Ahsanullah University of Science & Technology

Department of Computer Science and Engineering

Project Name: Restaurant Management System

CSE 4126 Spring - 2018 - Project Report

Submitted by:

Debashis Roy	**(15.01.04.070)**

Other Members

Khandakar Mahmudur Rahman	**(15.01.04.066)**
Prottasa Karim	**(15.01.04.080)**

Submitted to:

Mr. Mohammad Imrul Jubair

Assistant Professor

Safrun Nesa Saira

Lecturer

Introduction:

“Restaurant management system” is a management system for managing the branches of the restaurant such as daily sales, working schedule, coordinating the employees etc. It is very common to use a centralized database in a restaurant. But for a successful restaurant it is common to have many branches in different cities or locations or in different country. So, in this case, it is badly in need of a distributed database management system to manage and look over the whole database management system. Distributed database management system not only makes the system easier but also makes operation faster to operate in a particular location. Searching for a particular data in the whole database is unnecessary and complicated for the operating system as well. So, we can avoid such inessential operations just by making the database system distributed.

Overview of the System:

Restaurant Management System is an automation system for a restaurant which performs different functionalities such as storing customer info, branch info, sale info, employee info etc. Depending on this management system many activities can be performed in future like notifying customers about new items or offers on food items or new branch opening through email or phone etc.

The flow of this system is that, the librarian logs into the system. Now, different branches have different managers. They store daily sale info, employee info and customer info. According to ‘ID’, customers and employees are distributed in different sites.

Features and Functionalities –

Storing information about customers, employees, menu items, categories etc. Operations like searching particular food items, inserting in branches, updating and delete information, creating fragments of 9 global relations for different sites and assigning data in corresponding sites.

User – right now the user is only the manager of the brunch and admin.

Sites – There are two sites in this management system. They are distributed location wise to operate the system fast and with less complexity.

Relations and Sites:

<u>Global Relations</u>:

employee(EID, EName, Address, Phone, Position, Salary)

catagories(CID, CName)

menuitems(ItemID, ItemName, Price, Description, CID)

customer(CustomerID, CustomerName, Address)

reservation(RID, duration, RDate, RTime, CustomerID,)

diningTables(DID, WID, ChairCount)

orders(OID, ODate, Quantity, CustomerID, ItemID)

diningTableTrack(Serial,OID, DID, TableServesDate, TableServesTime)

reserves(RID, DID)

bills(BID, OID, CustomerID, Discount¸ Amount, BDate, BTime)

<u>Fragmentation Schema</u>:

employee1=$SL_{EID<3000}$(employee)

employee2=$SL_{EID>3000}$(employee)

orders1=$SL_{OID<=2298}$(orders)

orders2=$SL_{OID>=2298}$(orders)

catagories1=$SL_{CID<3000}$(catagories)

catagories2=$SL_{CID>3000}$(catagories)

menuitems1=$SL_{ItemID<3000}$(menuitems)

menuitems2=$SL_{ItemID >3000}$(menuitems)

customer1=$SL_{CustomerID<3000}$(customer)

customer2=$SL_{CustomerID >3000}$(customer)

reservation1=$SL_{RID<3000}$(reservation)

reservation2=SL $_{\text{RID >3000}}$(reservation)

diningTables1=SL $_{\text{DID<3000}}$(diningTables)

diningTables2=SL $_{\text{DID >3000}}$(diningTables)

diningTableTrack1=SL $_{\text{Serial<3000}}$(diningTableTrack)

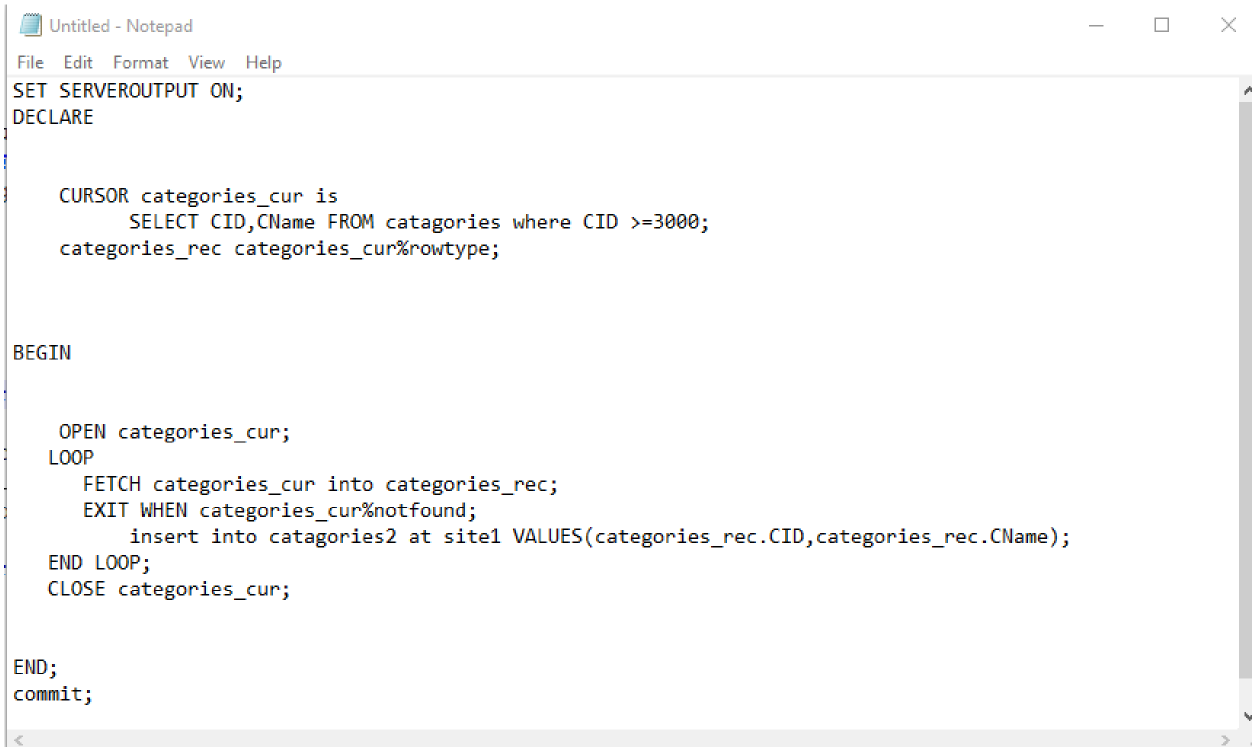
diningTableTrack2=SL $_{\text{Serial >3000}}$(diningTableTrack)

bills1=SL $_{\text{BID<3000}}$(bills)

bills2=SL $_{\text{BID >3000}}$(bills)

How Fragmentation populated:

For populating our fragments from the global tables we used plslq codes. For the insertion of new rows into the fragments we checked for the required conditions and insert in into respective fragment. A snapshot of the code is



```
SET SERVEROUTPUT ON;
DECLARE


    CURSOR categories_cur is
          SELECT CID,CName FROM catagories where CID >=3000;
    categories_rec categories_cur%rowtype;



BEGIN


    OPEN categories_cur;
   LOOP
      FETCH categories_cur into categories_rec;
      EXIT WHEN categories_cur%notfound;
          insert into catagories2 at site1 VALUES(categories_rec.CID,categories_rec.CName);
   END LOOP;
   CLOSE categories_cur;


END;
commit;
```

Allocation Schema:

There are two sites.

Site 1(Dhanmondi): employee1, orders1, catagories1, menuitems1, customer1, reservation1, diningTables1, diningTableTrack1, bills1, reserves.

Site 2(Bailyroad):): employee2, orders2, catagories2, menuitems2, customer2, reservation2, diningTables2, diningTableTrack2, bills2, reserves.

Contribution:

My contributions in this project are to implement i)Functions Procdures ii)Sequence iii)Fragmentation iv)database profile v)Trigger procedure package

1.Function Procedures:

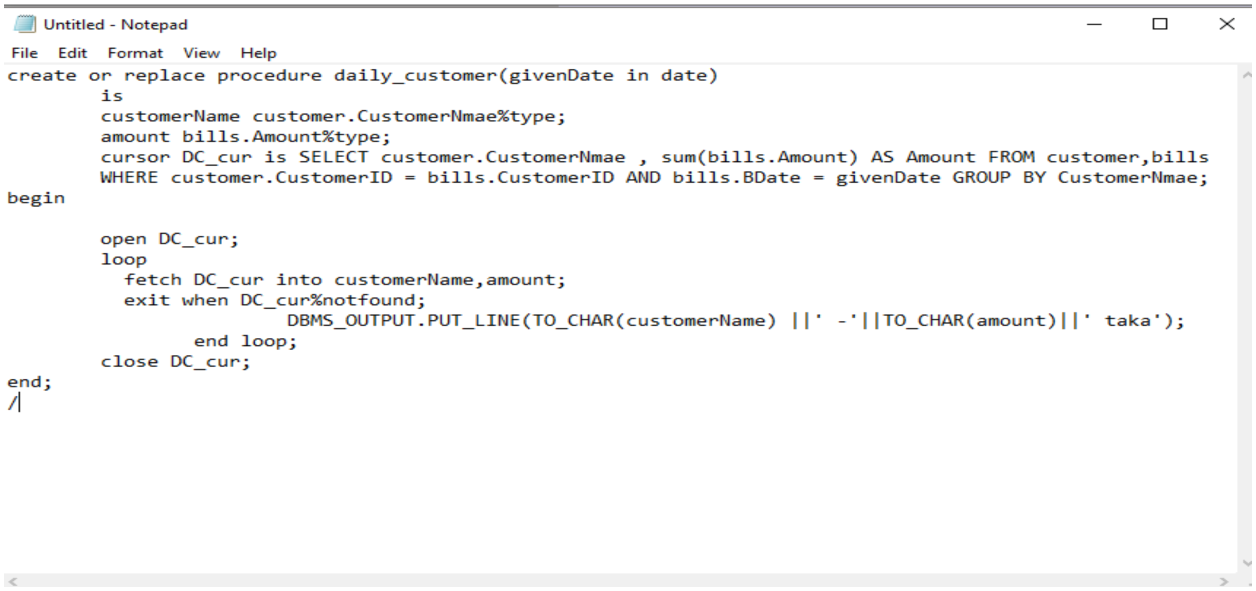
Functions can have only input parameters for it whereas Procedures can have input or output parameters . Functions can be called from Procedure whereas Procedures cannot be called from a Function.

Different functions and procedures are used for different purposes.

i.daily_Customer:

This procedure is used to fetch the information of the customers who is visited in our restaurant and buy some food. The date is used as input.

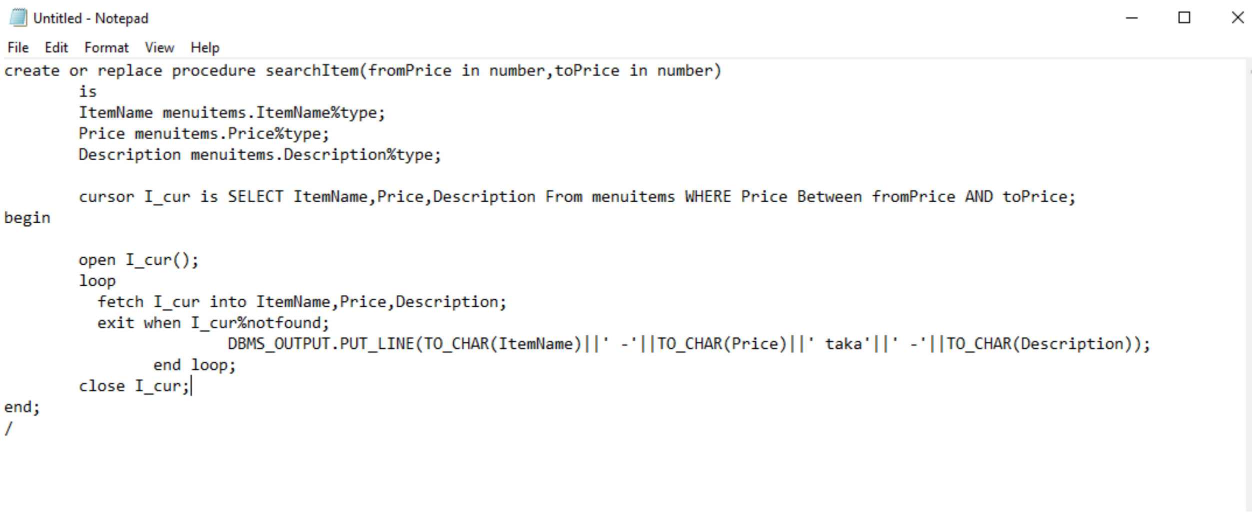
CODE:



```
create or replace procedure daily_customer(givenDate in date)
        is
        customerName customer.CustomerNmae%type;
        amount bills.Amount%type;
        cursor DC_cur is SELECT customer.CustomerNmae , sum(bills.Amount) AS Amount FROM customer,bills
        WHERE customer.CustomerID = bills.CustomerID AND bills.BDate = givenDate GROUP BY CustomerNmae;
begin

        open DC_cur;
        loop
          fetch DC_cur into customerName,amount;
          exit when DC_cur%notfound;
                      DBMS_OUTPUT.PUT_LINE(TO_CHAR(customerName) ||' -'||TO_CHAR(amount)||' taka');
              end loop;
        close DC_cur;
end;
/
```

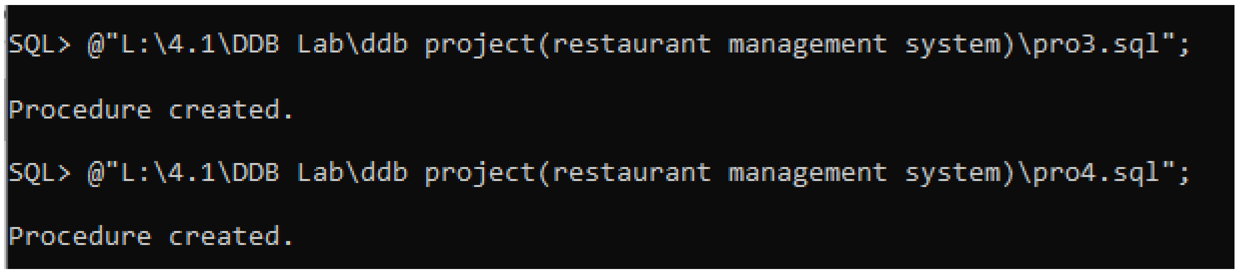
ii) SearchItem:

The aim of using this procedure is to get the information of the item in the given price range. The two price range is used as input.



```
create or replace procedure searchItem(fromPrice in number,toPrice in number)
        is
        ItemName menuitems.ItemName%type;
        Price menuitems.Price%type;
        Description menuitems.Description%type;

        cursor I_cur is SELECT ItemName,Price,Description From menuitems WHERE Price Between fromPrice AND toPrice;
begin

        open I_cur();
        loop
          fetch I_cur into ItemName,Price,Description;
          exit when I_cur%notfound;
                        DBMS_OUTPUT.PUT_LINE(TO_CHAR(ItemName)||' -'||TO_CHAR(Price)||' taka'||' -'||TO_CHAR(Description));
                end loop;
        close I_cur;
end;
/
```

OUTPUT :

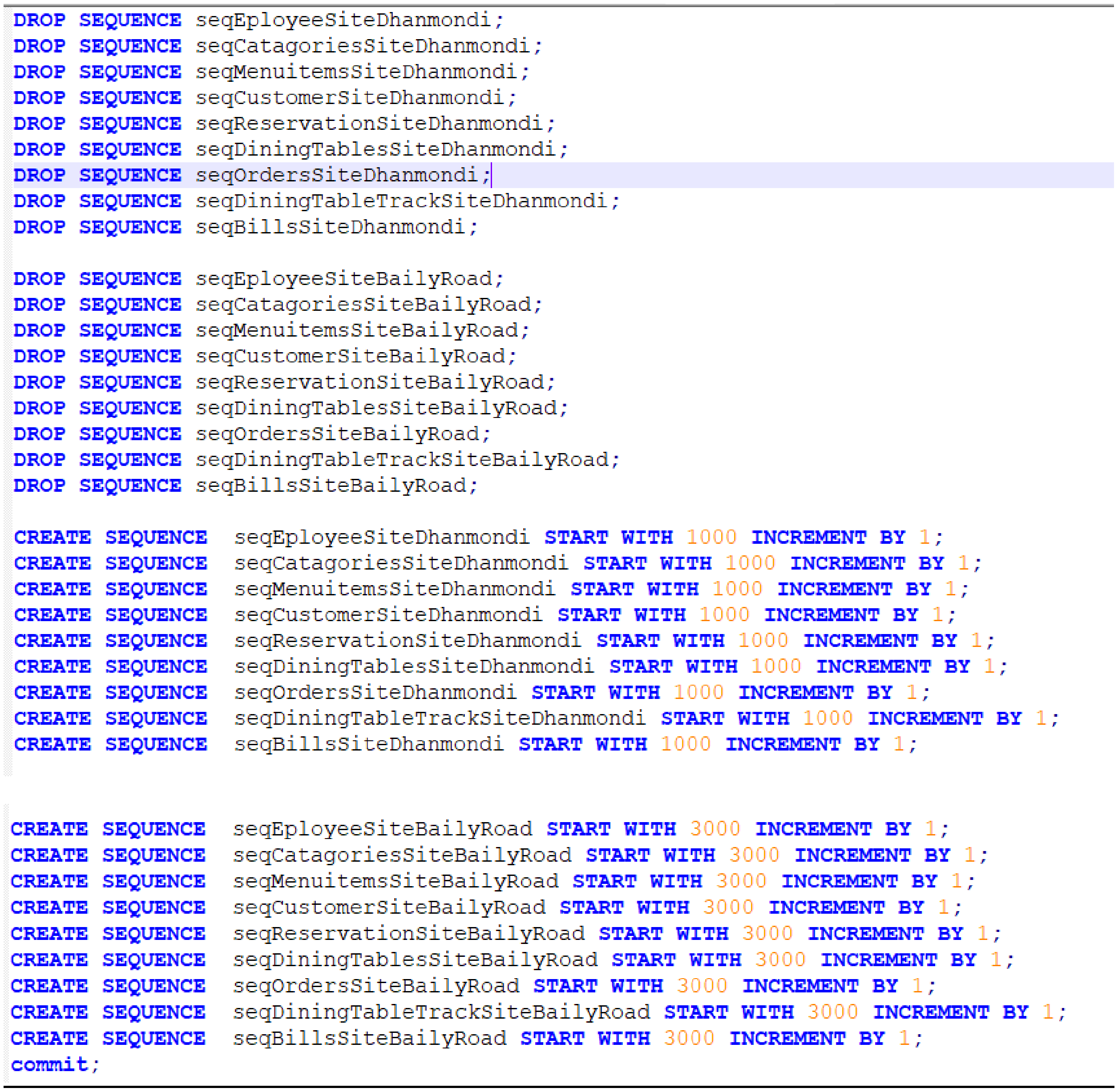


```
SQL> @"L:\4.1\DDB Lab\ddb project(restaurant management system)\pro3.sql";

Procedure created.

SQL> @"L:\4.1\DDB Lab\ddb project(restaurant management system)\pro4.sql";

Procedure created.
```

2.Sequence:

Sequence is a feature supported by some **database systems** to produce unique values on demand. Some **DBMS** like MySQL supports AUTO_INCREMENT in place of **Sequence**.

CODE:

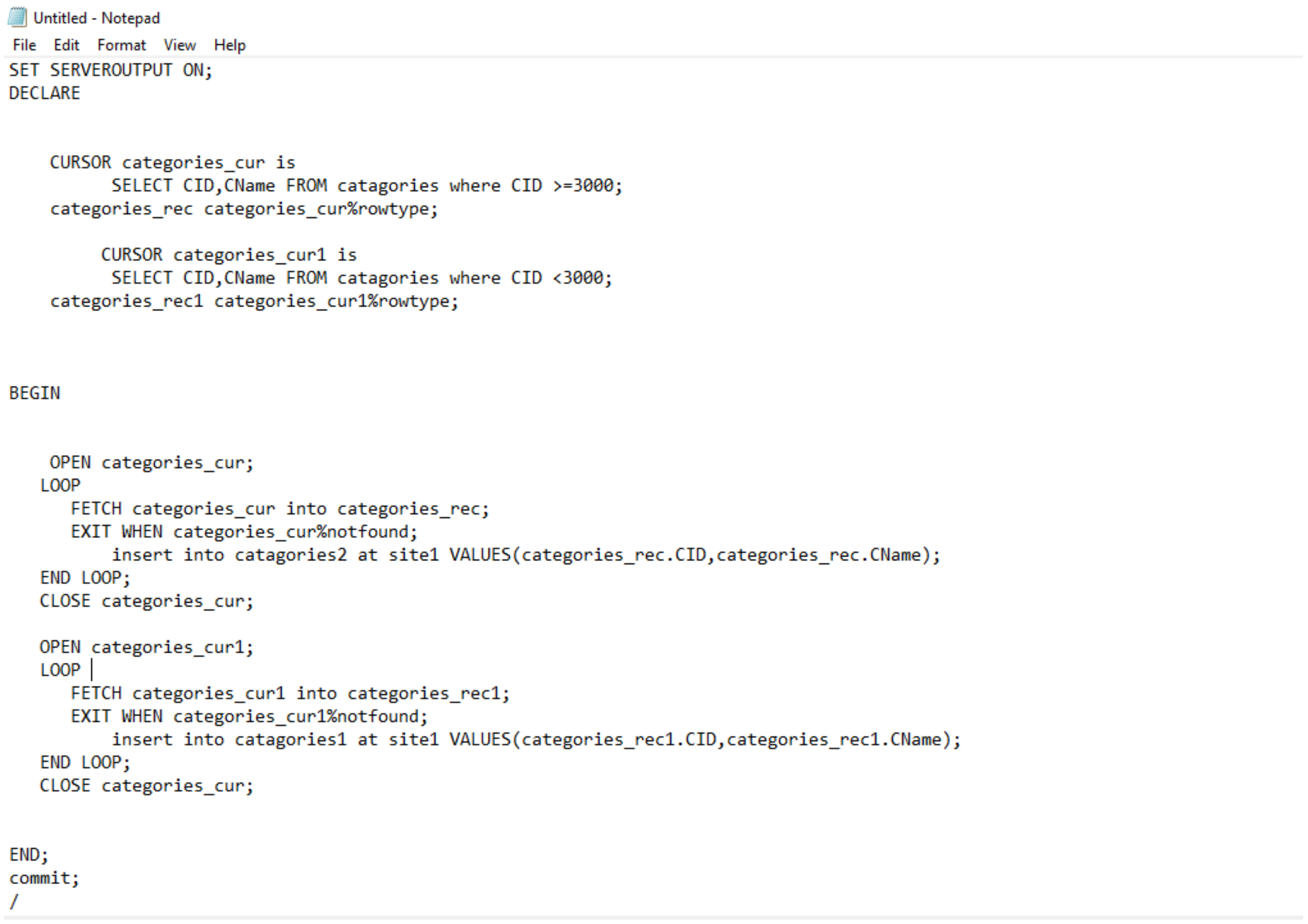


```sql
DROP SEQUENCE seqEployeeSiteDhanmondi;
DROP SEQUENCE seqCatagoriesSiteDhanmondi;
DROP SEQUENCE seqMenuitemsSiteDhanmondi;
DROP SEQUENCE seqCustomerSiteDhanmondi;
DROP SEQUENCE seqReservationSiteDhanmondi;
DROP SEQUENCE seqDiningTablesSiteDhanmondi;
DROP SEQUENCE seqOrdersSiteDhanmondi;
DROP SEQUENCE seqDiningTableTrackSiteDhanmondi;
DROP SEQUENCE seqBillsSiteDhanmondi;

DROP SEQUENCE seqEployeeSiteBailyRoad;
DROP SEQUENCE seqCatagoriesSiteBailyRoad;
DROP SEQUENCE seqMenuitemsSiteBailyRoad;
DROP SEQUENCE seqCustomerSiteBailyRoad;
DROP SEQUENCE seqReservationSiteBailyRoad;
DROP SEQUENCE seqDiningTablesSiteBailyRoad;
DROP SEQUENCE seqOrdersSiteBailyRoad;
DROP SEQUENCE seqDiningTableTrackSiteBailyRoad;
DROP SEQUENCE seqBillsSiteBailyRoad;

CREATE SEQUENCE  seqEployeeSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqCatagoriesSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqMenuitemsSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqCustomerSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqReservationSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqDiningTablesSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqOrdersSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqDiningTableTrackSiteDhanmondi START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE  seqBillsSiteDhanmondi START WITH 1000 INCREMENT BY 1;


CREATE SEQUENCE  seqEployeeSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqCatagoriesSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqMenuitemsSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqCustomerSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqReservationSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqDiningTablesSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqOrdersSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqDiningTableTrackSiteBailyRoad START WITH 3000 INCREMENT BY 1;
CREATE SEQUENCE  seqBillsSiteBailyRoad START WITH 3000 INCREMENT BY 1;
commit;
```

3.Fragmentation:

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. **Fragmentation** can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical).

Here I use horizontal fragmentation with level 3 transparency.

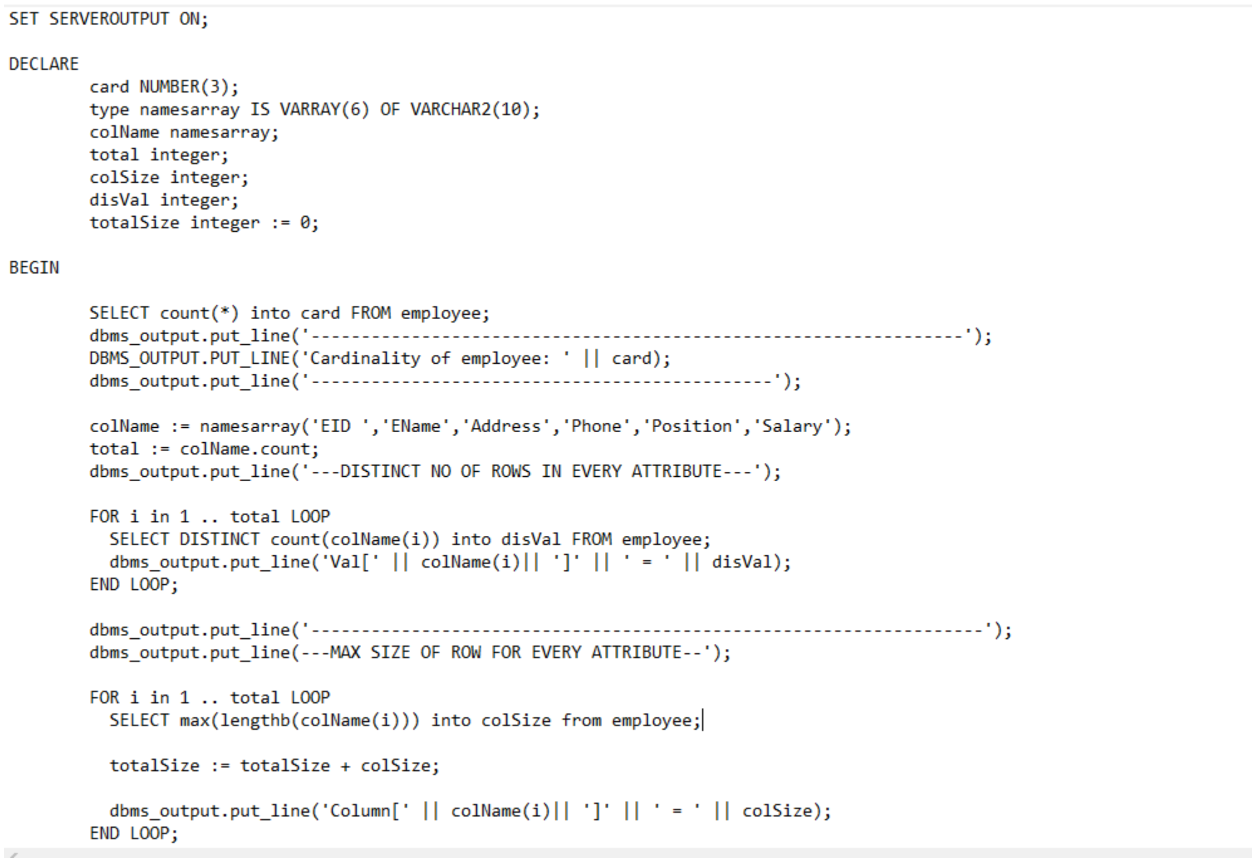


```
SET SERVEROUTPUT ON;
DECLARE


    CURSOR categories_cur is
          SELECT CID,CName FROM catagories where CID >=3000;
    categories_rec categories_cur%rowtype;

         CURSOR categories_cur1 is
          SELECT CID,CName FROM catagories where CID <3000;
    categories_rec1 categories_cur1%rowtype;



BEGIN


    OPEN categories_cur;
   LOOP
      FETCH categories_cur into categories_rec;
      EXIT WHEN categories_cur%notfound;
          insert into catagories2 at site1 VALUES(categories_rec.CID,categories_rec.CName);
   END LOOP;
   CLOSE categories_cur;

   OPEN categories_cur1;
   LOOP
      FETCH categories_cur1 into categories_rec1;
      EXIT WHEN categories_cur1%notfound;
          insert into catagories1 at site1 VALUES(categories_rec1.CID,categories_rec1.CName);
   END LOOP;
   CLOSE categories_cur;


END;
commit;
/
```

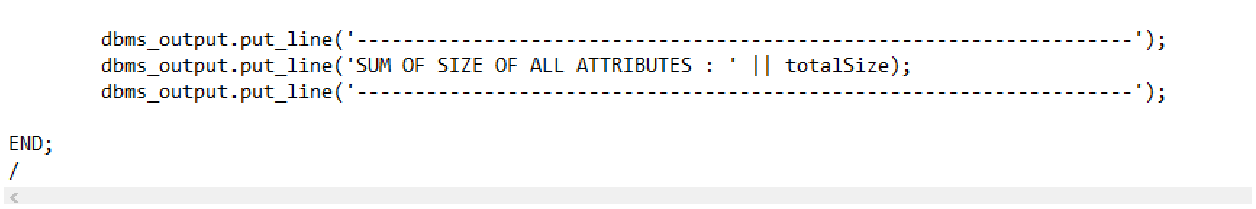
4.Database Profile :

Database profiles represent the data set to be retrieved from or written to a relational database. Data sets contain the SQL code used to select, insert, update, or delete records.

CODE:



```
SET SERVEROUTPUT ON;

DECLARE
        card NUMBER(3);
        type namesarray IS VARRAY(6) OF VARCHAR2(10);
        colName namesarray;
        total integer;
        colSize integer;
        disVal integer;
        totalSize integer := 0;

BEGIN

        SELECT count(*) into card FROM employee;
        dbms_output.put_line('-----------------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('Cardinality of employee: ' || card);
        dbms_output.put_line('-----------------------------------------------');

        colName := namesarray('EID ','EName','Address','Phone','Position','Salary');
        total := colName.count;
        dbms_output.put_line('---DISTINCT NO OF ROWS IN EVERY ATTRIBUTE---');

        FOR i in 1 .. total LOOP
          SELECT DISTINCT count(colName(i)) into disVal FROM employee;
          dbms_output.put_line('Val[' || colName(i)|| ']' || ' = ' || disVal);
        END LOOP;

        dbms_output.put_line('-------------------------------------------------------------------');
        dbms_output.put_line(---MAX SIZE OF ROW FOR EVERY ATTRIBUTE--');

        FOR i in 1 .. total LOOP
          SELECT max(lengthb(colName(i))) into colSize from employee;

          totalSize := totalSize + colSize;

          dbms_output.put_line('Column[' || colName(i)|| ']' || ' = ' || colSize);
        END LOOP;
```

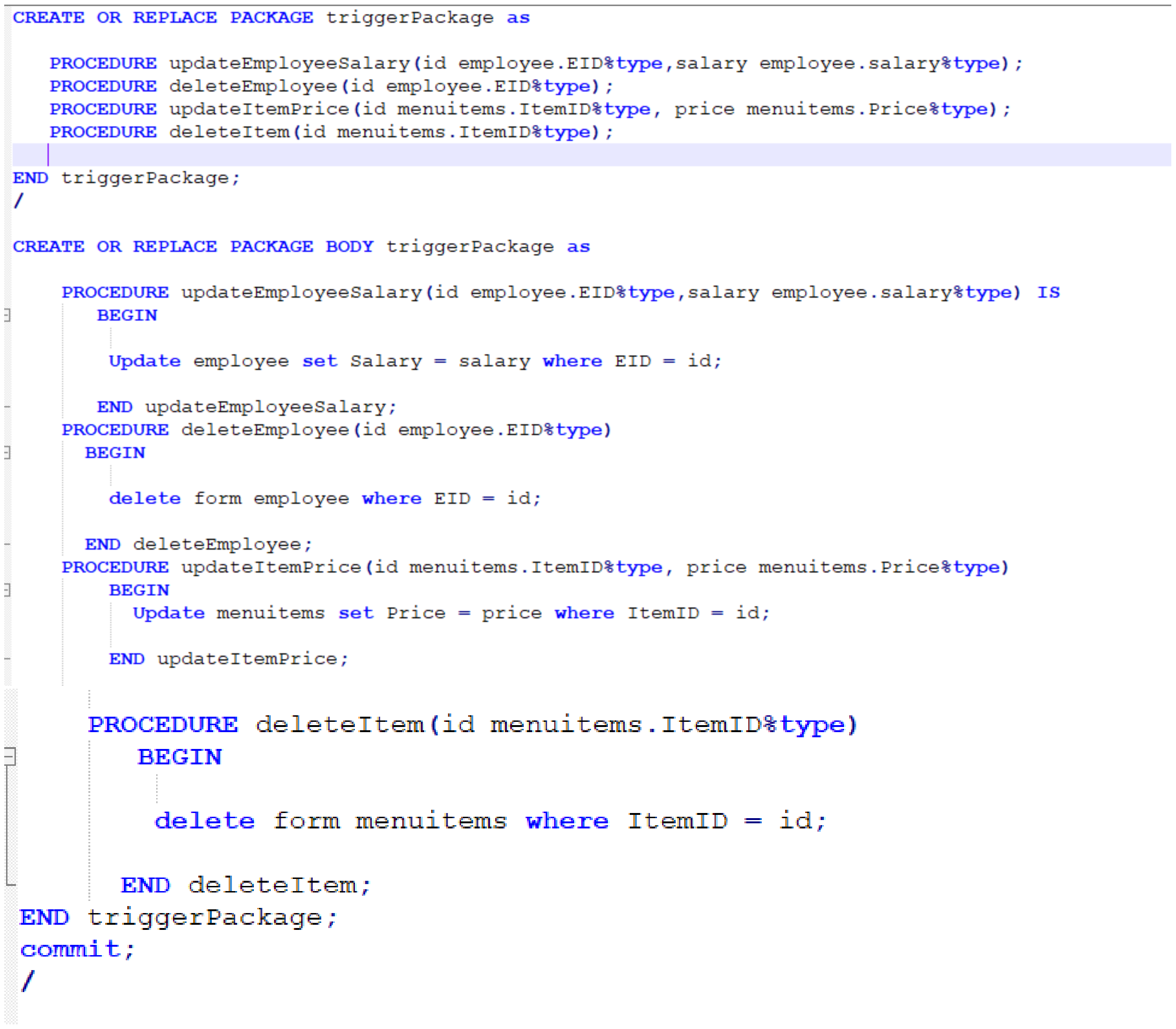


```
        dbms_output.put_line('-------------------------------------------------------------------');
        dbms_output.put_line('SUM OF SIZE OF ALL ATTRIBUTES : ' || totalSize);
        dbms_output.put_line('-------------------------------------------------------------------');

END;
/
```

5.Trigger procedure package:

This package holds all the procedure of two trigger that track while an employee or food item information updated.

CODE :



```
CREATE OR REPLACE PACKAGE triggerPackage as

   PROCEDURE updateEmployeeSalary(id employee.EID%type,salary employee.salary%type);
   PROCEDURE deleteEmployee(id employee.EID%type);
   PROCEDURE updateItemPrice(id menuitems.ItemID%type, price menuitems.Price%type);
   PROCEDURE deleteItem(id menuitems.ItemID%type);

END triggerPackage;
/

CREATE OR REPLACE PACKAGE BODY triggerPackage as

    PROCEDURE updateEmployeeSalary(id employee.EID%type,salary employee.salary%type) IS
       BEGIN

        Update employee set Salary = salary where EID = id;

       END updateEmployeeSalary;
    PROCEDURE deleteEmployee(id employee.EID%type)
      BEGIN

        delete form employee where EID = id;

      END deleteEmployee;
    PROCEDURE updateItemPrice(id menuitems.ItemID%type, price menuitems.Price%type)
        BEGIN
          Update menuitems set Price = price where ItemID = id;

        END updateItemPrice;

      PROCEDURE deleteItem(id menuitems.ItemID%type)
         BEGIN

          delete form menuitems where ItemID = id;

        END deleteItem;
END triggerPackage;
commit;
/
```

Conclusion:

I want work further and extend the project with new features. This was a great experience working on this project and I would like to thank the teachers for making the subject very clear to us and also for helping whenever we approached them.