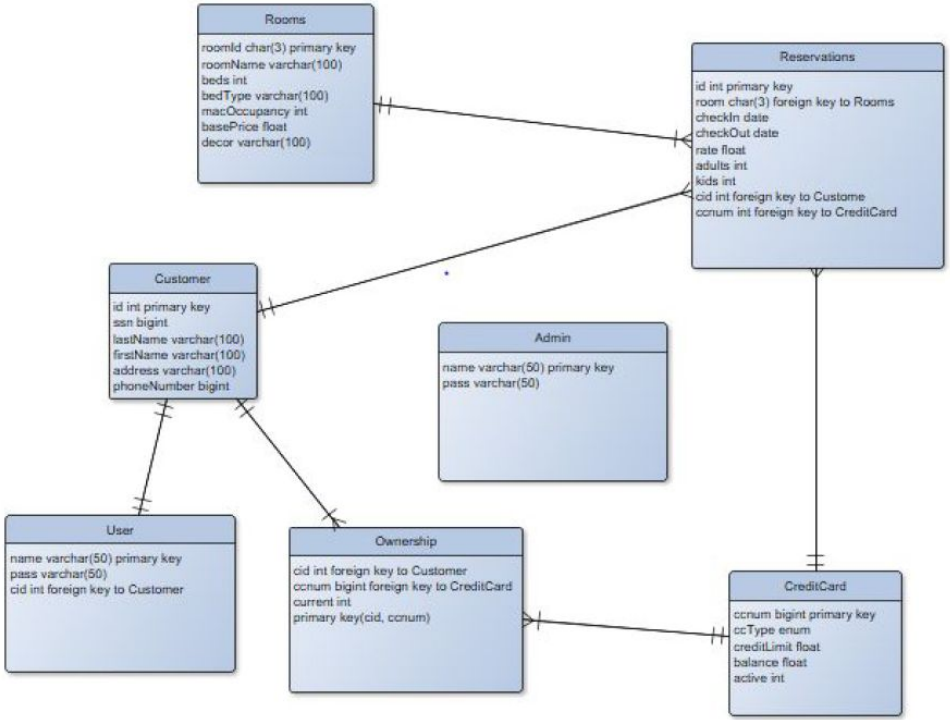
Hotel Booking Management System

Group 01: Matthew Choi, Dylan Martin, Riley Wartenberg, Kevin Yoo

E-R Diagram


Rooms
roomId char(3) primary key
roomName varchar(100)
beds int
bedType varchar(100)
maxOccupancy int
basePrice float
decor varchar(100)
Reservations
id int primary key
room char(3) foreign key to Rooms
checkIn date
checkOut date
rate float
adults int
kids int
cid int foreign key to Custome
ccnum int foreign key to CreditCard
Customer
id int primary key
ssn bigint
lastName varchar(100)
firstName varchar(100)
address varchar(100)
phoneNumber bigint
Admin
name varchar(50) primary key
pass varchar(50)
User
name varchar(50) primary key
pass varchar(50)
cid int foreign key to Customer
Ownership
cid int foreign key to Customer
ccnum bigint foreign key to CreditCard
current int
primary key(cid, ccnum)
CreditCard
ccnum bigint primary key
ccType enum
creditLimit float
balance float
active int

Database Schema

```
CREATE TABLE `rooms` (
  `roomId` char(3) NOT NULL,
  `roomName` varchar(100) DEFAULT NULL,
  `beds` int DEFAULT NULL,
  `bedType` varchar(100) DEFAULT NULL,
  `maxOccupancy` int DEFAULT NULL,
  `basePrice` float DEFAULT NULL,
  `decor` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`roomId`),
  UNIQUE KEY `roomName` (`roomName`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Rooms

- The rooms table is used to store all of the information about the rooms that are available in this hotel.
- The table stores information about the max occupancy for thee room as well as the decor,bed type, roomname, number of beds, and price.
- The roomid attribute is used to join with the reservation table so that information about the room can be used in conjunction with the availabilities of the rooms.

Database Schema (cont.)

```
CREATE TABLE `Reservations` (
  `Cid` int DEFAULT NULL,
  `id` int NOT NULL AUTO_INCREMENT,
  `room` char(3) DEFAULT NULL,
  `checkIn` date DEFAULT NULL,
  `checkout` date DEFAULT NULL,
  `rate` float DEFAULT NULL,
  `adults` int DEFAULT NULL,
  `kids` int DEFAULT NULL,
  `ccnum` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `room` (`room`,`checkIn`),
  KEY `Cid` (`Cid`),
  CONSTRAINT `Reservations_ibfk_1` FOREIGN KEY (`Cid`) REFERENCES `customer` (`cid`),
  CONSTRAINT `Reservations_ibfk_2` FOREIGN KEY (`room`) REFERENCES `rooms` (`roomId`)
) ENGINE=InnoDB AUTO_INCREMENT=47522 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Reservations

- The reservations table stores all the information about the reservations that have been made in the database.
- The table stores information about the Customer ID of who made it, the reservation id, the room thay reserverved, the checkin and checkout dates, the rate, the number of kids and adults.
- The reservation uses the room attribute to join with the rooms table so that the customer can get information about their room that they are reserving.

Database Schema (cont.)

```
CREATE TABLE `Users` (
  `name` varchar(50) NOT NULL,
  `pass` varchar(50) NOT NULL,
  `cid` int NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Users

- The Users table is used to allow each person who signs in to have an individualized experience and access the website.
- The table stores information about the Customer ID of who made it, their username, and their password
- The User table uses the customer ID in conjunction with the reservation table to get information about current and previous reservations and display it to the user.

Database Schema (cont.)

```
CREATE TABLE `customer` (
  `ssn` bigint NOT NULL,
  `cid` int NOT NULL AUTO_INCREMENT,
  `lastName` varchar(100) DEFAULT NULL,
  `firstName` varchar(100) DEFAULT NULL,
  `address` varchar(100) DEFAULT NULL,
  `phoneNumber` bigint DEFAULT NULL,
  PRIMARY KEY (`ssn`),
  KEY `cid` (`cid`),
  CONSTRAINT `customer_chk_1` CHECK ((`ssn` < 999999999)),
  CONSTRAINT `customer_chk_2` CHECK ((`phoneNumber` < 9999999999))
) ENGINE=InnoDB AUTO_INCREMENT=72 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Customer

- The Customer table is used to track who each customer is that is using the website.
- The table stores information about the Social security number, customer id, last name, firstname, address, and phone number of each individual customer.
- The Customer table is used throughout the project to identify customers for personalization of the website as well as for functionality to ensure that each person is charged correctly and has the correct records.

Database Schema (cont.)

```
CREATE TABLE `creditCard` (
  `ccnum` int NOT NULL AUTO_INCREMENT,
  `ccType` enum('Visa','MC','AmericanExpress','Discover') DEFAULT NULL,
  `creditLimit` float DEFAULT NULL,
  `balance` float DEFAULT '0',
  `active` int DEFAULT '0',
  PRIMARY KEY (`ccnum`),
  CONSTRAINT `creditCard_chk_1` CHECK ((`balance` <= `creditLimit`))
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Credit Card

- The Credit Card table is used to track how payments will be handled through our website.
- The table stores information about the credit card number, the credit card type, the credit card limit, and the balance.
- The Credit Card is used in conjunction with the Customer and the Transaction table to ensure that each person who reserves a room has enough money in their account to book the room.

Database Schema (cont.)

```
CREATE TABLE `ownership` (
  `cid` int NOT NULL,
  `ccnum` int NOT NULL,
  `current` int DEFAULT '0',
  PRIMARY KEY (`cid`,`ccnum`),
  KEY `unique_ccn` (`ccnum`),
  CONSTRAINT `unique_ccn` FOREIGN KEY (`ccnum`) REFERENCES `creditCard` (`ccnum`),
  CONSTRAINT `unique_id` FOREIGN KEY (`cid`) REFERENCES `customer` (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Ownership

- The Ownership table is used to track which customer is associated with which credit card.
- The table stores information about the customer ID, the credit card number, and whether the card is current.
- The Ownership tabel is used primarily to link the customer table with the credit card table so that way credit cards can be tracked for each purchase regardless of person and still check to make sure there is enough money in the account.

Database Schema (cont.)

```
CREATE TABLE `Admin` (
  `name` varchar(50) NOT NULL,
  `pass` varchar(50) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Admin

- The Admin table is used to allow a user to login as an administrator instead of a regular user which allows for additional functionality on the website.
- The table stores information about the username, and password of the admin account.
- The Admin table is used to allow a user with admin privileges to access a portion of the website which displays the revenue per room for each each month and the total revenue per month in addition to the total revenue per year.

Transactions

- Handled by method “Transaction” in DaoManager class
  - Turns sets **autocommit = false**
  - Executes SQL query/command
  - **Commit** changes to database
- Avoid double booking when 2 users try to book at the same time

```
public String transactionUpdate(DaoCommand command){
    try{
        this.conn.setAutoCommit(false);
        System.out.println("yes");
        String returnValue = ((ReservationsDaoCommandImpl)command).executeUpdate( daoManager: this);
        System.out.println("no");
        System.out.println(returnValue);
        this.conn.commit();
        System.out.println("commit");
        return returnValue;
```

Transactions (cont.)

- SQL triggers
  - Used to **a) avoid double bookings** and **b) successfully charge user credit card before booking**

```
DELIMITER $
CREATE TRIGGER check_overbook BEFORE INSERT ON Reservations
    FOR EACH ROW
    BEGIN
        IF 0 < (SELECT COUNT(*) FROM Reservations r
                    WHERE NEW.room = r.room AND
                            ((NEW.checkIn BETWEEN r.checkIn AND r.checkOut) OR
                            (NEW.checkOut BETWEEN r.checkIn AND r.checkOut) OR
                            (NEW.checkIn < r.checkIn AND NEW.checkOut > r.checkOut))) THEN
            SIGNAL SQLSTATE '12345'
            SET MESSAGE_TEXT = 'These dates are unavailable';
        END IF;
    END$
DELIMITER ;
```

Transactions (cont.)

- SQL triggers
  - Used to **a) avoid double bookings** and **b) successfully charge user credit card before booking**

```
DELIMITER $
CREATE TRIGGER charge_creditcard AFTER INSERT ON Reservations
    FOR EACH ROW
    BEGIN
        UPDATE creditCard SET balance = balance + (NEW.rate * DATEDIFF(NEW.CheckOut, NEW.CheckIn))
            WHERE ccnum = NEW.ccnum;
    END$
DELIMITER ;
```

DEMO