**Provide a detailed document for truncate command in PostgreSQL along with theory, syntax, 2 clear examples and best practices.**

The TRUNCATE command in PostgreSQL is used to quickly remove all rows from a set of tables. It's a **Data Definition Language (DDL)** command, which makes it much faster than DELETE for removing all rows.

**Theory**

TRUNCATE effectively deallocates all storage space used by the table data and resets the table to its initial empty state. Unlike DELETE, which removes rows one by one, TRUNCATE doesn't scan the table or process each row, making it a highly efficient operation. It's similar to a DROP followed by a CREATE, but it preserves the table's structure, including indexes, columns, and constraints (though not IDENTITY sequence values, which are reset).

TRUNCATE is also **transaction-safe**, meaning you can roll it back. However, it requires more privileges than DELETE, as it's a DDL command.

**Syntax**

The basic syntax for the TRUNCATE command is:

SQL

TRUNCATE TABLE table\_name [, table\_name2, ... ]

[ RESTART IDENTITY | CONTINUE IDENTITY ]

[ CASCADE | RESTRICT ];

* TRUNCATE TABLE table\_name: Specifies the table(s) you want to empty. You can truncate multiple tables in a single command.
* RESTART IDENTITY: (Optional) Resets any IDENTITY or SERIAL columns to their starting value. This is the default behavior in PostgreSQL versions 10 and later.
* CONTINUE IDENTITY: (Optional) Keeps the sequence counters as they are. New rows will continue from where the last one left off.
* CASCADE: (Optional) Automatically truncates all tables that have foreign-key references to the table(s) you're truncating.
* RESTRICT: (Optional) Throws an error if any tables have foreign-key references. This is the default behavior.

**Clear Examples**

Let's assume we have a table named logs.

**Example 1: Truncating a single table**

To quickly delete all records from the logs table and reset its SERIAL column:

SQL

TRUNCATE TABLE logs;

This command will remove all rows and reset any SERIAL columns so the next new row will start from 1 again.

**Example 2: Truncating with foreign key dependencies**

Assume we have a users table and a user\_posts table that references users. To truncate both tables at once and maintain consistency:

SQL

TRUNCATE TABLE users, user\_posts CASCADE;

This command will first truncate user\_posts (the child table) and then users (the parent table), ensuring all dependent data is removed correctly.

**Best Practices**

* **Use TRUNCATE for Mass Deletions**: Use TRUNCATE instead of DELETE with no WHERE clause for a significant performance boost when you want to remove all rows from a table.
* **Be Aware of Foreign Keys**: If you're truncating a table with foreign key dependencies, use the CASCADE option or truncate the dependent tables first to avoid errors.
* **Transactions are your friend**: Always wrap TRUNCATE commands within a transaction (BEGIN; ... COMMIT;). This allows you to ROLLBACK the operation if you realize you made a mistake, which is crucial since TRUNCATE is a destructive command.
* **Backup First**: Before performing any destructive operation, including TRUNCATE, ensure you have a recent backup of your database.