PostgreSQL

PostgreSQL is an free open-source database system that supports both relational (SQL) and non-relational (JSON) queries. PostgreSQL is a back-end database for dynamic websites and web applications.
PostgreSQL is a powerful open-source database system known for its reliability, performance, and advanced features. It supports a wide range of data types, ACID compliance, robust security, extensibility, high availability, full-text search, and more.

PostgreSQL Architecture

PostgreSQL is a relational database management system (RDBMS) with a client-server architecture. It includes the following components:
• Client: The user or application that connects to the PostgreSQL database. It sends queries to the server.
• Server: The core component that processes and executes queries. It manages the database instances, including the following:
• Postmaster: The main PostgreSQL process that listens for client connections and launches the backend server processes.
• Backend processes: These processes handle the execution of queries. Each client connection is served by a separate backend process.
• Shared memory: Used for communication between backend processes.
• Database files: Where the data is stored, including the transaction log, data files, etc.

PostgreSQL Data Types

PostgreSQL supports several data types, including:

1. Integer Types:

• integer: Stores a 4-byte integer (e.g., 42).
• bigint: Stores an 8-byte integer (e.g., 10000000000).

2. Character Types:

• varchar(n): Stores a variable-length string (e.g., 'Hello').
• text: Stores a variable-length string without a limit (e.g., 'This is a long text').

3. Date/Time Types:

• date: Stores a date (e.g., '2024-12-03').
• timestamp: Stores both date and time (e.g., '2024-12-03 14:30:00').

4. Boolean Types:
• boolean: Stores TRUE, FALSE, or NULL (e.g., TRUE).
5. Numeric Types:
• numeric: Stores arbitrary precision numbers (e.g., 123.45).

PostgreSQL Operators

PostgreSQL supports various operators:
• Arithmetic Operators:
• +, -, *, /, % (e.g., SELECT 3 + 5; results in 8).
• Comparison Operators:
• =, !=, <, >, <=, >= (e.g., SELECT * FROM employees WHERE age > 30;).
• Logical Operators:
• AND, OR, NOT (e.g., SELECT * FROM employees WHERE age > 30 AND department = 'HR';).
• String Operators:
• || (concatenation) (e.g., SELECT 'Hello' || ' World'; results in 'Hello World').
• Range Operators:
• @> (contains), <@ (contained by) (e.g., SELECT * FROM ranges WHERE range @> 5;).

DDL (Data Definition Language) Operations

DDL operations are used to define and manage database schema. Examples include:
• CREATE TABLE: Defines a new table.

```
CREATE TABLE employees (
employee_id SERIAL PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
hire_date DATE
);
```

• ALTER TABLE: Modifies an existing table.

```
ALTER TABLE employees ADD COLUMN department VARCHAR(50);
```

• DROP TABLE: Deletes an existing table.

```
DROP TABLE employees;
```

• CREATE INDEX: Creates an index to improve query performance.
CREATE INDEX idx_employee_name ON employees (last_name);

DML (Data Manipulation Language) Operations

DML operations are used to manipulate data in the database. Examples include:
• INSERT: Adds new rows to a table.
INSERT INTO employees (first_name, last_name, hire_date)
VALUES ('John', 'Doe', '2024-12-03');
• SELECT: Retrieves data from one or more tables.
SELECT * FROM employees WHERE department = 'HR';
• UPDATE: Modifies existing rows in a table.
UPDATE employees SET department = 'Sales' WHERE employee_id = 1;
• DELETE: Removes rows from a table.
DELETE FROM employees WHERE employee_id = 1;

DCL (Data Control Language) Operations

DCL operations are used to control access to data in the database. Examples include:
• GRANT: Gives privileges to a user or role.
GRANT SELECT, INSERT ON employees TO user_name;
• REVOKE: Removes privileges from a user or role.
REVOKE INSERT ON employees FROM user_name;