SQL in Teradata

In this lecture

- DDL, DML, DCL (review of SQL commands).
- Teradata built-in and aggregate functions

Many faces of SQL

Formally, SQL is actually *three* languages:

- **DDL:** Data definition language.
- **DML:** Data manipulation language.
- **DCL:** Data control language.

DDL

Table 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.

Table types (cont.)

- **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.

`SET` VS `MULTISET`

- Teradata classifies the tables as `SET` or `MULTISET` tables based on how the duplicate records are handled.
- A table defined as `SET` table doesn't store the duplicate records.
- A `MULTISET` table can store duplicate records.
- **Tip**: If you are sure rows *will* be unique, you can set the table as multiset. This avoids unnecessary uniqueness checks, for instance, for `insert`.

CREATE TABLE

The syntax is:

```
CREATE <SET/MULTISET> TABLE <Tablename>
< Table Options>
<Column Definitions>
<Index Definitions>;
```

Syntax decoded

- `<Table Options>` specifies the physical attributes of the table such as `JOURNAL` and `FALLBACK`.
    - `JOURNAL`: Maintain data integrity in the event of component or process failure (restore to prescribed point in time).
    - `FALLBACK`: Storing a second copy of each row on a different AMP.
- `<Column Definitions>` specifies the list of columns, data types and their attributes.
- `<Index Definitions>` additional indexing options such as Primary Index.

Example

```
CREATE SET TABLE EMPLOYEE,FALLBACK
(
      EmployeeNo INTEGER,
      FirstName VARCHAR(30) ,
      LastName VARCHAR(30) ,
      DOB DATE FORMAT 'YYYY-MM-DD',
      JoinedDate DATE FORMAT 'YYYY-MM-DD',
      DepartmentNo BYTEINT
 )
UNIQUE PRIMARY INDEX ( EmployeeNo );
```

SHOW TABLE

```
 SHOW TABLE Employee;
 *** Text of DDL statement returned.
 *** Total elapsed time was 1 second.
------------------------------------------------------------
CREATE SET TABLE EMPLOYEE ,FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      EmployeeNo INTEGER,
      FirstName VARCHAR(30),
      LastName VARCHAR(30),
      DOB DATE FORMAT 'YYYY-MM-DD',
      JoinedDate DATE FORMAT 'YYYY-MM-DD',
      DepartmentNo BYTEINT)
UNIQUE PRIMARY INDEX ( EmployeeNo );
```

ALTER TABLE

```
ALTER TABLE <tablename>
ADD <columnname> <column attributes>
DROP <columnname>;
```

Example:

```
ALTER TABLE employee
ADD BirthDate DATE FORMAT 'YYYY-MM-DD',
DROP DOB;
```

DROP TABLE

- `DROP TABLE <tablename>;`
- `DROP TABLE IF EXISTS <tablename>;` sadly not existant.
- From Teradata 13.10, you can use BTEQ syntax in SQLA:

```
SELECT 1 FROM dbc.TablesV
WHERe databasename = <your db>
AND TableName = '<table>';
.if activitycount = 0 then GoTo ok
DROP TABLE <table>;
.label ok
```

In general:

- `ALTER`
- `CREATE`
- `DROP`
- `MODIFY`
- `RENAME`
- `REPLACE`
- `SET ROLE`, `SET SESSION`, `SET TIME ZONE`

VIEWS

- Same DDL commands apply to **views**.
- Views are queries build on demand, they are useful to restrict data to a level of aggregation.

  Example

```
CREATE VIEW  products_more_than_3_sold AS
SELECT  productid, productname, productprice
FROM  product
WHERE  productid IN (
                        SELECT  productid
                        FROM  soldvia
                        GROUP BY productid
                        HAVING SUM(noofitems) > 3);
```

DML

`SELECT`, `INSERT`

```
SELECT column1, column2 FROM table
```

```
INSERT INTO table (column1, column2)
VALUES ('value1','value 2')
```

```
INSERT INTO <tablename>
(column1, column2, column3,…)
SELECT
column1, column2, column3…
FROM
<source table>;
```

`UPDATE`, `DELETE`

```
UPDATE table
SET column2 = 'value 3' WHERE column1 = 'value1'
```

```
 DELETE <databasename>
 DELETE <username>
 DELETE FROM <TABLENAME>
```

Tip: `DELETE` + `INSERT INTO` is usually faster than `UPDATE`.

DCL

- `GIVE`
- `GRANT`
- `GRANT LOGON`
- `REVOKE`
- `REVOKE LOGON`

Set Operators

- These operators combine results from multiple `SELECT` statements.
  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` is used to combine results from multiple SELECT statements. It ignores duplicates.

```
SELECT col1, col2, col3…
FROM  <table 1>
[WHERE condition]
UNION
SELECT col1, col2, col3…
FROM  <table 2>
[WHERE condition];
```

`UNION ALL`

- `UNION ALL` is similar to `UNION`. It combines results from multiple tables including duplicate rows.

Example

`UNION ALL`	`UNION`
EmployeeNo	EmployeeNo
101	101
104	102
102	103
105	104
103	105
101	
104	

`INTERSECT` and `MINUS` / `EXCEPT`

- `INTERSECT` returns the intersection (common rows).
- `MINUS` or `EXCEPT` return the rows in the first table, but removing the rows in the second table.

```
SELECT EmployeeNo
FROM
Employee
INTERSECT
SELECT EmployeeNo
FROM
Salary;
```

Built-in and aggregate functions

Built-in functions

- These are functions which do not need any sort of arguments but still return information about the system; user, date, time, session etc.
- Built-in functions are sometimes referred to as special registers.
- Syntax: `SELECT <Built-In Function>`

Built-in functions (cont.)

- `ACCOUNT:` – Your Teradata Account information.
- `CURRENT_DATE:` Returns the current system date.
- `CURRENT_TIME:` This function returns the current system time and current session ‘Time Zone’ displacement.
- `CURRENT_TIMESTAMP:` Returns the current system timestamp (including year, month and day) and current session.

Built-in functions (cont.)

- `DATABASE:` Returns the name of the default database for the current user.
- `DATE:` Same as `CURRENT_DATE`.
- `SESSION:` Returns a number for the session the current user is in.
- `TIME:` Current time based on a 24-hour day; mean to say for 4:00 pm, you would see 16:00:00.
- `USER:` If you have forgotten your username after you have logged in, this command would come to your rescue.

Aggregate

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.

Your turn!

Exercise/Lab

- On the `tutorial` database we created, look for the `product` table.

1. Create a view that shows a business user the `productid`, `productname` and `productprice` of products sold more than three times.
2. Create a view from the same table (with the same columns) that shows products sold in multiple transactions.

- **Hint:** You can borrow inspiration from the slides, be sure you understand what is going on.

Exercise/Lab (cont.)

3. Which products are sold often (more than three times) and to many customers (appear in many transactions).
4. Which products are sold often, but in one transaction?
5. Which products (from the `product` table) are sold at a price below average?