POSTGRESQL INSERT Command

Purpose

The **INSERT** command is used to add one or more new rows of data into a table. Think of a table as a spreadsheet; the INSERT command is how you add a brand-new row to it. This is a fundamental operation for populating your database with data.

Syntax

The basic syntax for the INSERT command comes in a couple of common forms.

1. Inserting a new row into all columns:

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

2. Inserting a new row into specific columns:

```
INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2,
value3, ...);
```

Parameters

- **table_name**: The name of the table where you want to add the new data.
- **(column1, column2, ...)**: An optional, comma-separated list of columns you want to insert data into. If you omit this list, PostgreSQL assumes you are providing a value for every column in the table, in the same order they were created.
- **VALUES**: A required keyword that signals the start of the data values you are providing.
- **(value1, value2, ...)**: A comma-separated list of the actual data values to be inserted. The number of values and their data types must match the columns you specified (either explicitly or implicitly).

Examples

Let's imagine we have a table called products with the following columns: id, name, price, and stock.

Example 1: Inserting into all columns
If we want to add a new product and provide values for every column, we can use this command. Notice the values are in the same order as the columns were defined (id, name, price, stock).

```
INSERT INTO products VALUES (1, 'Laptop', 1200.00, 50);
```

Example 2: Inserting into specific columns
What if we want to add a new product but don't know the id yet (it's automatically generated) and the stock is zero? We can explicitly name the columns we are providing data for.

```
INSERT INTO products (name, price) VALUES ('Mouse', 25.50);
```

Example 3: Inserting multiple rows
You can insert several rows with a single command by separating the value lists with commas.

```
INSERT INTO products (name, price, stock) VALUES
('Keyboard', 75.00, 30),
('Monitor', 250.00, 15);
```

Use Cases

- **User Registration:** When a new user signs up on a website, an INSERT command adds a new row to the users table with their name, email, and password.
- **E-commerce:** When a new product is added to an online store's catalog, an INSERT command populates the products table with its name, price, and description.
- **Logging:** In an application, every time an important event occurs (like a successful login or an error), an INSERT command can add a new record to a logs table.

Best Practices

- **Match your data types:** Make sure the values you are inserting match the data types of the columns. For example, don't try to insert text into a numeric column.
- **Quote text and dates:** Always enclose text (VARCHAR) and date (DATE, TIMESTAMP) values in single quotes. Numbers do not need quotes.
- **Specify columns:** It's generally a good practice to explicitly list the columns you are inserting into (INSERT INTO table (col1, col2) ...). This makes your code more readable and prevents errors if the table structure changes in the future (e.g., a new column is added).
- **Use prepared statements:** When writing code that interacts with a database, use prepared statements or parameter binding to prevent SQL injection attacks. This means you use placeholders in your query instead of directly concatenating user input.

Related Commands

- **SELECT**: Retrieves data from a table.
- **UPDATE**: Modifies existing rows in a table.
- **DELETE**: Deletes existing rows from a table.
- **CREATE TABLE**: Creates a new table.