HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

BBM 473 FALL 2019

DATABASE MANAGEMENT SYSTEMS LABORATORY

ADVISOR: Arş. Gör. Merve Özdeş, Arş. Gör. Nebi Yılmaz

HOTEL RESERVATION SYSTEM

Phase 2

Stored Procedures, Views, Triggers, and SQL queries

PROJECT MEMBER

21527142 - Doğukan Berat KARATAŞ

21527381 - Meltem TOKGÖZ

21527189 - Yahya KOÇAK

Functionality of Procedures

1- Balance Table Procedures

We do not perform direct transactions in the Balance table. We call these functions in procedures when dealing with users or hotels. Because instead of giving user id when adding balance, we give balance id when adding a user.

- ***addbalance():*** With this method, we are adding balance to the database. The last parameter, balance_id, is a parameter of type “out olan, a special type of MySQL. In other words, while adding person to the system, we also create balance by calling “addBalance ()” function. There is a “balance_id” column in the Person table indicating which balance it has. In order to fill this column, we send the balance of the balance to the place where it is called and save it in the person table.

addbalance(:balance_money, :balance_date, :balance_id (output))

- ***updatebalance():*** With this method, we update the balance in the database.

updatebalance(:balance_id, :balance_money, :balance_date)

- ***deletebalance():*** With this method, we delete the balance in the database.

deletebalance(:balance_id)

2- Hotel Table Procedures

- ***addhotel():*** With this method, we are adding hotels to the database. For this, we first check that there is no such hotel in the database. We do this control according to the “name” column, which we define as unique. If not, we first create balance and then add hotels.

addhotel(:name, :address, :telephone, :hotel_info, :star, :hotel_type)

- ***updatehotel():*** If there is a hotel in the database according to the “hotel_id verilen given by this method, we are updating this hotel information.

updatehotel(:hotel_id, :name, :address, :telephone, :hotel_info, :star, :hotel_type, :balance_money)

- ***deleteHotel():*** With this method, if there is a hotel in the database according to the given “hotel_id”, we delete it.

deletehotel(:hotel_id)

3- Person (Customer, Manager & Employee) Table Procedures

- ***addperson():*** There are 3 types of users in our system: “Customer”, “Manager” and “Employee”. And we have a “Person" table to which all 3 users are connected. Instead of writing individual procedures for these users, we combined the attributes of our person table and the specific properties of each user, we typed it into the addPerson () method, and also set a “person_type” parameter to

determine which user should register. Thus, instead of 3 procedures with almost the same parameters, we have done this with a single procedure. The function first checks whether there is such a person in the database. We do this check according to the "email" and "telephone" in the person table because these columns are unique. If it does not, we process which person type to add according to "person_type" in the case structure. We also form the balance that is linked to it when we form each person.

addperson(:firstname, :lastname, :passwrd, :mail, :address, :phone, :age, :salary, :username, :hotel_name, :person_type)

- ***updateperson():*** With the same logic as in the "addPerson" function, we combine 3 users and perform operations with a single update function. In order to do this process, such a person needs to be in the database. We check this and determine which person. If any, will be updated with the case structure.
-

updateperson(:person_id, :firstname, :lastname, :passwrd, :mail, :address, :phone, :age, :salary, :username, :hotel_name, :person_type)

- ***deleteperson():*** Only with the "person_id " parameter, we delete from the database, if there is such a person. Just deleting it from the person table will also delete it from other tables that reference "person_id".

deleteperson(:person_id)

4- Room Table Procedures

- ***addroom():*** With this method, we are adding rooms to the database. The administrator can add the room with the features he wants by finding the hotel id from the hotel name. So every room is connected to a hotel. When adding the room you must enter information about the room, price, room number, capacity, features, and type of room. Because the customer decides to keep the room and examine this information. Only the manager can perform this operation. Employee is not authorized to do so. There are 2 types of rooms in our system. These are: Corner Room, Suite Room and King Room entering special type rooms. Standard Room and Connection Room are Standard type entering rooms.

  When I add a room, I put the room number in a different process. It is added to the table as "hotel_id - standard(0) / special(1) - room_number". For example, if the hotel id is 1, room number 5 and room type is standard, the room number in the table is called "1-0-5".

addroom(:room_info, :room_price, :room_number, :status, :capacity, :feature, :hotel_name, :room_type)

- ***updateroom():*** If there is such a room in the database with "room_id" parameter, we update the room by entering new information. So if the hotel is undergoing a renovation or change, it can change the room type. Only the manager can do this.

updateperson(:room_id, :room_info, :room_price, :room_number, :status, :capacity, :feature, :hotel_name, :room_type)

- ***deleteroom():*** If there is such a room in the database with "room_id" parameter, we delete it. Only the manager can do this.

deleteroom(:room_id)

5- Rezervation Table Procedures

- ***addreservation():*** Customers registered in the system can make reservations by entering the booking day and departure date and the room number requested. In order to make a reservation, their choice by room must not be in use and must be empty. When making a reservation, the cost of the room they reserve is calculated according to the number of days they will stay and allocated if there is sufficient balance. The room number is not unique because you can have room with the same number in different hotels. So we made a different combination to the room number. We give the room number "Hotel id - standard (0) / special (1) - Room number".

addreservation(:reservation_date, :finish_date, :customer_id, :room_number)

- ***updatereservation():*** It is possible to update the reservation dates according to the reservation id.

updatereservation(:reservation_id, :reservation_date, : finish_date, :price, :customer_id, :room_number)

- ***deletereservation():*** If there is such a reservation in the database according to the reservation id, it can be deleted.

deletereservation(:reservation_id)

6- Organization Table Procedures

- ***addorganization():*** The Manager can add an organization to the hotel by entering the information and price of the hotel according to the organizations he wants to do. In doing so, operations are done according to the hotel_name. Which hotel id is given, the organization is added to that hotel.

addorganization(:name, :org_info, :price, :hotel_name)

- ***updaterganization():*** If there is such an organization in the given organization_id, the manager can update the content and price of the organization.

updateorganization(:organization_id, :name, :org_info, :price)

- ***deleteorganization():*** With the "organization_id" parameter, if there is such an organization, we delete it from the database.

deleteorganization(:organization_id)

7- Rent Organization Table Procedures

- ***addrentorganization() :*** The customer can rent the organization from any hotel by entering the organization's ID. Organization price allocated if there is sufficient customer balance.

addrentorganization(:customer_id, :organization_id)

- ***deleterentorganization():*** The customer can delete the organization he has rented.

deleterentorganization(:customer_id, :organization_id)

8- Extra Service Table Procedures

- ***addextraservice():*** Manager adds the services he wants to add by entering their information and price. Extra services include: Cleaning, transport, wifi, dry cleaning, cargo.

addextraservice(:service, :service_price, :service_point, :room_number, :service_id (output))

- ***updateextraservice():*** If there is such an extra service according to the given service_id, the administrator can update the content and price of the service.

updateextraservice(:service_id, :service, :service_price, :service_point, :room_number)

- ***deleteextraservice():*** If there is such an extra service in the database with the “service_id” parameter, we delete it.

deleteextraservice(:service_id)

9- Extra Service Table Procedures

- ***addroom_extraservice():*** Employee adds the services requested to be added to the room according to the customer's request by entering their information and price. In doing so, operations are performed according to room_id. Which "room_id" is given, extra service is added to that room. At the same time, customer_id is entered and the service's charge is deducted from the customer's account.

add_rentroom_extraservice(:service_id, :room_id)

- ***deleteroom_extraservice():*** With the "service_id ”and“ room_id ”parameters, if there is such an extra service in the given room in the database, we delete it.

delete_rentroom_extraservice(:service_id, :room_id)

10- Food Service Table Procedures

- ***addfoodservice():*** Employee adds food services to the room according to the customer's request by entering their information and price. In doing so, operations are performed according to room_number. Which room_number is given, food service is added to that room. Food services include: Breakfast, dinner, lunch, brunch, open buffet, dessert service, Indian cuisine, traditional Turkish cuisine, Italian cuisine, Far Eastern cuisine. Only special rooms can use this service.

addfoodservice(:service, :service_price, :service_point, :food_detail, :room_number)

- ***updatefoodservice():*** If there is such a food service in the given room_number, the administrator can update the content and price of the service.

updatefoodservice(:food_id, :service, :service_price, :service_point, :food_detail, :room_number)

- ***deletefoodservice():*** If there is such a food service in the database with "food_id" parameter, we delete it.

deletefoodservice(:food_id)

Functionality of Views

- ***customer_view:*** Keeps the customer's information from the person table using the person id. So it is like a person table that includes only customers.
  Columns of view: `p.id,p.firstname,p.lastname,c.username,p.passwrd,c.id as customer_id`
  `p:person c:customer`

- ***employee_view:*** Keeps the employee's information from the person table using the person id. So it is like a person table that includes only employee.
  Columns of view: `p.id, p.firstname, p.Lastname, e.hotel_id, p.balance_id, p.passwrd`
  `p:person e:employee`

- ***manager_view:*** Keeps the manager's information from the person table using the person id. So it is like a person table that includes only manager.
  Columns of view: `p.id,p.firstname,p.lastname,p.balance_id,p.passwrd`

- ***organization_view:*** This view includes organizations' information.
  Columns of view: `hotel_id,name,price`

- ***person_balance:*** It is a view containing the balance status of persons and persons information.
  Columns of view: `p.id, p.firstname, p.lastname, p.passwrd, b.money, b.balance_date`
  `p:person b:balance`

- ***hotel_view:*** This view incluedes hotels' information.
  Columns of view: `id,name,address,star,telephone`

- ***customer_reservation:*** This view includes customers' booking information.
  Columns of view: `cv.id as customer_id, cv.firstname, cv.lastname, cv.username, r.id as reservation_id, r.price, r.start_date, r.finish_date`
  `cv:customer_view r:reservation`

- ***reservation_view:*** This view includes details of the rooms booked.
  Columns of view: `r.id asroom_id, r.hotel_id, r.room_price, re.customer_id, re.id, re.start_date, re.finish_date`
  `r:room re:reservation`

Revisions in Our Tables

1) Filter relationship table removed. Because filtering is a SQL query to write, it would be unreasonable to tabulate such a relationship. The relationship still exists but is not registered in the system as a table.

2) The Registered Customer table has been removed. Of course, there are 2 customers registered and unregistered in the application. But since we don't know the information about unregistered customers, we can only keep registered customers in the database. That's why the Customer table holds registered customers.

3) Rent_Organization table added. An intermediate table was added because there was a many to many relationships between the Customer and the Organization tables.

4) Room_ExtraService table added. An intermediate table was added because there was a many to many relationships between the Room and the ExtraService tables.

5) Room_Reservation table added. An intermediate table was added because there was a many to many relationships between the Room and the Reservation tables.