CS 202
Design Phase
Schema Design

FIRAT KARAKURT, RÜZGAR GÜMÜŞ

firat.karakurt@ozu.edu.tr, ruzgar.gumus@ozu.edu.tr

The design for a comprehensive and effective hotel management system utilizing a MySQL database and Java JDBC is being delivered by our team with enthusiasm. This system is intended to simplify and automate a number of hotel administration and operation-related operations, including making reservations, keeping track of visitors, and generating reports.

To do this, we have carefully created a MySQL database to hold all the relevant information about the hotel, its rooms, and its visitors. The database is made to be adaptable and scalable so that it can meet the demands of a variety of hotels. It may be accessed and changed using Java JDBC, which enables us to create straightforward and effective SQL statements to query and manage the data.

The Entity-Relationship (ER) Diagram of our database, which shows the many entities (such as rooms and guests) and their relationships, will be shown in this report initially (e.g. a guest can book multiple rooms). The DML (Data Modification Language) and DDL (Data Definition Language) statements that we utilized to build and populate the database will then be provided. These statements demonstrate how the database's structure, including the tables and their attributes, was constructed as well as how data was added to the tables.

The design decisions we took when creating the database will then be discussed. This will include a justification for our choices and how they affect the operation and effectiveness of the hotel management system.

We hope that our design will prove to be a useful tool for hotel management because we think it is both simple and effective.

We have used MySQL to build a database for our hotel management system design that contains all the necessary data about the hotel, its rooms, and its visitors. Five tables make up the database, which is built to accommodate the many duties needed to administer a hotel and to hold different types of data.

Users of the system, including both guests and employees, are listed in the first table, User. We can keep track of each user's contact information and personal information using this table, which is useful for doing things like making reservations for rooms and interacting with visitors.

Information on the hotel rooms is kept in the second table, Rooms. For purposes like making reservations and monitoring the inventory of rooms, this table enables us to keep track of the various room kinds, their capacities, and their amenities.

The third table, Bookings, contains data regarding the reservations that visitors have made. This table enables us to keep track of each reservation's specifics, such as the person who made it, the room it was for, and the dates it was made. Additionally, it enables us to monitor each booking's payment status, which is vital for jobs like creating reports and overseeing the hotel's finances.

The fourth table, Housekeeping, stores information about the housekeeping tasks performed by the staff. This table allows us to track the details of each task, including the staff member who performed the task, the room where the task was performed, and the dates and times when the task was started and completed. This information is necessary for tasks such as managing the workload of the housekeeping staff and ensuring that all rooms are properly cleaned and maintained.

Finally, the fifth table, Payments, stores information about the payments made by the guests. This table allows us to track the details of each payment, including the booking for which the payment was made, the payment type, the payment amount, and the payment date. This information is necessary for tasks such as generating reports and managing the financial aspects of the hotel.

Overall, the five tables in our database provide a comprehensive and efficient solution for managing a hotel and its operations. They allow us to store and retrieve all the necessary data in an organized and efficient manner, and they facilitate the various tasks that are required to manage a hotel effectively.

'User' Table

```
7  CREATE TABLE User (
8      user_id          int            not null auto_increment,
9      user_type        varchar(50)    not null,
10     name             varchar(100)   not null,
11     email            varchar(50)    not null,
12     phone            int            not null,
13
14     primary key(user_id)
15     );
```

- This table stores information about the users of the hotel management system. The user_id column is the primary key and is used to uniquely identify each user. The user_type column indicates whether the user is a guest, staff member, receptionist or an admin. The name, email, and phone columns store the name, email address, and phone number of the user, respectively.

'Rooms' Table

```
17 CREATE TABLE Rooms (
18     room_id          int          not null auto_increment,
19     room_name        varchar(50)  not null,
20     room_type        varchar(50)  not null, -- Single, Twin, Double, Deluxe Rooms, Rooms with a View, Suites
21     bed_size         varchar(50)  not null,
22     capacity         int          not null,
23     amentities       varchar(50)  not null, -- a desirable or useful feature or facility of a building or place
24
25     primary key(room_id)
26     );
```

- This table stores information about the rooms in the hotel. The room_id column is the primary key and is used to uniquely identify each room. The room_name column stores the name of the room, while the room_type column indicates the type of room (e.g. single, twin, double). The bed_size column indicates the size of the bed(s) in the room, and the capacity column indicates the maximum number of people the room can accommodate. Finally, the amentities column stores a list of amenities that are available in the room.

'Bookings' Table

```
28  CREATE TABLE Bookings (
29      booking_id          int          not null auto_increment,
30      guest_id            int          not null,
31      room_id             int          not null,
32      check_in            datetime     not null,
33      check_out           datetime     not null,
34      payment_status      varchar(50)  not null,
35
36      primary key(booking_id),
37      foreign key(guest_id) references User(user_id) on delete cascade,
38      foreign key(room_id) references Rooms(room_id) on delete cascade
39  );
```

- This table stores information about the bookings made by the guests. The booking_id column is the primary key and is used to uniquely identify each booking. The guest_id and room_id columns store the ID of the guest who made the booking and the ID of the room that was booked, respectively. The check_in and check_out columns store the dates and times when the guest is expected to check in and check out of the room. Finally, the payment_status column indicates whether the payment for the booking has been made or not. This table also has foreign keys that reference the User and Rooms tables to ensure that the data is consistent.

'Housekeeping' Table

```
41  CREATE TABLE Housekeeping (
42      housekeeping_id     int         not null auto_increment,
43      staff_id            int         not null,
44      room_id             int         not null,
45      start_time          datetime    not null,
46      end_time            datetime    not null,
47
48      primary key(housekeeping_id),
49      foreign key(staff_id) references User(user_id) on delete set null, -- if user was deleted, atleast we know that certain room was cleaned
50      foreign key(room_id) references Rooms(room_id) on delete set null -- if room was deleted, atleast we know a staff cleaned it, so he/she gets paid :)
51  );
```

- This table stores information about the housekeeping tasks performed by the staff. The housekeeping_id column is the primary key and is used to uniquely identify each task. The staff_id and room_id columns store the ID of the staff member who performed the task and the ID of the room where the task was performed, respectively. The start_time and end_time columns store the dates and times when the task was started and completed. This table also has foreign keys that reference the User and Rooms tables to ensure that the data is consistent.

'Payments' Table

```sql
CREATE TABLE Payments (
payment_id          int         not null auto_increment,
booking_id          int         not null,
payment_type        varchar(50) not null,
payment_amount      int         not null,
payment_date        datetime    not null,

primary key(paymnet_id),
foreign key(booking_id) references Bookings(booking_id) on delete set null -- if booking id was deleted, we keep payment data
);
```

- This table stores information about the payments made by the guests. The payment_id column is the primary key and is used to uniquely identify each payment. The booking_id column stores the ID of the booking for which the payment was made. The payment_type column indicates the type of payment (e.g. cash, credit card). The payment_amount column stores the amount paid, and the payment_date column stores the date and time when the payment was made. This table also has a foreign key that references the Bookings table to ensure that the data is consistent.

Next we will be checking if our database is in 3NF form.

The User table is in 1NF because all of its attributes are atomic. It also has no partial dependencies or transitive dependencies, so it is in 2NF.

The Rooms table is in 1NF because all of its attributes are atomic. It also has no partial dependencies or transitive dependencies, so it is in 2NF.

The Housekeeping table is in 1NF because all of its attributes are atomic. It also has no partial dependencies or transitive dependencies, so it is in 2NF.

The Payments table is in 1NF because all of its attributes are atomic. It also has no partial dependencies or transitive dependencies, so it is in 2NF.

The Bookings table is in 1NF because all of its attributes are atomic. It also has no partial dependencies, but it does have a transitive dependency (check_out depends on check_in). To make it 2NF, we would need to remove this transitive dependency by separating the check_in and check_out attributes into their own table.

To make our database 3NF, we can start by removing the transitive dependency in the Bookings table. To do this, we can create a new table called Booking_Dates that stores the check_in and check_out dates for each booking:

CREATE TABLE Booking_Dates (booking_id int not null, check_in datetime not null, check_out datetime not null, primary key(booking_id), foreign key(booking_id) references Bookings(booking_id) on delete cascade);

We can then remove the check_in and check_out columns from the Bookings table. With these changes, the Bookings table is now in 2NF.

With these changes the database is now in 3NF.

Updated *'Bookings'* Table and new *'Booking_Dates'* Table

```
28  CREATE TABLE Bookings (
29    booking_id         int          not null auto_increment,
30    guest_id           int          not null,
31    room_id            int          not null,
32    payment_status     varchar(50)  not null,
33
34    primary key(booking_id),
35    foreign key(guest_id) references User(user_id) on delete cascade,
36    foreign key(room_id) references Rooms(room_id) on delete cascade
37  );
38
39  CREATE TABLE Booking_Dates (
40    booking_id         int          not null,
41    check_in           datetime     not null,
42    check_out          datetime     not null,
43
44    primary key(booking_id),
45    foreign key(booking_id) references Bookings(booking_id) on delete cascade
46  );
```

ER diagram:

'User' Entity:
In this diagram, the entities are represented by rectangles and the relationships between them are represented by lines connecting the entities.
Attributes: user_id (primary key), user_type, name, contact_info, personal_info
Relationships:
One-to-many relationship with the 'Bookings' entity
One-to-many relationship with the 'Housekeeping' entity
'Rooms' Entity:

Attributes: room_id (primary key), room_name, room_type, bed_size, capacity, amenities
Relationships:
Many-to-many relationship with the 'Bookings' entity
Many-to-one relationship with the 'Housekeeping' entity
'Bookings' Entity:

Attributes: booking_id (primary key), user_id, room_id, payment_status
Relationships:
One-to-one relationship with the 'Booking_Dates' entity
One-to-many relationship with the 'Payments' entity
'Booking_Dates' Entity:

Attributes: booking_id (primary key), check_in, check_out
Relationships:
One-to-one relationship with the 'Bookings' entity
'Housekeeping' Entity:

Attributes: housekeeping_id (primary key), user_id, room_id, start_time, end_time
Relationships:
Many-to-one relationship with the 'Rooms' entity
'Payments' Entity:

Attributes: payment_id (primary key), booking_id, payment_type, amount, payment_date

Relationships:
One-to-many relationship with the 'Bookings' entity

The 'User' entity is connected to the 'Bookings' entity with a line that has a one-to-many cardinality symbol (a line with an arrow on one end) on the 'User' side. This indicates that one user can make multiple bookings.

The 'Rooms' entity is connected to the 'Bookings' entity with a line that has a many-to-many cardinality symbol (a double line) on both ends. This indicates that one room can be booked multiple times and one booking can include multiple rooms.

The 'Bookings' and 'Booking_Dates' entities are connected with a line that has a one-to-one cardinality symbol (a line with an arrow on both ends) on both sides. This indicates that each booking has a corresponding set of check-in and check-out dates.

The 'Housekeeping' and 'Payments' entities are both connected to the 'Bookings' entity with a line that has a one-to-many cardinality symbol on the 'Bookings' side. This indicates that one booking can have multiple housekeeping tasks or payments associated with it.