Lab assignment-11

Cursors:

A cursor is a variable that runs through the tuples of some relation. This relation can be a stored table, or it can be the answer to some query. By fetching into the cursor each tuple of the relation, we can write a program to read and process the value of each such tuple. If the relation is stored, we can also update or delete the tuple at the current cursor position. The example below illustrates a cursor loop. It uses our example relation T1(e,f) whose tuples are pairs of integers. The program will delete every tuple whose first component is less than the second, and insert the reverse tuple into T1.

- Types:
  - Implicit (Generated by oracle)
  - Explicit
  - Steps for creating and using explicit cursor
  - Declare cursor
  - Open cursor
  - Fetch data rows
  - Close cursor

CURSOR *cursor_name* IS *select_query*;

cursor_name: Any valid PL/SQL name.

select_query: The query that retrieves the data values

Considering the tables of Assignment 7, perform the following tasks:

1. Create a cursor to fetch the count of customers and sellers.
2. Create a cursor to display all the product details with rating more than 4.5.
3. Create a cursor to display all the products category wise.
4. Display Seller ID, Seller name and Rating of all employees using cursors.
5. Display Product details having highest Amount using cursor.
6. Display Rating of all Sellers in descending order using cursor.

Triggers:

A trigger (essentially, a stored SQL statement associated with a table) is a database object that defines events that happen when some other event, called a triggering event, occurs. Create a trigger by using the CREATE TRIGGER statement. Triggers execute when an INSERT, UPDATE, or DELETE modifies a specified column or columns in the subject table. Typically, the stored SQL statements perform an UPDATE, INSERT, or DELETE on a table different from the subject table.

Sometimes a statement fires a trigger, which in turn, fires another trigger. Thus the outcome of one triggering event can itself become another trigger. The Teradata RDBMS processes and optimizes the triggered and triggering statements in parallel to maximize system performance.

Trigger Functions :

Use triggers to perform various functions:

• Define a trigger on the parent table to ensure that UPDATEs and DELETEs performed to the parent table are propagated to the child table.

• Use triggers for auditing. For example, you can define a trigger which causes INSERTs in a log record when an employee receives a raise higher than 10%.

• Use a trigger to disallow massive UPDATEs, INSERTs, or DELETEs during business hours.

For example, you can use triggers to set thresholds for inventory of each item by store, to create a purchase order when the inventory drops below a threshold, or to change a price if the daily volume does not meet expectations.

Restrictions on Using Triggers :

Teradata triggers do not support FastLoad and MultiLoad utilities and, and you must disable triggers before you run load utilities. In addition, a positioned (updatable cursor) UPDATE or DELETE is not allowed to fire a trigger and generates an error.

Note: You cannot define a join index on a table with a trigger.

```
CREATE TRIGGER <triggername> AFTER UPDATE/INSERT/DELETE  OF   <COLUMN
NAME> ON <TABLENAME> FOR EACH ROW
BEGIN
-----
-----
executable statements;
-----
-----
END;
```

Considering the tables of Assignment 7, perform the following tasks:

1. Create a trigger to update the remaining quantity of product in the product table, when a new entry in order_products table is inserted.

2. Create a trigger to update product rating and seller rating when a new entry in the order_products table is inserted.

3. Create a trigger to check when a new entry is to be inserted in the order_products table the quantity column satisfies the remaining quantity column from the product table.

***