Wolf Inns Database System

For Wolf Inns Hotel Chain

CSC 540: Database Management Concepts and Systems

Project Report #3

Project Team # 7:
Sachin Kumar
Shubhankar Reddy Katta
Shyam Prasad Katta
Qaiss Khan Alokozai

1) Assumptions with Transactions

i) Checkin /Assign room

Here is the pseudocode of transaction with following chain of events are done in order :-

{Start Transaction with autocommit false

Set Transaction as TRANSACTION_READ_UNCOMMITTED

Create Reservation followed by :
    Inserting payment details
    Inserting checkin details

Update assigned room availability to Occupied

Transaction commit
}
If Exception
Then Rollback
Set autocommit true

Code:-

```
package com.wolfinn.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
import java.util.List;
import java.util.Scanner;

import com.wolfinn.Connection.ConnectionManager;
import com.wolfinn.beans.RoomBean;
import com.wolfinn.beans.RoomClass;
import com.wolfinn.model.RoomsInfo;

public class RoomDAO {

    private Connection conn = null;
    private PreparedStatement ps = null;
    private Statement stmt = null;
    private ResultSet rs = null;

public boolean assignRoom(int customerid,int roomno,int hotelId,int staffId,String
payType,String billingAddress,String cccDetails){

          try {
                    conn = ConnectionManager.getConnection();
 //Set transaction isolation level  as read uncommited to read uncommited reservationid
and billingid
conn.setTransactionIsolation(conn.TRANSACTION_READ_UNCOMMITTED);

//setting autocommit as false
          conn.setAutoCommit(false);
                    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                    Date date = new Date();
                    String checkintime=dateFormat.format(date);
          // inserting reservation details
                    ps = conn.prepareStatement("INSERT INTO Reservation (Hotel_id,
Checkin_time, Checkout_time) VALUES (?,?,?)");
                    ps.setInt(1,hotelId );
                    ps.setString(2, checkintime);
                    ps.setString(3,"0000-00-00 00:00:00");
                    ps.executeUpdate();
                    ps.close();
```

```
                    ps = conn.prepareStatement("select max(reservation_id) as
reservation_id from Reservation;");
                    rs = ps.executeQuery();
                    int reservationId=0;
                    if(rs.next())
                    {
                      reservationId=rs.getInt("reservation_id");
                    }
                    ps.close();
                   rs.close();
 // initializing billing record
                   ps = conn.prepareStatement("INSERT INTO billing_info (amount)
VALUES ('0')");
                    ps.executeUpdate();
                    ps.close();
//Getting billingid of this record
                    ps = conn.prepareStatement("select max(billing_id) as billing_id from
Billing_info;");
                    rs = ps.executeQuery();
                    int billingId=0;
                    if(rs.next())
                    {
                     billingId=rs.getInt("billing_id");
                    }
                    ps.close();
 //inserting checkin records
                    ps = conn.prepareStatement("INSERT INTO check_in_info (Hotel_id,
Staff_id, Customer_id,Room_no,Reservation_id,Billing_id) "
                                    + "VALUES (?,?,?,?,?,?)");
                    ps.setInt(1,hotelId );
                    ps.setInt(2,staffId );
                    ps.setInt(3,customerid );
                    ps.setInt(4,roomno );
                    ps.setInt(5,reservationId );
                    ps.setInt(6,billingId );
                    ps.executeUpdate();
                    ps.close();
 //inserting payment details
```

```
                ps = conn.prepareStatement("INSERT INTO payment_details
(Payment_type, Billing_address, Ccc_details) "
                                + "VALUES (?,?,?)");
                ps.setString(1,payType );
                ps.setString(2,billingAddress );
                ps.setString(3,cccDetails );
                ps.executeUpdate();
                ps.close();
  //getting payment id of this transaction
                ps = conn.prepareStatement("select max(payment_id) as payment_id
from payment_details;");
                rs = ps.executeQuery();
                int paymentid=0;
                if(rs.next())
                {
                        paymentid=rs.getInt("payment_id");
                }
                ps.close();
  // inserting paymentid and billinid relation
                ps = conn.prepareStatement("INSERT INTO payment_relation
(Payment_id, Billing_id) "
                                + "VALUES (?,?)");
                ps.setInt(1,paymentid );
                ps.setInt(2,billingId );
                ps.executeUpdate();
                ps.close();
  //updating room availability to occupied
                ps = conn.prepareStatement("UPDATE Room SET Availability = ?
WHERE Room_no = ? AND Hotel_id = ?");
                ps.setString(1, "Occupied");
                ps.setInt(2, roomno);
                ps.setInt(3, hotelId);
                ps.executeUpdate();
                ps.close();

   // Committing transaction
        conn.commit();

         } catch (SQLException e) {
```

```
                System.out.println("Room assignment failed,check your input
parameters");
                if (conn != null) {
                        try {
//Rolling back transaction
                                conn.rollback();
                               // setting back autocommit to true
                                conn.setAutoCommit(true);
                                } catch (SQLException e1) {

                                e.printStackTrace();
                                }       }
        }
        catch (Exception e) {
                e.printStackTrace();
                System.out.println("Exception");
        }
        finally{
                //finally block used to close resources
                try{
                if(stmt!=null)
                conn.close();
                }catch(SQLException se){
                }// do nothing
                try{
                if(conn!=null)
                conn.close();
                }catch(SQLException se){
                se.printStackTrace();
                }//end finally try
                }
        return false;
   }
```

2) Checkout /Generate Bill

Here is the pseudocode of transaction with following chain of events are done in order :-

Getbilldetails
{

{
Start Transaction with autocommit false

Get checkin time

Update reservation set checkintime and checkout time(current time)

Update room availability to Available

Get payment type

Get final bill amount

Update total bill amount in database for record purpose

Print bill

Transaction commit

}
If Exception
Then Rollback
Set autocommit true
}

Code:-

```
package com.wolfinn.model;
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

import com.wolfinn.Connection.ConnectionManager;
import com.wolfinn.beans.BillingDetailsBean;

public class BillingAccounts {

   public void getBillDetails(int hotelId)
   {
         Connection conn=null;
         Statement stmt=null;
         System.out.println("Enter reservation id \n");
         Scanner scan= new Scanner(System.in);
         int reservation_id= scan.nextInt();

         System.out.println("Enter room number \n");
         int roomno= scan.nextInt();

         try {

                  conn = ConnectionManager.getConnection();
  // getting checkin_time
                  PreparedStatement ps5 = conn.prepareStatement("select checkin_time
from reservation where reservation_id= ?");
                  ps5.setInt(1,reservation_id);
                  ResultSet rs3 = ps5.executeQuery();
                  String checkintime="";
                  if(rs3.next())
                  {
```

```
                checkintime=rs3.getString("checkin_time");
              }
              ps5.close();
              rs3.close();


        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date date = new Date();
        String checkouttime=dateFormat.format(date);


        //setting autocommit as false
        conn.setAutoCommit(false);
  //Updating reservation checkintime and checkout time
        PreparedStatement ps2 = conn.prepareStatement("UPDATE reservation SET
checkin_time=?,checkout_time=? WHERE Reservation_id =? AND Hotel_id =?");
        ps2.setString(1, checkintime);
        ps2.setString(2, checkouttime);
        ps2.setInt(3, reservation_id);
        ps2.setInt(4, hotelId);
        ps2.executeUpdate();
        ps2.close();
   //Updating room availability to available
        PreparedStatement ps3 = conn.prepareStatement("UPDATE Room SET
Availability = ? WHERE Room_no = ? AND Hotel_id = ?");
        ps3.setString(1, "Available");
        ps3.setInt(2, roomno);
        ps3.setInt(3, hotelId);
        ps3.executeUpdate();
        ps3.close();
  //Getting billingid of this transaction
        PreparedStatement ps4 = conn.prepareStatement("select billing_id from
check_in_info where reservation_id= ?");
        ps4.setInt(1,reservation_id);
        ResultSet rs2 = ps4.executeQuery();
        int billingId=0;
        if(rs2.next())
        {
         billingId=rs2.getInt("billing_id");
```

```
        }
        ps4.close();

  //Getting paymenttype of this transaction
        String payment_type=getPaymentType(hotelId, reservation_id);

        System.out.println("Stored payment mode is "+payment_type);

//Getting paymentamount

        List<BillingDetailsBean> payamt = getpayAmount(checkouttime,hotelId,
reservation_id,payment_type);

        double amountCharged = payamt.get(payamt.size()-1).getAmountCharged();

// Updating final bill for records
        PreparedStatement ps6 = conn.prepareStatement("UPDATE billing_info SET
amount = ? WHERE billing_id = ? ");
        ps6.setDouble(1, amountCharged);
        ps6.setInt(2, billingId);
        ps6.executeUpdate();
        ps6.close();



  // Committing transaction

        conn.commit();

        System.out.format("%32s%32s", "itemName", "amount_charged");
        System.out.printf("%n");
        for(int i=0;i<payamt.size();i++)
        {
                String itemName=payamt.get(i).getItemName();
                double amount_charged=payamt.get(i).getAmountCharged();
                System.out.format("%32s%32.2f", itemName,amount_charged);
                System.out.printf("%n");

        }
```

```
        }
        catch (SQLException e) {
                System.out.println("Room billing failed,check your input parameters");
                if (conn != null) {
                        try {

//Rolling back transaction
                                conn.rollback();
  // setting back autocommit to true
                               conn.setAutoCommit(true);
                        } catch (SQLException e1) {

                                e.printStackTrace();
                        }       }
        }
        catch (Exception e) {
                e.printStackTrace();
                System.out.println("Exception");
        }
        finally{
                //finally block used to close resources
                try{
                if(stmt!=null)
                conn.close();
                }catch(SQLException se){
                }// do nothing
                try{
                if(conn!=null)
                conn.close();
                }catch(SQLException se){
                se.printStackTrace();
                }//end finally try
                }

   }

   public String getPaymentType(int hotelid,int reservation_id)
   {
```

```
        String sqlQuery="select payment_type from Payment_details pd join
Payment_relation pr on pd.Payment_id=pr.Payment_id"
                        + " join Check_in_info cii on cii.Billing_id=pr.Billing_id where
reservation_id="+reservation_id+" and cii.Hotel_id="+hotelid+";" ;

        System.out.println(sqlQuery);
        String payment_type="cash";
        try{
                Connection conn1 = ConnectionManager.getConnection();
                Statement stmt = conn1.createStatement();
                ResultSet rs=stmt.executeQuery(sqlQuery);
        while(rs.next()){
                        payment_type=rs.getString("payment_type");
                                        }
                                rs.close();
                }
        catch(SQLException se){
                //Handle errors for JDBC
                se.printStackTrace();
                }catch(Exception e){
                //Handle errors for Class.forName
                e.printStackTrace();
                }
        return payment_type;
   }

   public List<BillingDetailsBean> getpayAmount(String checkouttime,int hotelid,int
reservation_id,String paymentType)
   {
        String roomChargesQuery="select 'Room_charges' as  'itemName'
,(TIMESTAMPDIFF(HOUR,checkin_time,'"+checkouttime+"')/24) * price as
'amount_charged' from Reservation r join "
                        + "Check_in_info cii on r.reservation_id=cii.reservation_id join
Room ro on cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join "
                        + "Rooms_price_listing rpl on h.city=rpl.city and
rpl.category=ro.category and r.reservation_id="+reservation_id+";" ;

        String serviceChargesQuery="select service_name as 'itemName',
sum(service_price) as 'amount_charged' from Services s join "
```

```
                                        + "    Services_used su on
s.service_id=su.service_id join Reservation r on r.reservation_id=su.reservation_id "
                                        + "where r.reservation_id="+reservation_id+"
group by service_name;";
        List<BillingDetailsBean> billDetails=new ArrayList<BillingDetailsBean>();
        System.out.println(roomChargesQuery);

        try{
                Connection conn1 = ConnectionManager.getConnection();
                Statement stmt = conn1.createStatement();

                ResultSet rs2=stmt.executeQuery(serviceChargesQuery);
        double totalServiceCharges=0.0;
        double roomCharges=0.0;

        while(rs2.next()){
                                        String itemName=rs2.getString("itemName");
                                        double
amountCharged=Double.parseDouble(rs2.getString("amount_charged"));
                        BillingDetailsBean billingBean=new BillingDetailsBean();
                        billingBean.setItemName(itemName);
                        billingBean.setAmountCharged(amountCharged);
                        billDetails.add(billingBean);
                        totalServiceCharges+=amountCharged;
                        }
                                        rs2.close();
        ResultSet rs=stmt.executeQuery(roomChargesQuery);
        while(rs.next()){
                                        String itemName=rs.getString("itemName");
                                        double
amountCharged=Double.parseDouble(rs.getString("amount_charged"));
                                        roomCharges=amountCharged;
                        BillingDetailsBean billingBean=new BillingDetailsBean();
                        billingBean.setItemName(itemName);
                        billingBean.setAmountCharged(amountCharged);
                        billDetails.add(billingBean);
                                        }
                                rs.close();
```

```
                        double discount=0;
                        BillingDetailsBean billingBean=new BillingDetailsBean();
                        double total_charges=0.0;
                        if(paymentType.equals("hotel credit"))
                        {
                                discount=0.05*roomCharges;
                                billingBean.setItemName("Discount");
                               billingBean.setAmountCharged(discount);
                               billDetails.add(billingBean);
                                total_charges=totalServiceCharges+roomCharges-discount;

                        }else
                        {
                                total_charges=totalServiceCharges+roomCharges;
                        }
                        BillingDetailsBean billingBean2=new BillingDetailsBean();
                        billingBean2.setItemName("Total payable");
                       billingBean2.setAmountCharged(total_charges);
                        billDetails.add(billingBean2)    ;
                }
        catch(SQLException se){
                //Handle errors for JDBC
                se.printStackTrace();
                }catch(Exception e){
                //Handle errors for Class.forName
                e.printStackTrace();
                }
        return billDetails;
   }


}
```

2) High level decisions

The mechanical approach was used to create the global schema with a few exceptions.

1. Each entity set was made into a relation with the same set of attributes
2. Relationships were replaced by a relation whose attributes are the keys for the connected entity sets

The E/R viewpoint was used to convert the subclasses into relations. This method was used so:

1. The system can differentiate between manager, front-desk representatives, and service staff.
2. All people including manager, front-desk representatives, and service staff can be referenced from one single table (Staff).

Many-to-one relationships were combined with other relations. Combining relations in this way makes it more efficient to answer queries that involve attributes of one relation than to answer queries involving attributes of several relations.

Following assumptions are made:

1. Executive Manager is administrator of all the hotels belonging to WolfInn chain of hotels.
2. There is only one Executive Manager for WolfInn chain of hotels.
3. We will be having a global Services table with services description defined for service id used in the project.
4. Each hotel has only one manager.
5. Manager is uniquely identified by manager_staff_id provided in hotel entity.
6. For all the local ER diagrams except that of Executive Manager, the diagram is represented for single hotel only.
7. Reports are not shown in the ER diagrams as we will be generating reports by joins across multiple tables, so since there is no dedicated table associated with it, hence it is not worthwhile showing it on ER diagram.
8. In every local ER diagram, we are just showing functionality applicable to that user and other detailed functionalities are shown in respective local diagrams.
9. Manager is the administrator of a particular hotel records, to which he is associated with.
10. Every relation will have a table associated with it storing keys of entities to the relations.
11. A master list of Services offered by a particular hotel and their associated prices is created.
12. All Services are offered only during the office hours and the customer has to explicitly avail the service by requesting the associated service staff personnel.
13. Dedicated Staff means that staff is assigned to a reservation for its entire duration of existence, with that particular staff is unable to serve other

reservations for that duration. However non-dedicated staff can serve more than one reservations.

14. Service charge is applicable to each time the service is availed. The same charge will be reflected in a particular customer reservation.So for every instance of service offered there will be a separate row inserted for that offering.
15. All the individual Services rates is assumed to be uniform across all the hotels of WolfInn chain. Each hotel which provides a service has the same service charges as that of other hotels in WolfInn chain, is our assumption.
16. A flag named “Serving_premium” column will be maintained in service staff table to keep track of people who’re assigned to the presidential suite and stores the reservation id as long as they’re serving that particular suite/reservation.
17. All the unique identifiers of different tables like reservationid,paymentid,billingid,customerid and staffid are generated by auto increment functionality.
18. Prices vary by location and type of services offered. We maintain two global tables price_by_location and price_by_room_class.
19. A room unit prices is calculated by summation of price by its location and class of the services of a room.
20. Executive manager can add entries in the prices tables by region and category. In case of price absence we use default prices.
21. It is assumed that these two tables(price_by_location and price_by_room_class) are global tables and are not represented in any ER diagrams.
22. It is assumed that the Front Desk Representative checks if the customer preferred room category is available. If available makes the reservation, if not let the customer know the available room types. Later, makes the reservation accordingly.
23. There are two attributes under staff, namely department and job_title. The distinction between them is that one or more people with different job_title can be present in same department. Detailed examples can be seen in insert statement of staff under SQL sub section(3).
24. When a entry for hotel is inserted then by default manager_staff_id will be 1, which we will be updating with another manager’s staff id as required.

3) Team role in Reports

i) Project Report 1

Team member :	**Role:**
Sachin Kumar	- Database Designer(p), Test Plan Engineer(b)
Shubhankar Reddy Katta	- Application Programmer(p), Software Engineer(b)

Shyam Prasad Katta	-	Test Plan Engineer(p), Application Programmer(b)
Qaiss Khan Alokozai	-	Software Engineer(p), Database Designer(b)

ii) Project Report 2

Team member :	**Role:**	
Sachin Kumar	-	Software Engineer(p), Application Programmer(b)
Shubhankar Reddy Katta	-	Database Designer(p), Test Plan Engineer(b)
Shyam Prasad Katta	-	Application Programmer(p), Database Designer(b)
Qaiss Khan Alokozai	-	Test Plan Engineer(p), Software Engineer(b)

iii) Project Report 3

Team member :	**Role:**	
Sachin Kumar	-	Application Programmer(p), Software Engineer(b)
Shubhankar Reddy Katta	-	Database Designer(p), Application Programmer(b)
Shyam Prasad Katta	-	Software Engineer(p), Test Plan Engineer (b)
Qaiss Khan Alokozai	-	Test Plan Engineer(p), Database Designer(b)

p -----> Primary

b -----> Backup