**Teradata**

**(https://www.tutorialspoint.com/teradata/teradata\_sub\_queries.htm)**

**Teradata Tutorial**

Teradata is a popular Relational Database Management System (RDBMS) suitable for large data warehousing applications. It is capable of handling large volumes of data and is highly scalable.

**Prerequisites**

You should have a basic understanding of Relational concepts and basic SQL. It will be good if you have worked with any other RDBMS product.

**What is Teradata?**

Teradata is one of the popular Relational Database Management System. It is mainly suitable for building large scale data warehousing applications. Teradata achieves this by the concept of parallelism. It is developed by the company called Teradata.

**Teradata\_Table Types**

Types Teradata supports different types of tables.

* **Permanent Table** − This is the default table and it contains data inserted by the user and stores the data permanently.
* **Volatile Table** − The data inserted into a volatile table is retained only during the user session. The table and data is dropped at the end of the session. These tables are mainly used to hold the intermediate data during data transformation.
* **Global Temporary Table** − The definition of Global Temporary table are persistent but the data in the table is deleted at the end of user session.
* **Derived Table** − Derived table holds the intermediate results in a query. Their lifetime is within the query in which they are created, used and dropped

|  |  |
| --- | --- |
| **Sr.No** | **Table Commands & Description** |
| 1 | [**Create Table**](https://www.tutorialspoint.com/teradata/teradata_create_table.htm) -- CREATE TABLE command is used to create tables in Teradata. |
| 2 | [**Alter Table**](https://www.tutorialspoint.com/teradata/teradata_alter_table.htm) -- ALTER TABLE command is used to add or drop columns from an existing table. |
| 3 | [**Drop Table**](https://www.tutorialspoint.com/teradata/teradata_drop_table.htm) --- DROP TABLE command is used to drop a table. |

**Teradata select syntax**

Following is the basic syntax of SELECT statement.

SELECT

column 1, column 2, .....

FROM

tablename;

**Teradata - Data Manipulation**

**Insert Records**

INSERT INTO statement is used to insert records into the table.

Syntax

Following is the generic syntax for INSERT INTO.

INSERT INTO <tablename>

(column1, column2, column3,…)

VALUES

(value1, value2, value3 …);

Teradata supports the following logical and conditional operators. These operators are used to perform comparison and combine multiple conditions.

|  |  |
| --- | --- |
| **Syntax** | **Meaning** |
| **>** | Greater than |
| **<** | Less than |
| **>=** | Greater than or equal to |
| **<=** | Less than or equal to |
| **=** | Equal to |
| **BETWEEN** | If values within range |
| **IN** | If values in <expression> |
| **NOT IN** | If values not in <expression> |
| **IS NULL** | If value is NULL |
| **IS NOT NULL** | If value is NOT NULL |
| **AND** | Combine multiple conditions. Evaluates to true only if all conditions are met |
| **OR** | Combine multiple conditions. Evaluates to true only if either of the conditions is met. |
| **NOT** | Reverses the meaning of the condition |

**Teradata set operators**

SET operators combine results from multiple SELECT statement. This may look similar to Joins, but joins combines columns from multiple tables whereas SET operators combines rows from multiple rows.

Rules

* The number of columns from each SELECT statement should be same.
* The data types from each SELECT must be compatible.
* ORDER BY should be included only in the final SELECT statement.

UNION

UNION statement is used to combine results from multiple SELECT statements. It ignores duplicates.

**Syntax**

Following is the basic syntax of the UNION statement.

SELECT col1, col2, col3…

FROM

<table 1>

[WHERE condition]

UNION

SELECT col1, col2, col3…

FROM <table 2>

[WHERE condition];

Teradata provides several functions to manipulate the strings. These functions are compatible with ANSI standard.

|  |  |
| --- | --- |
| **Sr.No** | **String Function & Description** |
| 1 | **||:--**Concatenates strings together |
| 2 | **SUBSTR:--**Extracts a portion of a string (Teradata extension) |
| 3 | **SUBSTRING:--**Extracts a portion of a string (ANSI standard) |
| 4 | **INDEX:--**Locates the position of a character in a string (Teradata extension) |
| 5 | **POSITION:--**Locates the position of a character in a string (ANSI standard) |
| 6 | **TRIM:--**Trims blanks from a string |
| 7 | **UPPER:--**Converts a string to uppercase |
| 8 | **LOWER:--**Converts a string to lowercase |

**Tera data built-in functions**

Teradata provides built-in functions which are extensions to SQL. Following are the common built-in functions.

|  |  |
| --- | --- |
| **Function** | **Result** |
| SELECT DATE; | Date -------- 16/01/01 |
| SELECT CURRENT\_DATE; | Date -------- 16/01/01 |
| SELECT TIME; | Time -------- 04:50:29 |
| SELECT CURRENT\_TIME; | Time -------- 04:50:29 |
| SELECT CURRENT\_TIMESTAMP; | Current TimeStamp(6) -------------------------------- 2016-01-01 04:51:06.990000+00:00 |
| SELECT DATABASE; | Database ------------------------------  TDUSER |

**Tera data aggregate functions**

Teradata supports common aggregate functions. They can be used with the SELECT statement.

* **COUNT** − Counts the rows
* **SUM** − Sums up the values of the specified column(s)
* **MAX** − Returns the large value of the specified column
* **MIN** − Returns the minimum value of the specified column
* **AVG** − Returns the average value of the specified column

**Teradata case condition**

This chapter explains the CASE and COALESCE functions of Teradata.

CASE Expression

CASE expression evaluates each row against a condition or WHEN clause and returns the result of the first match. If there are no matches then the result from ELSE part of returned.

Syntax

Following is the syntax of the CASE expression.

CASE <expression>

WHEN <expression> THEN result-1

WHEN <expression> THEN result-2

ELSE

Result-n

END

* **Unique Primary Index(UPI) & Non Unique Primary Index(NUPI)**

Primary index is defined while creating a table. There are 2 types of Primary Indexes.

* Unique Primary Index(UPI)
* Non Unique Primary Index(NUPI)

Unique Primary Index (UPI)

If the table is defined to be having UPI, then the column deemed as UPI should not have any duplicate values. If any duplicate values are inserted, they will be rejected.

Create Unique Primary Index

The following example creates the Salary table with column EmployeeNo as Unique Primary Index.

CREATE SET TABLE Salary (

EmployeeNo INTEGER,

Gross INTEGER,

Deduction INTEGER,

NetPay INTEGER

)

UNIQUE PRIMARY INDEX(EmployeeNo);

Ezoic

Non Unique Primary Index (NUPI)

If the table is defined to be having NUPI, then the column deemed as UPI can accept duplicate values.

Create Non Unique Primary Index

The following example creates the employee accounts table with column EmployeeNo as Non Unique Primary Index. EmployeeNo is defined as Non Unique Primary Index since an employee can have multiple accounts in the table; one for salary account and another one for reimbursement account.

CREATE SET TABLE Employee \_Accounts (

EmployeeNo INTEGER,

employee\_bank\_account\_type BYTEINT.

employee\_bank\_account\_number INTEGER,

employee\_bank\_name VARCHAR(30),

employee\_bank\_city VARCHAR(30)

)

PRIMARY INDEX(EmployeeNo);

**Teradata joins**

Join is used to combine records from more than one table. Tables are joined based on the common columns/values from these tables.

There are different types of Joins available.

* Inner Join
* Left Outer Join
* Right Outer Join
* Full Outer Join
* Self Join
* Cross Join
* Cartesian Production Join

INNER JOIN

Inner Join combines records from multiple tables and returns the values that exist in both the tables.

Syntax

Following is the syntax of the INNER JOIN statement.

SELECT col1, col2, col3….

FROM

Table-1

INNER JOIN

Table-2

ON (col1 = col2)

<WHERE condition>;

**Teradata subquery**

A subquery returns records from one table based on the values from another table. It is a SELECT query within another query. The SELECT query called as inner query is executed first and the result is used by the outer query. Some of its salient features are −

* A query can have multiple subqueries and subqueries may contain another subquery.
* Subqueries doesn't return duplicate records.
* If subquery returns only one value, you can use = operator to use it with the outer query. If it returns multiple values you can use IN or NOT IN.

Syntax

Following is the generic syntax of subqueries.

SELECT col1, col2, col3,…

FROM

Outer Table

WHERE col1 OPERATOR ( Inner SELECT Query);

SELECT EmployeeNo, NetPay

FROM Salary

WHERE NetPay =

(SELECT MAX(NetPay)

FROM Salary);

**Difference between sql and teradata sql**

SQL (Structured Query Language) is a standard language for managing relational databases, while Teradata SQL is a dialect of SQL specifically designed for Teradata databases.

**Key differences:**

SQL (Standard)

1. ANSI/ISO standard

2. Portable across various databases (e.g., MySQL, PostgreSQL, SQL Server)

3. Supports basic SQL features:

SELECT, INSERT, UPDATE, DELETE

- JOIN, SUBQUERY, AGGREGATE functions

- INDEX, VIEW, STORED PROCEDURE

**Teradata SQL**

1. Proprietary dialect for Teradata databases

2. Optimized for parallel processing and large-scale data warehousing

3. Extensions to standard SQL:

**- Supports advanced features:**

- MULTI-LEVEL SUBQUERIES

- CORRELATED SUBQUERIES

- FULL OUTER JOIN

**- Teradata-specific functions:**

- HASH Functions (e.g., HASHBUCKET, HASHROW)

- DATA DEMOCRATIZATION (e.g., SAMPLE, STRATIFY)

- Support for Teradata's MPP (Massively Parallel Processing) architecture

Teradata-specific features:

1. Parallel processing: Teradata SQL optimizes queries for parallel execution across multiple nodes.

2. Hashing: Teradata's hashing algorithms improve data distribution and query performance.

3. Data democratization: Teradata SQL provides functions for data sampling and stratification.

4. Multi-level subqueries: Teradata SQL supports complex subquery structures.

5. Correlated subqueries: Teradata SQL optimizes correlated subqueries for better performance.

SQL features not supported in Teradata SQL:

1. Common Table Expressions (CTEs)

2. Window functions (limited support)

3. MERGE statement

4. SEQUENCE object