Hotel Database Design

User requirements

Data requirements

Hotel

Information about the hotels will need to be stored in the database. This includes the hotel name, rating, address, phone and breakfast price. I am assuming that all hotels will at least offer breakfast. They can also store any number of additional facilities with a boolean value to check if its free or not. Another assumption I am making is the price for breakfast in a hotel is the same for both children and adults and that there is a choice to have breakfast every day of their stay or none at all.

Room

Information about the rooms in the hotel will also need to be stored. Information such as the number, type, price, and discount of the room will need to be stored. If a room ever undergoes construction, information for when construction starts and ends will need to be stored. I am assuming that for the same hotel the cost of each room type is the same for all rooms of that type. Typical room types would be double, twin, single etc.

Customer

Information on customers that are staying at the hotel will need to be stored. The database should store their full name and title, address, card details, email and phone number.

Reservation

When a room is reserved by a customer, information about the booking will need to be stored. There will need to be booking date, check-in date, check-out date, the number of children and adults staying, the total number of nights stayed and special instructions if a customer needs to give them. Customers can cancel reservations if they wish, so there is also a need to store the cancellation date, if this date is at least one day before their check-in date, the customer will be refunded. Another assumption is that refurbishments are planned ahead of time. Due to this the refurbishment will not conflict with any existing booking, avoiding the need to cancel a reservation.

Payment

When a customer makes a reservation, information on the payment should also be stored such as the method of payment (pay by card, pay with cash, etc), the payment time and if the payment has been completed. Using this information an invoice can be generated.

Transaction requirements

Queries

- I will assume that any queries made (such as a hotel booking request) do not need to be stored anywhere in the database.

User

- List all hotels that match desired criteria such as in a city or postcode, has available room type, has a high enough rating, below a certain price, has free parking.
- List all rooms in a hotel that match desired criteria such as availability on given date, room type, price.

Admin

- List everyone who has booked a room and give some basic information on them as well as how much they have payed.
- Produce a booking status report that includes all the rooms that are booked on a given date and who is booking them and if they have payed.
- Total the number of guests staying in the hotel on a given date.
- List the availability of a room in a given time frame, if on a day a room is booked then show the customer who booked it.
- Total the number of breakfasts ordered on a given day.
- Get an invoice from payments.

Manipulation

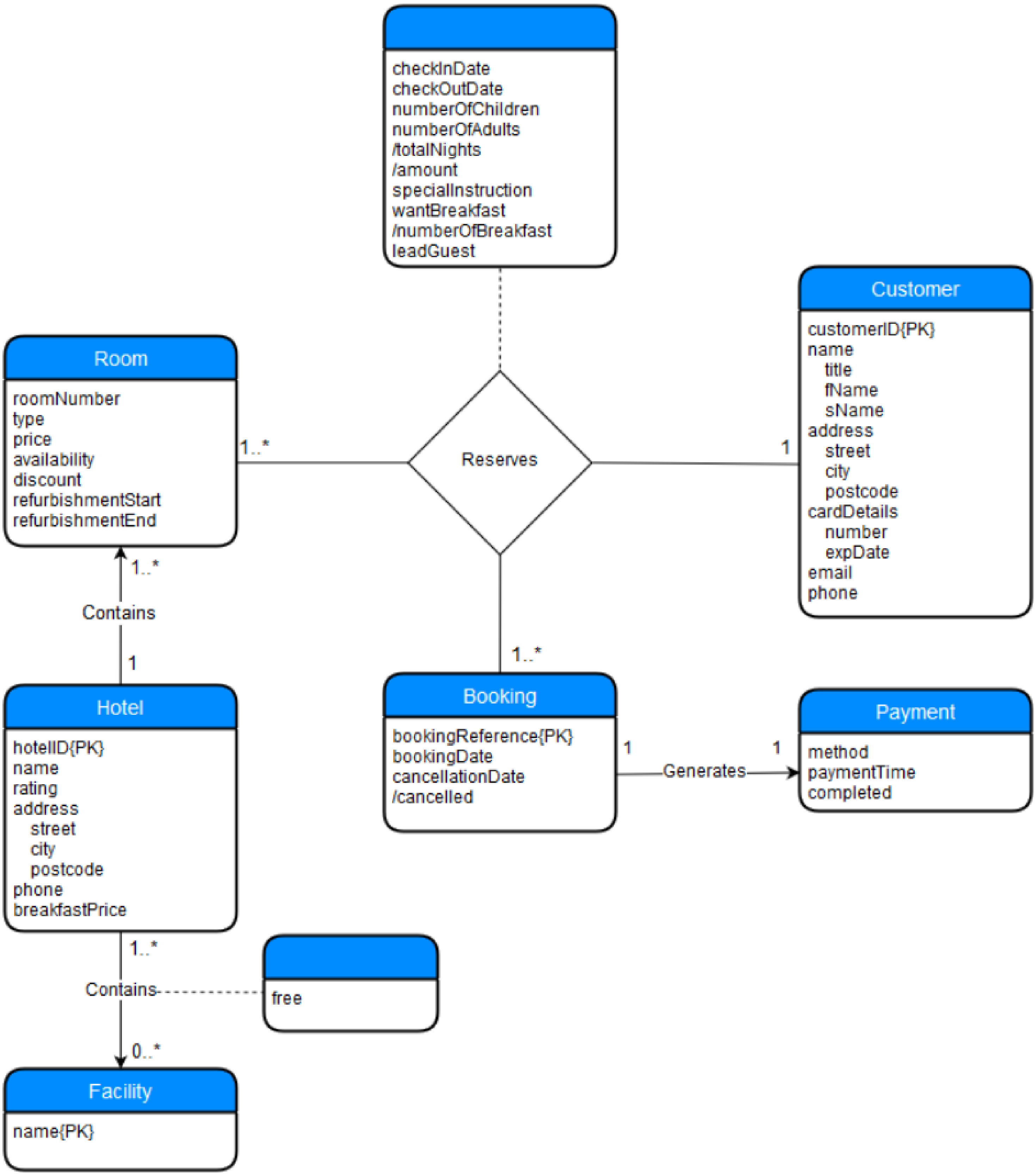
- Update public information about the hotel such as ratings, prices.
- Update information about rooms such as if they are available.
- The ability to delete reservations. I am assuming reservations cannot be updated.
- The ability to update room discounts.
- The ability to delete rooms for cases when hotels undergo contractions and parts of the hotel are changed.
- The ability to delete hotels.

Insertions

- The ability to insert a new hotel.
- The ability to insert new rooms.
- The ability to insert new reservations.
- The ability to insert new facilities.

Conceptual Model

Diagram



Entity and Attributes Reasoning

Hotel

Hotel is a strong entity as it won't be dependent on any other entity. It will use a unique ID as its primary key, this is needed as hotels under the same branch may use the same name. Address is a composite attribute composed of street, city and postcode.

Customer

Customer is another strong entity that uses a unique ID as its primary key, as it is possible for two customers to have the same name and all other attributes may change in the future. The attributes name, address and cardDetails are all composite attributes made up of more specific attributes.

Room

Room is a weak entity type as its existence is dependent on Hotel, you cannot have a room that does not belong to a hotel.

Booking

Booking is a strong entity that uses a bookingReference as its primary key, as the bookingDate and cancellationDate may not be unique, it's possible that two customers book and cancel on the same dates. The attribute cancelled can be derived from the cancellation date. If the cancellation date is not populated, then it means the booking has not been cancelled.

Payment

Payment is a weak entity as there must be a booking for it to exist.

Facility

Facility is a weak entity as facilities in this context must belong to a hotel. The attribute name can be used as the primary key as there cannot be different facilities of the same name.

Reserves

Reserves is a relational attribute that stores information on a customer's reservation. The attribute totalNights is a derived attribute that can be calculated from the difference between the check-in-date and check-out-date. The attribute amount will store the cost of the reservation and is also a derived attribute that can be calculated from many other attributes. The attribute numberOfBreakfast can be derived from the total number of guests and nights and the wantBreakfast attribute. This can be done since I have assumed a person either has breakfast every day or none at all.

Relationships and Multiplicity Reasoning

Reserves

Reserves is a complex ternary relationship involving Customer, Room and Booking. The Customer and Booking pair may reserve many rooms, since one reservation may reserve many rooms. The Room and Customer pair have a one-to-many relationship with Booking, as one customer may book one room many times on different dates. The Room and Booking pair may be reserved by only one customer as a booking to a room made by a customer only belongs to that customer.

Contains Relationship Between Hotel and Facility

This relationship has an attribute called free. This is needed to determine if a hotel wants to make said Facility free or not.

Other Entities

Hotel has a one-to-many relationship with Room as hotels must contain at least one, and practically will contain many rooms. In the reverse case, Room has a one-to-one relationship with Hotel, as each room must only belong to one hotel. Hotel also has a zero-to-many relationship with Facility, depending on the quality of the hotel it may have lots of facilities or none at all. I am also assuming that when a facility is created it must be offered by at least one hotel, thus Facility has a one-to-many relationship with Hotel.

Payment has a one-to-one relationship with Booking as when a booking is made only one payment will ever be generated.

Logical Design and Reasonings

Relational Model

Hotel(hotelID, hotelName, rating, street, city, postcode, phone, breakfastPrice)
Primary Key: hotelID

Room(roomNumber, hotelID, price, roomType, available, discount, refurbishmentStart, refurbishmentEnd)
Primary Key: roomNumber, hotelID
Foreign Key: hotelID references **Hotel**(hotelID)

IsFree(hotelID, facilityName, free)
Primary Key: hotelID, facilityName
Foreign Key: hotelID references **Hotel**(hotelID)
Foreign Key: name references **Facility**(name)

Facility(facilityName)
Primary Key: facilityName

Reservation(hotelID, roomNumber, bookingReference, customerID, checkInDate, checkOutDate, numberOfChildren, numberOfAdults, specialInstructions, wantBreakfast, leadGuest)
Primary Key: hotelID, roomNumber, bookingReference
Foreign Key: hotelID references **Room**(hotelID)
Foreign Key: roomNumber references **Room**(roomNumber)
Foreign Key: bookReference references **Booking**(bookingReference)
Foreign Key: customerID references **Customer**(customerID)

Booking(bookingReference, bookingDate, cancellationDate, paymentMethod, paymentTime, paymentCompleted)
Primary Key: bookingReference

Customer(customerID, title, fName, sName, street, city, postcode, cardNumber, expDate, email, phone)
Primary Key: customerID

Relational Model Reasoning

Hotel and Facility

The attribute for name in the hotel and facility was changed to hotelName and facilityName since 'name' is a reserved keyword in MySQL.

Room

Rooms can be uniquely identified using a composite key made out of number and hotelID, under the assumption that a hotel will not have two rooms of the same number.

The attribute for type has been renamed to roomType since type is a reserved keyword in MySQL.

IsFree

I converted the relational attribute between Hotel and Facility into its own table; IsFree. Its primary key is a composite key made out of hotelID and name (from facility), which are also foreign keys. This will always be unique under the assumption that no two facilities have the same name and every hotel can only have one of each facility.

Reservation

The Reserves relationship has been transformed into a Reservation table, in order to uniquely identify this table I used the foreign keys from the 'many(*)'-side of the relationship.

Payment

Since Payment is a weak entity with a one-to-one relationship with Booking there is no need to create a new table, instead attributes from Payment can be stored in the Booking table.

Normalisation

In order to ensure our table was in 3rd normal form I made the following changes to the customer table. This due to the following transitive dependency; customerID→cardNumber→expDate.

Another transitive dependency can be found in both the customer and hotel tables. The transitive relations are as follows; hotelID→postcode→city and customerId→postcode→city.

Original Customer

Customer(customerID, title, fName, sName, street, city, postcode, cardNumber, expDate, email, phone)
Primary Key: customerID
Foreign Key: cardNumber references **Customer**(cardNumber)

Original Hotel

Hotel(hotelID, hotelName, rating, street, city, postcode, phone, breakfastPrice)
Primary Key: hotelID

Normalised Customer and Hotel

CardDetail(cardNumber, expDate)
Primary Key: cardNumber

City(postcode, city)
Primary Key: postcode

Customer(customerID, title, fName, sName, street, postcode, cardNumber, email, phone)
Primary Key: customerID
Foreign Key: cardNumber references **CardDetail**(cardNumber)
Foreign Key: postcode references City(postcode)

Hotel(hotelID, hotelName, rating, street, postcode, phone, breakfastPrice)
Primary Key: hotelID
Foreign Key: postcode references **City**(postcode)

To improve the integrity of the data I have decide to move the discount and room price to a new table. For example, if the owner of wants to update the discount for a certain room type, they would need to update every room of that type. With a new table, updating one attribute will allow the owner to update the prices for all rooms of that type.

Original Room Table

Room(roomNumber, hotelID, price, roomType, available, discount)
Primary Key: roomNumber, hotelID
Foreign Key: hotelID references **Hotel**(hotelID)

New Room Table and RoomPricing Table

Room(roomNumber, hotelID, roomType, available)
Primary Key: roomNumber, hotelID
Foreign Key: hotelID references **Hotel**(hotelID)
Foreign Key: (hotelID, roomType) references **RoomPricing**(hotelID, roomType)

RoomPricing(hotelID, roomType, discount, price)
Primary Key: hotelID, roomType

Physical Model Design

Derived Data

I have decided to not store our derived data for totalNights and amount as the calculations needed for these attributes are cheap and can be done quickly meaning more space can be saved in the database. The cancelled attribute can also be calculated each time, since checking if the cancellationDate attribute is null should be an inexpensive operation.

Calculating Availability of Rooms

To check if rooms are available, I will calculate this each time when queried. If I were to save the availability of rooms, I would need another table with attributes for room number, hotel ID, date and availability, since there could be an indefinite amount of dates (I need to check availability on every possible date for every possible room) it would take up too much storage space.

Refunds

Whether a return is needed will also be calculated each time as it will only be needed a few times. This can be calculated from the cancellation date and the booking date when needed.

Check out the SQL files to see the database design in action with example queries and updates.