**RETAIL BUSINESS MANAGEMENT SYSTEM (TEAM REPORT)**

Members:

Neha Lad

Rajvi Mistry

Shweta Jadhav

**Description**

The Retail Business Management System was implemented using oracle PL/SQL and JDBC. It involved the work of writing procedures, triggers, cursors and sequences at the backend, establishing JDBC connectivity and displaying output on demand using Java code.

**Details**

We started working on this project from 13 November, 2017. First we implemented the question 1 as sequences are an important since they generate primary keys and id values for the tables. Shweta and Neha did the work of writing triggers, cursors and procedures. Hence, Neha worked on 1, 3, 4, 5, 8 and Shweta worked on Items 2, 6, 7,9. Rajvi worked mainly on establishing JDBC connectivity with the database and working with ref cursor on the PL/SQL code blocks, which Neha and Shweta completed.

We arranged meetings on alternate days from Nov 13, 2017 till the deadline of this project. During the meetings, we discussed individual tasks to be completed and problems if we were stuck on a certain task till next meeting and deciding the number of resources that can work on a particular task.

**Experience**

Previously we never worked on concepts like Sequences, Cursors, Procedures and Triggers in Oracle. This project gave us an practical experience of working on Oracle PL/SQL.

**Individual Tasks and Roles:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Neha** | **Shweta** | **Rajvi** |
| 14 Nov 2017 | Completed working on query 1 | Working on query 2 | Working on JDBC installation and connection. |
| 16 Nov 2017 | Working on query 3 | Completed query 2 and started to work on query 6 | Solving connectivity issues with remote server. |
| 20 Nov 2017 | Started to work on query 4. | Completed query 6 and working on modifying query 3. | Implemented query 1 in JDBC. |
| 22 Nov 2017 | Completed working on query 4 and started to work on query 5 | Started to work on query 7. | Worked on issues with ref cursor in query 3. |
| 23 Nov 2017 | Completed query 5 and started on query 8 | Working on query 7 c, d part | Implemented 2,3,4,5 in JDBC. |
| 24 Nov 2017 | Working on query 8 | Completed query 7 and working on query 9. | Started to Work on the GUI part of the project. |
| 25 Nov 2017 | Completed query 8 and started with testing of all the queries. | Completed query 9 and started on testing. | Implemented all querries in User Interface while still working on GUI part. |
| 26 Nov 2017 | Working on report | Working on report | Working on report |

**PL/SQL Code Description**

The project is included in Project2\_Package. This package contains following files :

1. sequences.sql : In this file we declared 3 sequences as pur#\_seq, sup#\_seq and log#\_seq for purchases, suppliers and logs table
2. Triggers.sql : This file creates all the triggers that are mentioned in question 6 and solves some part of question 7 after adding a tuple to purchases table.
3. Project2\_header.sql : This file contains the initialization of all function headers and procedure headers that are useful for the project.
4. Project2\_body.sql : This file contains the all the definitions of Project2\_headers.
5. Proj2\_data.sql : It contains a script to initialize tables and insert data into tables.

**Procedures and Functions**

1. show\_employees () – This function is used to display all tuples that are currently present in the employees Table.
2. show\_customers () – This function is used to display all tuples that are currently present in the customers Table.
3. show\_products () - This function is used to display all tuples that are currently present in the products Table.
4. show\_purchases () – This function is used to display all tuples that are currently present in the purchases Table.
5. show\_suppliers () – This function is used to display all tuples that are currently present in the suppliers Table.
6. show\_supplies() - This function is used to display all tuples that are currently present in the supplies Table.
7. show\_discounts() - This function is used to display all tuples that are currently present in the discounts Table.
8. show\_logs () – This function is used to display all tuples that are currently present in the logs Table.
9. purchase\_saving(pur#) – This function is used to total saving of a purchase with pur# as a parameter
10. monthly\_sale\_activities(employee\_id) – This procedure takes employee id as a parameter and calculates the monthly sale activity of an employee of those months when the employee has made an actual sale.
11. add\_Customer(c\_id, c\_name, c\_telephone#) – This procedure adds a tuple to customers table with customers id, name and telephone number as given parameters.
12. add\_Purchases(e\_id, p\_id, c\_id, pur\_qty) – This procedure adds a new purchase to purchases table with eid, pid, cid and qty as parameters. New added pur# is automatically generated by pur#\_seq. We calculated total\_price using the formula

total\_price= (original\_price \* (1-discnt\_rate)) \* qty.

Before adding a tuple to purchases table, this procedure checks the condition for the involved product, the quantity to be purchased is equal to or smaller than the quantity on hand (qoh) i.e. If (total\_qoh-pur\_qty)>=0 then only a purchase is added or the purchase request is rejected.

1. delete\_Purchase(pur#) – This procedure deletes a purchase with pur# as a parameter to simulate the returning purchase. After deletion all data in database is made consistent by using triggers setQohTrigger and setVisitsTrigger.

**Sequences**

1. pur#\_seq is a sequence for purchase table. We start the sequence with 100015 (since 100014 is the last entry in the table) and increment by 1 each time. Also, the numbers are generated in order.
2. sup#\_seq is a sequence for supplies table. We start the sequence with 1010 (since 1009 is the last entry currently in the table) and increment by 1 each time. After this value, we can cycle. The numbers are generated in order.
3. log#\_seq is a sequence for logs table. We start the sequence with 10000 (since it is the smallest 5 digit number) and increment by 1 each time. It can have a maximum value of 99999. Also, the numbers are generated in order.

**Triggers**

1. insertCustomersTrigger: This will add a tuple to logs table every time we added a new entry to customers table with tuple\_pkey is cid of newly added customer.
2. updateLastVisitDateTrigger: This trigger that will add a tuple into a logs table every time we update the last\_visit\_date of customer. The tuple\_pkey is cid of affected customer.
3. updateQohTrigger: This trigger will add a tuple into a logs table every time we update the qoh value of a product. The tuple\_pkey is pid of affected product.
4. insertSuppliesTrigger: This trigger will add a tuple into a logs table every time we insert new entry into supplies table. Here according to the quantity of supply ordered of a product we update the qoh of product.
5. insertPurchaseTrigger: This trigger that will add a tuple into a logs table every time we insert a new entry to purchases table. After adding tuple to logs table we update the qoh of products table to be reduced by the quantity of the product that was just purchased. Then we check if the new qoh is below qoh\_threshold.

If qoh is less than qoh\_threshold, we print a message that current qoh is less than qoh\_threshold and a new supply is required. And we order a supply with using sequence for next sup#.

The quantity of the new supply is computed using 10 + M + qoh, where M is the minimum value for quantity such that M + qoh > qoh\_threshold, and we used sysdate for sdate. Sid is the min(sid). We updated the qoh with this new value of quantity.

Next we updated the visits\_made attribute of customers. We compare the last\_visit\_date of the customer with the date entered while inserting into purchases. If the date is not equal, then we update the last\_visit\_date of the customer.

1. setQohTrigger : After deleting a purchase from purchases table the setQohTrigger will update the Qoh of products table with returning quantity of products in deleted purchase that is it will update the Qoh with addition of the current qoh in products and deleted purchases products quantity.
2. setVisitsTrigger : This trigger will update the visits\_made and last\_visit\_date of a customer after a purchase is deleted. The visits\_made will be increased by 1 and last\_visit\_date will set to sysdate since the visit to return a purchase is also considered a visit to the retail business.