

RESTAURENT MANAGEMENT SYSTEM

AUST CSE 4126 Spring 2018 - Project Report

Roll: 15.01.04.080(Prattasa karim)

15.01.04.070, 15.01.04.066

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:

Global Relations:

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)

Fragmentation Schema:

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_{RID}>3000(reservation)

diningTables1=SL_{DID}<3000(diningTables)

diningTables2=SL_{DID}>3000(diningTables)

diningTableTrack1=SL_{Serial}<3000(diningTableTrack)

diningTableTrack2=SL_{Serial}>3000(diningTableTrack)

bills1=SL_{BID}<3000(bills)

bills2=SL_{BID}>3000(bills)

Allocation Schema:

There are two sites.

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

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

Contribution:

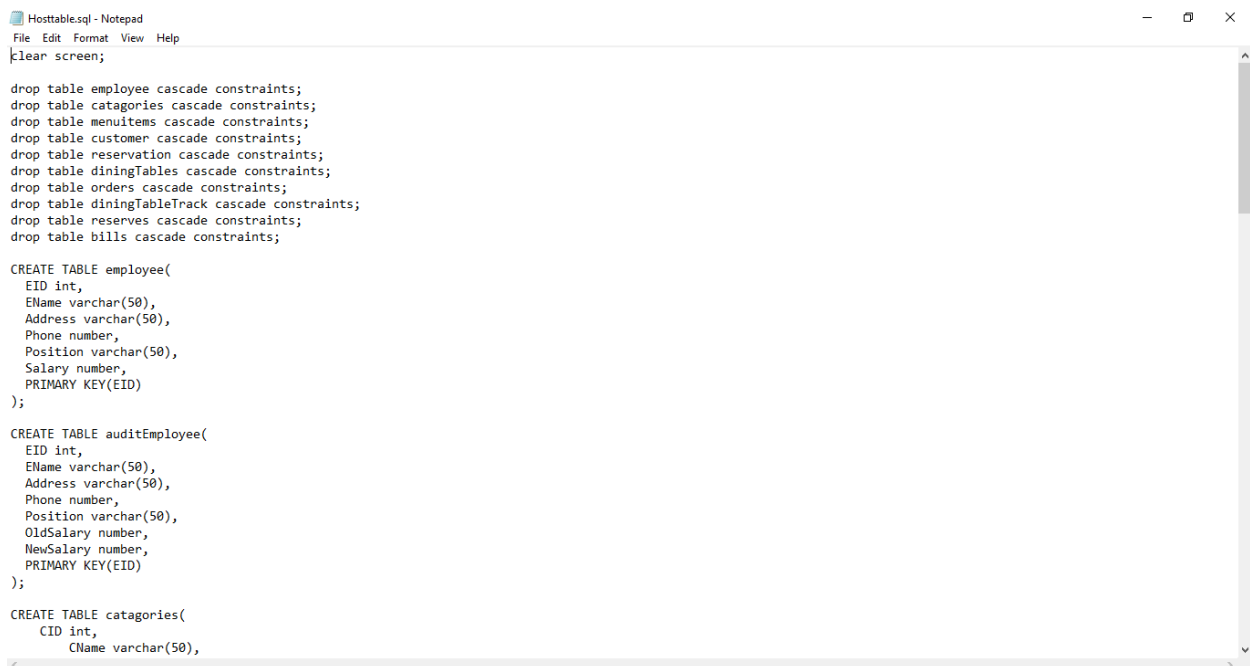
My contributions in this project are to implement 'Connection with host', 'creating database', 'Procedure', 'Fragmentation', 'Trigger'.

1. Connection with host:

This part of my work was connecting with the host.

2. Creating database:

This part of my work was creating database and tables.



```
Hosttable.sql - Notepad
File Edit Format View Help
|clear screen;

drop table employee cascade constraints;
drop table categories cascade constraints;
drop table menuitems cascade constraints;
drop table customer cascade constraints;
drop table reservation cascade constraints;
drop table diningTables cascade constraints;
drop table orders cascade constraints;
drop table diningTableTrack cascade constraints;
drop table reserves cascade constraints;
drop table bills cascade constraints;

CREATE TABLE employee(
  EID int,
  EName varchar(50),
  Address varchar(50),
  Phone number,
  Position varchar(50),
  Salary number,
  PRIMARY KEY(EID)
);

CREATE TABLE auditEmployee(
  EID int,
  EName varchar(50),
  Address varchar(50),
  Phone number,
  Position varchar(50),
  OldSalary number,
  NewSalary number,
  PRIMARY KEY(EID)
);

CREATE TABLE catagories(
  CID int,
  CName varchar(50),
```

Fig:1

3. Procedure:

Pro2.sql defines the reservation of the selected date.



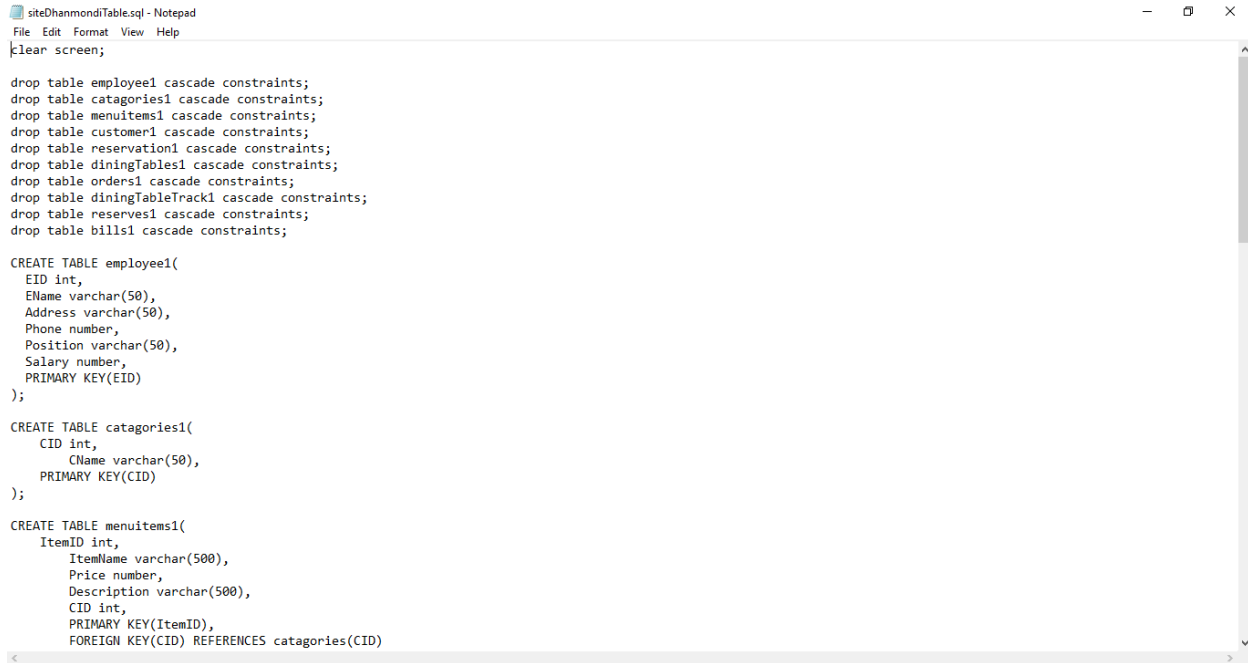
```
pro2.sql - Notepad
File Edit Format View Help
create or replace procedure reservationCheck(givendate in date)
is
    id reservation.RID%Type;
    Rtime reservation.RTime%Type;
    duration reservation.duration%Type;
    cursor R_cur is SELECT RID,duration,RTime From reservation WHERE RDate = givendate;

BEGIN
    open R_cur;
    loop
        fetch R_cur into id,duration,Rtime;
        exit when R_cur%notfound;
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(id)||' - '||TO_CHAR(givendate)||' - '||TO_CHAR(Rtime) ||' - '||TO_CHAR(duration));
    end loop;
    close R_cur;
end;
/
```

Fig:2

4. Fragmentation:

siteDhanmondiTable.sql also called site 1 which has all the fragmented tables from main database.



```
File Edit Format View Help
clear screen;

drop table employee1 cascade constraints;
drop table catagories1 cascade constraints;
drop table menuitems1 cascade constraints;
drop table customer1 cascade constraints;
drop table reservation1 cascade constraints;
drop table diningTables1 cascade constraints;
drop table orders1 cascade constraints;
drop table diningTableTrack1 cascade constraints;
drop table reserves1 cascade constraints;
drop table bills1 cascade constraints;

CREATE TABLE employee1(
  EID int,
  EName varchar(50),
  Address varchar(50),
  Phone number,
  Position varchar(50),
  Salary number,
  PRIMARY KEY(EID)
);

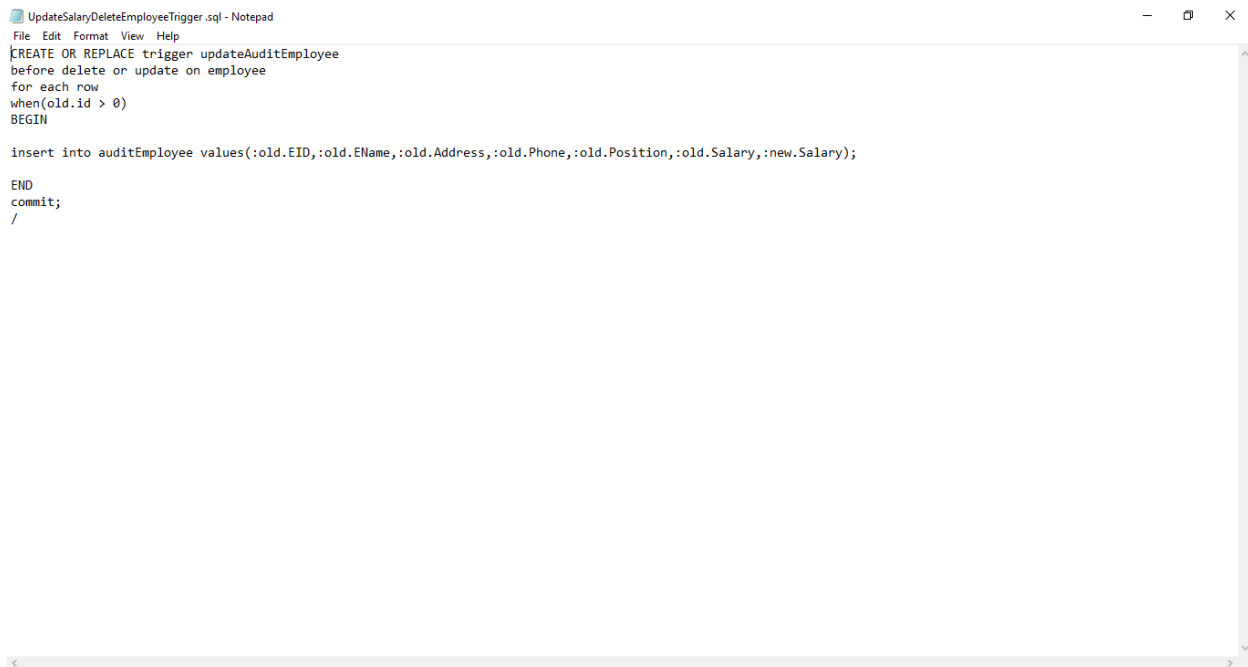
CREATE TABLE catagories1(
  CID int,
  CName varchar(50),
  PRIMARY KEY(CID)
);

CREATE TABLE menuitems1(
  ItemID int,
  ItemName varchar(500),
  Price number,
  Description varchar(500),
  CID int,
  PRIMARY KEY(ItemID),
  FOREIGN KEY(CID) REFERENCES catagories(CID)
);
```

Fig:3

5. Trigger:

UpdateSalaryDeleteEmployeeTrigger.sql which has all the data of previous salary and updated salary and also previous working employee and current working employee.



```
File Edit Format View Help
CREATE OR REPLACE trigger updateAuditEmployee
before delete or update on employee
for each row
when(old.id > 0)
BEGIN

insert into auditEmployee values(:old.EID,:old.EName,:old.Address,:old.Phone,:old.Position,:old.Salary,:new.Salary);

END
commit;
/
```

Fig:4

