TED UNIVERSITY

CMPE 232: Relational Databases

Project Phase 2

Group 21: Trinity

Defne ŞAHAL - 28552496898

Gökmen ÇAĞLAR - 12590403284

Mehmet Anıl AKGÜL - 13090064036

Changes in Database:

- From room table, foreign key “res_fk” and “reservationid” are deleted. The reason is to be able to create room table before creating reservation table, to insert data without any problem.
- Minor changes in table column names are made. Such as deleting unnecessary spaces after the names: “city ”, “district ”, “phone ”. Changing the column name “floorNo” to “floorno” and “roomTypeid” to “roomtypeid” in order to make all column names in the same naming format.
- In guest table, data type of “phone” is changed from int to bigint (int8), to be able to hold phone numbers.
- In reservation table, reservdate’s not null feature is deleted. For “is null” query, a data from reservdate is deleted.
- With “alter table rename” command , “name” in the guest table is changed to “guestname”.
- With “alter table drop” command, district column is deleted from hotel table. Then, with “alter table add” command, we added district column again. By this way, we tested alter operations. After these operations, district column’s location was changed from middle of the table to the end of the table. So we changed the printing order of column names.

Description of Tables:

We have 8 different tables in our hotel database management system. These are: Guest, Employee, Duty, Payment, Room, RoomType, Hotel and Reservation. Every table of our hotel database management system has a unique role.

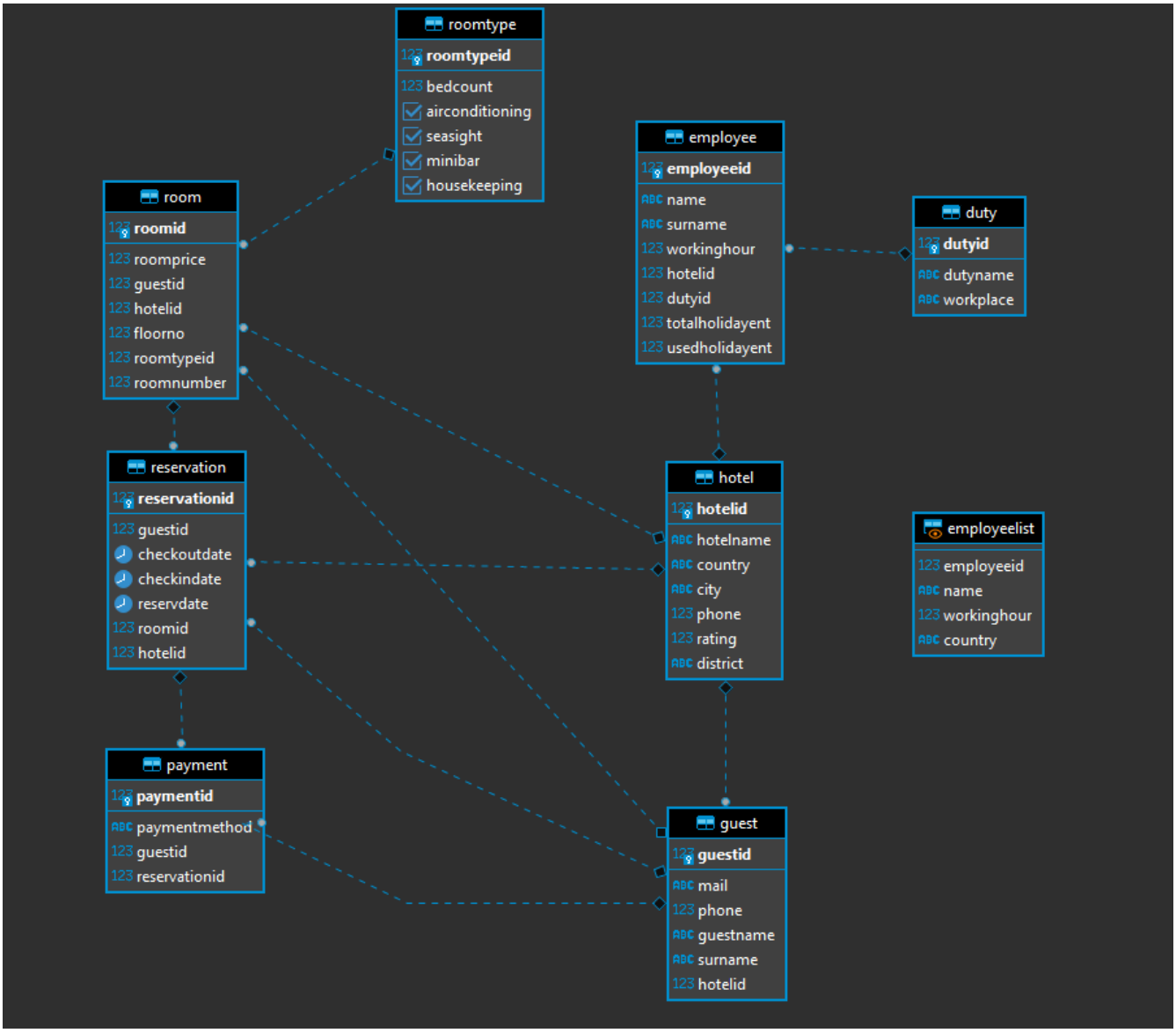
1. In “Guest” table, we can have detailed information about our guests. A guest should be stored in the system by “guestid” attribute, which is the primary key of this table. And we have attributes to see their names, surnames, phone numbers and mail addresses. Also, we have “hotelid” attribute, to understand which guest belongs to which hotel, as a foreign key in this table.

2. From “Employee” table, we can learn some information about hotel employees. An employee is identified by “employeeid” primary key. This table includes information about their working hour, total holiday entitlement, used holiday quantity with the related attributes. Also, this table has “hotelid” and “dutyid” as foreign keys to show which employee working at which hotel and at which job.
3. With “Duty” table, we can see employees’ job varieties. These are stored in the system by “dutyid” primary key. Also, we have attributes for duty name and workplace to have detailed 9information about these jobs.
4. “Payment” table includes some data about our guests’ payment process. These are identified by “paymentid” attribute, which is the primary key of this table. Also, we have an attribute to see payment methods. And this table has two foreign keys, which are “reservationid” and “guestid”.
5. “Room” table basically defines rooms in our hotel database system. A room is identified by a primary key called “roomid”. We can understand which room belongs to which hotel with “hotelid”, which room belongs to which guest with “guestid” and which room has which room type with “roomtypeid” foreign keys. Also, this table has 2 more attributes: “roomprice” and “floorno” to learn about a room’s price and its floor number at a hotel.
6. With “RoomType” table, we can have detailed information about the rooms. Type of a room is stored in the system by “roomtypeid” primary key. We have different attributes to see air conditioning, bed count, sea sight, minibar and housekeeping service information of our rooms in this table.
7. “Hotel” table basically defines hotels in our project. A hotel is identified by a primary key called “hotelid”. In this table, we have some attributes to learn about a hotel’s name, phone number and rating point by people who have stayed in that hotel. Also, we have attributes “country”, “city”, “district” to have detailed information about a hotel’s exact place.
8. “Reservation” table gives us some information about a reservation. A reservation can be recognized by “reservationid” primary key. We can see which reservation belongs to which hotel by “hotelid”, which reservation belongs to which guest by “guestid” and finally which reservation belongs to which room by “roomid” foreign keys. Also, we have

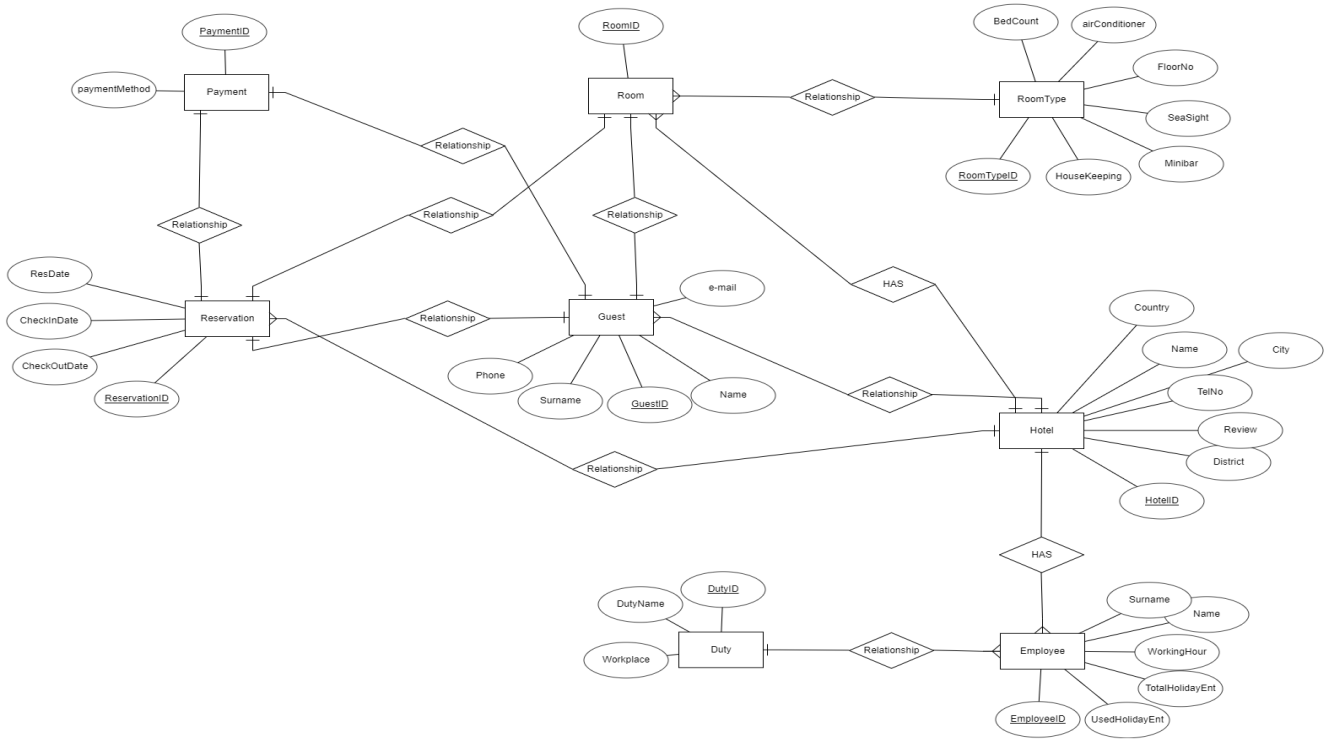
"checkoutdate", "checkindate" and "reservdate" attributes to learn about the exact date details of a reservation from this table.

Final Schema:

Relational Schema:



ER Diagram:



Java Program:

1. First, we set connections, statements and resultSets to null. Then, by using string host, string user and string password, we had access to our database. With this access, we managed to have control of our database from our java ide. After we dealt with the connection, we started to write generic codes. These are basically based on the logic of sending the sql statements that we learnt on our course to the database and getting the responses. So, we make the connection of the statement variable, we used scanner to have some information from the user and when our database is connected, we put the user into a while loop. With this method we had a system that working like a menu. In this menu, the user has 5 different options. These options can be chosen by entering an integer between 1 and 5. So if a user enters 1, it runs the function “List tables”. If a user enters 2 it runs the “Update Employee” function. Third option is “Delete From Payments”, fourth is “Insert Hotel” and the last one, fifth is for the exit from the menu, “Quit Menu”

function. As we mentioned before, we are waiting for the user to enter an integer input. For the case that the user does not enter an integer input, we used some try-catch. By this way, we have prevented the errors due to wrong input. So actually, we tried to create a system without any bugs. But it still contains some bugs in it.

We tried to do all these query operations in our main class. But, for some long operations we created "Methods" class and we put them in here. In this class, we have our printer part. We created printer to print some long strings or the menu. To print the menu, we created "MenuPrinter" in the method. After we called the method in main, we got an input from the user by using our printers. This input shows us which table will be listed according to the user's choice. Then, by using sql select statements that we learnt in our course, and by using "statement.executeQuery" and setting it to "resultSet" we had the input. With the appropriate select query we have managed to list employee, hotel and payment tables. And to print these tables to the user, we used "utils.writeResults" that we created in the "Methods" class.

```
public static void MenuPrinter() {
    System.out.println("Please select function from menu.");
    System.out.println("1-) List Tables");
    System.out.println("2-) Update Employee");
    System.out.println("3-) Delete From Payments");
    System.out.println("4-) Insert Hotel");
    System.out.println("5-) Quit Menu");
}
public static void writeResults(ResultSet resultset,int index) throws SQLException{
    if(index==1) {
        System.out.println("\n\t\tEmployee Table\n");
        System.out.println("EmployeeId | Name | Surname | WorkingHour | HotelID | DutyID | TotalHoliday | UsedHoliday |");
    }else if(index==2) {
        System.out.println("\n\t\t Payment Table\n");
        System.out.println("PaymentID | PaymentMethod | GuestID | ReservationID |");

    }else if(index==3) {
        System.out.println("\n\t\t Hotel Table\n");
        System.out.println("HotelId | HotelName | Country | City | Phone | Rating | District |");
    }else if(index==4) {
        System.out.println("\n\t\tEmployee Table\n");
        System.out.println("EmployeeId | Name | Surname | UsedHoliday |");
    }


    ResultSetMetaData rsmd=resultset.getMetaData();
    int totalCollumnNum=rsmd.getColumnCount();
    System.out.println("-------------------------------------------------------------------------------");
    while(resultset.next()) {
        for(int i=1;i<=totalCollumnNum;i++) {

            System.out.print("| "+resultset.getString(i)+ "\t  ");

        }
        System.out.println();
    }
    System.out.println("-------------------------------------------------------------------------------");
}
```

Menu print and Result Print(writeResults) part of Methods class

```
if(listUserInput==1) {
    resultSet=statement.executeQuery("select * from employee order by employeeid asc");

    utils.writeResults(resultSet,1);
}else if(listUserInput==2) {
    resultSet=statement.executeQuery("select * from payment order by paymentid asc");
    utils.writeResults(resultSet,2);
}else if(listUserInput==3){
    resultSet=statement.executeQuery("select * from hotel order by hotelid asc");
    utils.writeResults(resultSet,3);
}else {
    System.out.println("Please give a valid value");
}
System.out.println();
```

Listing, main method part based on user inputs

2. For “Update Employee” part, we wanted to update 1 significant column. So, we thought that we should change the “used holiday quantity” which is one of our special queries. Total holiday quantity was 30 days. So, we wrote a query that it will print the employees whose used holiday quantities are not equal to 30. Then, we used “statement.executeQuery” like we did in the first part. We asked to the user that who will be updated. We got the answer by an id number. Then again, we asked to the user that what will be the new used holiday quantity. Then according to the integer that we got from the user, we set “statement.executeUpdate” and “update employee set “usedholidayent” to our “updater” variable. “updater” is a static variable that we created before our main class. Then, to make the update more visible to user, row that effected by updater printed to user by “statement.executeUpdate”. And finally, the whole table printed to user to see the update is successful or not.

3. For the “delete” part, we chose payment because it was the only table that did not include any other table’s primary key. So, we knew that it would be harmless for relations if we delete it. For “Delete From Payments” part, first we printed all the table with “statement.executeQuery” with select query and “utils.writeResults”(utils means Methods helper class) to print which parts will be appropriate for the delete process to the user. Then with “int paymentid”, we asked the user which payment will be deleted. Then we got the response from the user with an integer. So, again we used “statement.executeUpdate” with a delete query to get the delete process done. Then again we used “statement.executeQuery” with “utils.writeResults” to show to the user that the delete process is done.

4. For “Insert Hotel” part, first we used “statement.executeQuery” again with select query. Then by using “utils.writeResults” we printed the whole hotel list to the user. Then, we wanted a hotel id, a hotel name, a country, a city, a district, a telephone number and a rating from the user respectively. Hotel id, telephone number and rating were integers, while hotel name, country, city and district were strings. And we tried to put some limitations like rating should be between 0 and 5, or a telephone number’s max length should be 9 digits. Then after we had all these inputs from the user, we used “statement.executeUpdate” with “ INSERT INTO hotel” method and we inserted the values. After that we set our “resultSet” to “statement.executeQuery” and we printed the hotel table again. By this way, we saw that our insertion was successful.

5. If the user enters “5” and selects the option “Quit Menu”, the user directly exits from the system. And after every process, we are asking the user if he/she wants to continue or not. According to the answer, we are putting the user into a while loop again or not. Then finally, we created a method called “close”. This method closes “resultSet”, “statement” and “connect” after we close the database to prevent the errors. If it doesn’t work, we are still able to see the error with some try-catch.

Basic code snippets:

```
try {
    int employeeid=scanner.nextInt();
    System.out.println("Now please to set used holiday collumn give an integer value");
    int usedHol=scanner.nextInt();
    updater=statement.executeUpdate("update employee set usedholidayent="+usedHol+" where employeeid="+employeeid);
    System.out.println("Rows impacted : " + updater );
    resultSet=statement.executeQuery("select employeeid,name,surname,usedholidayent from employee where usedholidayent<30 "
            + "order by employeeid asc");
    utils.writeResults(resultSet,4);

}catch(Exception e) {
    System.out.println("Please give an integer value!");
}
```

update part

```
}else if(userInput==3) {
    //This is the delete payment part
    resultSet=statement.executeQuery("select * from payment order by paymentid asc");
    utils.writeResults(resultSet,2);
    System.out.println("Select a payment id to delete it.");
    int paymentid=scanner.nextInt();
    updater=statement.executeUpdate("delete from payment where paymentid="+paymentid);
    resultSet=statement.executeQuery("select * from payment order by paymentid asc");
    utils.writeResults(resultSet,2);
```

delete part

```
try {
    updater=statement.executeUpdate("INSERT INTO hotel (hotelid, hotelname, country, city, district, phone, rating) "+
            "VALUES('"+hotelid+"', '"+hotelname+"', '"+country+"', '"+city+"', '"+district+"', '"+phone+"', '"+rating+"')");
    resultSet=statement.executeQuery("select * from hotel order by hotelid asc");
    utils.writeResults(resultSet,3);

}catch (SQLException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
```

insert part