns
SELECT * FROM employees;

-- Select specific columns
SELECT emp_id, name, salary FROM employees;

-- Select with calculations
SELECT name, salary, salary * 12 as annual_salary
FROM employees;

-- Using DISTINCT
SELECT DISTINCT department FROM employees;
```

WHERE Clause

```sql
-- Basic conditions
SELECT * FROM employees WHERE salary > 50000;

-- Multiple conditions
SELECT * FROM employees
WHERE department = 'IT' AND salary > 60000;

-- Pattern matching with LIKE
SELECT * FROM employees
WHERE name LIKE 'J%';  -- Names starting with J

SELECT * FROM employees
WHERE name LIKE '%son%';  -- Names containing 'son'

-- IN operator
SELECT * FROM employees WHERE department IN ('IT', 'HR');

-- BETWEEN operator
SELECT * FROM employees WHERE salary BETWEEN 40000 AND 70000;
```

ORDER BY Clause

```sql
-- Single column sort
SELECT * FROM employees ORDER BY name;

-- Multiple column sort
SELECT * FROM employees
ORDER BY department, salary DESC;

-- Sort by column position
SELECT name, department, salary
FROM employees
ORDER BY 2, 3 DESC;
```

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com