# CSC440 Project Report 3

## Team Structure

**Group-E**

Tsu-Hsin Yeh [tyeh3]: Database designer/administrator, prime software engineer, prime application programmer, backup test plan engineer

Grayson Jones [gcjones]: Project manager, Database designer, backup application programmer, backup software engineer, prime test plan engineer

Julie Persinger [jcpersin]: prime application programmer, backup test plan engineer.

Rodolphe N Sandjong [rnjiwas]: backup application programmer, prime test plan engineer.

## How to Compile/Run

1. add oracle 11
2. export CLASSPATH=".:/afs/bp.ncsu.edu/dist/oracle10g/.install/i386_linux26/OraHome1/jdbc/lib/classes12.jar"
3. alias java="java -cp .:/afs/bp.ncsu.edu/dist/oracle10g/.install/i386_linux26/OraHome1/jdbc/lib/classes12.jar"
4. add jdk142
5. javac *.java
6. java Main

## General Structure

Our application demo runs under a structured menu framework. There is a class takes care of basic user interface and there are several classes implement the desired task and operation. The detailed classes are described below:

1. Main – provides user friendly operation menu for user to select.
2. JDBCConnector – provides interface to access the object connected to Oracle database.
3. InformationProcessing – handles *Information Processing* task and operations specified in the narrative.
4. ServiceRecord – handles *Maintaining Service Availed Records* task and operations specified in the narrative.
5. BillingAccounts – handles Maintaining Billing *Accounts* task and operation specified in the narrative.
6. Reports – handles *Reports* task and operation specified in the narrative.
7. PrintTable – Helper class to print out tuples stored inside important tables.

# Transaction Support

## Transaction support 1: create new room

**Justification:** For creating a new room, we need to modify two tables (room_type and room) in the database. There is a possibility that the first INSERT statement into room_type is successful but the second INSERT statement fails because administrator provides the room number and hotel name combination which already exists in the database (primary key constrain violation). Therefore, a transaction is needed to ensure data integrity in this two tables.

**Code segment (Related part are bold and underlined)**:

```
public void newRoom(){
    boolean printSuccessMsg = true;
    boolean anythingWrong = false;

    try{
      conn.setAutoCommit( false );
    }catch (Exception e){
      anythingWrong = true;
      printSuccessMsg = false;
    }

    // prompt for the information needed
    System.out.println("Creating new room.");
    System.out.print("Room number: ");
    int room_number = input.nextInt();
    input.nextLine(); // discard new line character
    System.out.print("Category: ");
    String category = input.nextLine();
    System.out.print("Occupancy: ");
    int occupancy = input.nextInt();
    input.nextLine(); // discard new line character
    System.out.print("Nightly rate: ");
    int nightly_rate = input.nextInt();
```

```
    input.nextLine(); // discard new line character
    System.out.print("Hotel name: ");
    String hotel_name = input.nextLine();

    // get hotel id according to hotel name
    int hotel_id = -1;
    if(hotel_name.length() != 0){
      try{
        PreparedStatement ps = null;
        ps = conn.prepareStatement("SELECT * FROM hotel WHERE name = '" + hotel_name + "'");
        ResultSet rs = ps.executeQuery();
        if(rs.next()){
          hotel_id = rs.getInt("id");
        }
      } catch(SQLException e){
        e.printStackTrace();
        printSuccessMsg = false;
      }
    }

    // insert new room record
    if(hotel_id != -1){
      // We do get the hotel id provided hotel name
      try{
        PreparedStatement ps = null;
        ps = conn.prepareStatement("INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
(?, ?, ?)");
        ps.setString(1, category);
        ps.setInt(2, occupancy);
        ps.setInt(3, nightly_rate);
        ps.executeUpdate();

      } catch(SQLException e){
        if(e.getErrorCode() != 1){
```

```
        e.printStackTrace();
        printSuccessMsg = false;
      }
    }

    try{
      PreparedStatement ps = null;
      ps = conn.prepareStatement("INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (?, ?, ?, 1, ?)");
      ps.setInt(1, room_number);
      ps.setString(2, category);
      ps.setInt(3, occupancy);
      ps.setInt(4, hotel_id);
      ps.executeUpdate();

    } catch(SQLException e){
      System.out.println("(Room number, Hotel name) combination already exists. Primary key violation!");
      anythingWrong = true;
      printSuccessMsg = false;
    }
  } else {
    System.out.println("Hotel with given name cannot be found.");
    printSuccessMsg = false;
    anythingWrong = true;
  }


  if(anythingWrong){
    try{
      conn.rollback();
      System.out.println("Transaction is rolled back!");
      System.out.println();
    }catch(SQLException e){
```

```
    }
  }else{
    try{
      conn.commit();
      System.out.println("Transaction is commited!");
      System.out.println();
    }catch(SQLException e){

    }
  }

    try{
      conn.setAutoCommit( true );
    }catch (Exception e){

    }

  if(printSuccessMsg){
    System.out.println();
    System.out.println("Room Create successful!");
    System.out.println();
  }
}
```

## Transaction support 2: assign room to customer

**Justification:** For assigning room to the customer, there is a possibility that the front desk staff assigned a room which is already unavailable and the desired occupancy by customer is greater than the maximum occupancy supported by the room. In this case, we need roll back the transaction. In addition, there are three tables (room, checkin and reserve_for) in the database need to be modified for this operation to complete. We need to ensure that if **any** of the UPDATE operation fails, for example SQLException occurred, all of the operation need to roll back to protect data integrity.

### Code segment (Related part are bold and underlined):

```
public void assignRoom(){

    boolean printSuccessMsg = true;

    boolean anythingWrong = false;

    try{
      conn.setAutoCommit( false );
    }catch (Exception e){
      printSuccessMsg = false;
    }

    // prompt for the information needed
    System.out.println("Assign room to customer.");
    System.out.print("Customer name: ");
    String customer_name = input.nextLine();
    System.out.print("Hotel name: ");
    String hotel_name = input.nextLine();
    System.out.print("Room number: ");
    int room_number = input.nextInt();
    input.nextLine(); // discard new line character
    System.out.print("Current occupancy: ");
    int current_occupancy = input.nextInt();
    input.nextLine(); // discard new line character
```

```
System.out.print("Start date (format example: 12-OCT-2016): ");
String start_date = input.nextLine();
System.out.print("End date: ");
String end_date = input.nextLine();
System.out.print("Start time (formate example: 11:11:11): ");
String start_time = input.nextLine();
System.out.print("End time: ");
String end_time = input.nextLine();
System.out.print("Front desk staff name(if any): ");
String fd_name = input.nextLine();
System.out.print("Catering staff assigned name(if any): ");
String cs_name = input.nextLine();
System.out.print("Service staff assigned name(if any): ");
String ss_name = input.nextLine();

// retrieve the hotel id
int hotel_id = -1;
try{
  PreparedStatement ps = null;
  ps = conn.prepareStatement("SELECT * FROM hotel WHERE name = '" + hotel_name + "'");
  ResultSet rs = ps.executeQuery();
  if(rs.next()){
    hotel_id = rs.getInt("id");
  }
} catch(SQLException e){
  e.printStackTrace();
  printSuccessMsg = false;
}

// get the specified room availability
int current_availability = -1;
int max_occupancy = -1;
```

```
try{
  PreparedStatement ps = null;
  ps = conn.prepareStatement("SELECT * from room WHERE(room_number = ? AND hotel_id = ?)");
  ps.setInt(1, room_number);
  ps.setInt(2, hotel_id);
  ResultSet rs = ps.executeQuery();
  if(rs.next()){
    current_availability = rs.getInt("availability");
    max_occupancy = rs.getInt("occupancy");
  }
} catch(SQLException e){
  e.printStackTrace();
  printSuccessMsg = false;
}

if(current_availability == -1){
  // if room specified does not exisits, then roll back
  System.out.println("The room specified does not exist!");
  anythingWrong = true;
  printSuccessMsg = false;
}

if(current_availability == 0){
  // if current availability = 0 (unavailable), then roll back
  System.out.println("The room specified is already occupied!");
  anythingWrong = true;
  printSuccessMsg = false;
}
if(current_occupancy > max_occupancy){
  // if current occupancy > room max occupancy, then roll back
  anythingWrong = true;
```

```
    printSuccessMsg = false;
  }



  // set the corresponding room availability to be 0
  try{
    PreparedStatement ps = null;
    ps = conn.prepareStatement("UPDATE room SET availability = 0 WHERE(room_number = ? AND
hotel_id = ?)");
    ps.setInt(1, room_number);
    ps.setInt(2, hotel_id);
    ps.executeUpdate();
  } catch(SQLException e){
    e.printStackTrace();
    anythingWrong = true;
    printSuccessMsg = false;
  }


  // get service staff id if needed
  int ss_id = -1;
  if(ss_name.length() != 0){
    try{
      PreparedStatement ps = null;
      ps = conn.prepareStatement("SELECT id FROM staff WHERE name = ?");
      ps.setString(1, ss_name);
      ResultSet rs = ps.executeQuery();
      if(rs.next()){
        ss_id = rs.getInt("id");
      }
    } catch(SQLException e){
      e.printStackTrace();
```

```
    printSuccessMsg = false;
  }
}

// get catering staff id if needed
int cs_id = -1;
if(cs_name.length() != 0){
  try{
    PreparedStatement ps = null;
    ps = conn.prepareStatement("SELECT id FROM staff WHERE name = ?");
    ps.setString(1, cs_name);
    ResultSet rs = ps.executeQuery();
    if(rs.next()){
      cs_id = rs.getInt("id");
    }
  } catch(SQLException e){
    e.printStackTrace();
    printSuccessMsg = false;
  }
}

// get customer id
int customer_id = -1;
try{
  PreparedStatement ps = null;
  ps = conn.prepareStatement("SELECT id FROM customer WHERE name = ?");
  ps.setString(1, customer_name);
  ResultSet rs = ps.executeQuery();
  if(rs.next()){
    customer_id = rs.getInt("id");
  }
} catch(SQLException e){
```

```
  e.printStackTrace();
  printSuccessMsg = false;
}

// get front desk id
int fd_id = -1;
if(fd_name.length() != 0){
  try{
    PreparedStatement ps = null;
    ps = conn.prepareStatement("SELECT id FROM staff WHERE name = ?");
    ps.setString(1, fd_name);
    ResultSet rs = ps.executeQuery();
    if(rs.next()){
      fd_id = rs.getInt("id");
    }
  } catch(SQLException e){
    e.printStackTrace();
    printSuccessMsg = false;
  }
}

if(ss_name.length() != 0 && ss_id == -1){
  System.out.println("Service staff with given name cannot be found!");
    anythingWrong = true;
  printSuccessMsg = false;
}

if(cs_name.length() != 0 && cs_id == -1){
  System.out.println("Catering staff with given name cannot be found!");
    anythingWrong = true;
  printSuccessMsg = false;
```

```
    }

    if(fd_name.length() != 0 && fd_id == -1){
      System.out.println("Front desk staff with given name cannot be found!");
      anythingWrong = true;
      printSuccessMsg = false;
    }

    // insert into checkin table
    try{
      PreparedStatement ps = null;
      ps = conn.prepareStatement("INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id) VALUES (checkin_seq.nextval, ?, ?, ?,
?, ?, ?, ?, ?, ?)");
      ps.setInt(1, current_occupancy);
      ps.setString(2, start_date);
      ps.setString(3, end_date);
      ps.setString(4, start_time);
      ps.setString(5, end_time);

      if(cs_id == -1){
        ps.setString(6, null);
      }else{
        ps.setInt(6, cs_id);
      }

      if(ss_id == -1){
        ps.setString(7, null);
      }else{
        ps.setInt(7, ss_id);
      }
      if(fd_id == -1){
```

```
      ps.setString(8, null);
    }else{
      ps.setInt(8, fd_id);
    }
    ps.setInt(9, customer_id);

    ps.executeUpdate();
  } catch(SQLException e){
    e.printStackTrace();
    anythingWrong = true;
    printSuccessMsg = false;
  }

  // retrieve the checkin id
  int checkin_id = -1;
  try{
    PreparedStatement ps = null;
    ps = conn.prepareStatement("SELECT id FROM checkin WHERE customer_id = ? AND start_date =
?");
    ps.setInt(1, customer_id);
    ps.setString(2, start_date);
    ResultSet rs = ps.executeQuery();
    if(rs.next()){
      checkin_id = rs.getInt("id");
    }
  } catch(SQLException e){
    e.printStackTrace();
    printSuccessMsg = false;
  }

  // insert into reserve for table
  try{
```

```
    PreparedStatement ps = null;
    ps = conn.prepareStatement("INSERT INTO reserve_for(hotel_id, room_number, checkin_id)
VALUES (?, ?, ?)");
    ps.setInt(1, hotel_id);
    ps.setInt(2, room_number);
    ps.setInt(3, checkin_id);
    ps.executeUpdate();
  } catch(SQLException e){
    e.printStackTrace();
    anythingWrong = true;
    printSuccessMsg = false;
  }

  if(anythingWrong){
    try{
      conn.rollback();
      System.out.println("Transaction is rolled back!");
      System.out.println();
    }catch(SQLException e){

    }
  }else{
    try{
      conn.commit();
      System.out.println("Transaction is commited!");
      System.out.println();
    }catch(SQLException e){

    }
  }

  try{
```

```
    conn.setAutoCommit( true );
   }catch (Exception e){
   }

   if(printSuccessMsg){
     System.out.println();
     System.out.println("Assign Room successful!");
     System.out.println();
   }
 }
```