# Group 41 : Coursework 1 - Hotel System

A hotel has rooms of three types: excellent, deluxe and magnificent. It has several rooms of each of these types, but all rooms of a particular type are priced the same and have the same facilities. Each room is situated on only one floor of the hotel, and a room is identified by the floor number followed by another two-digit number.

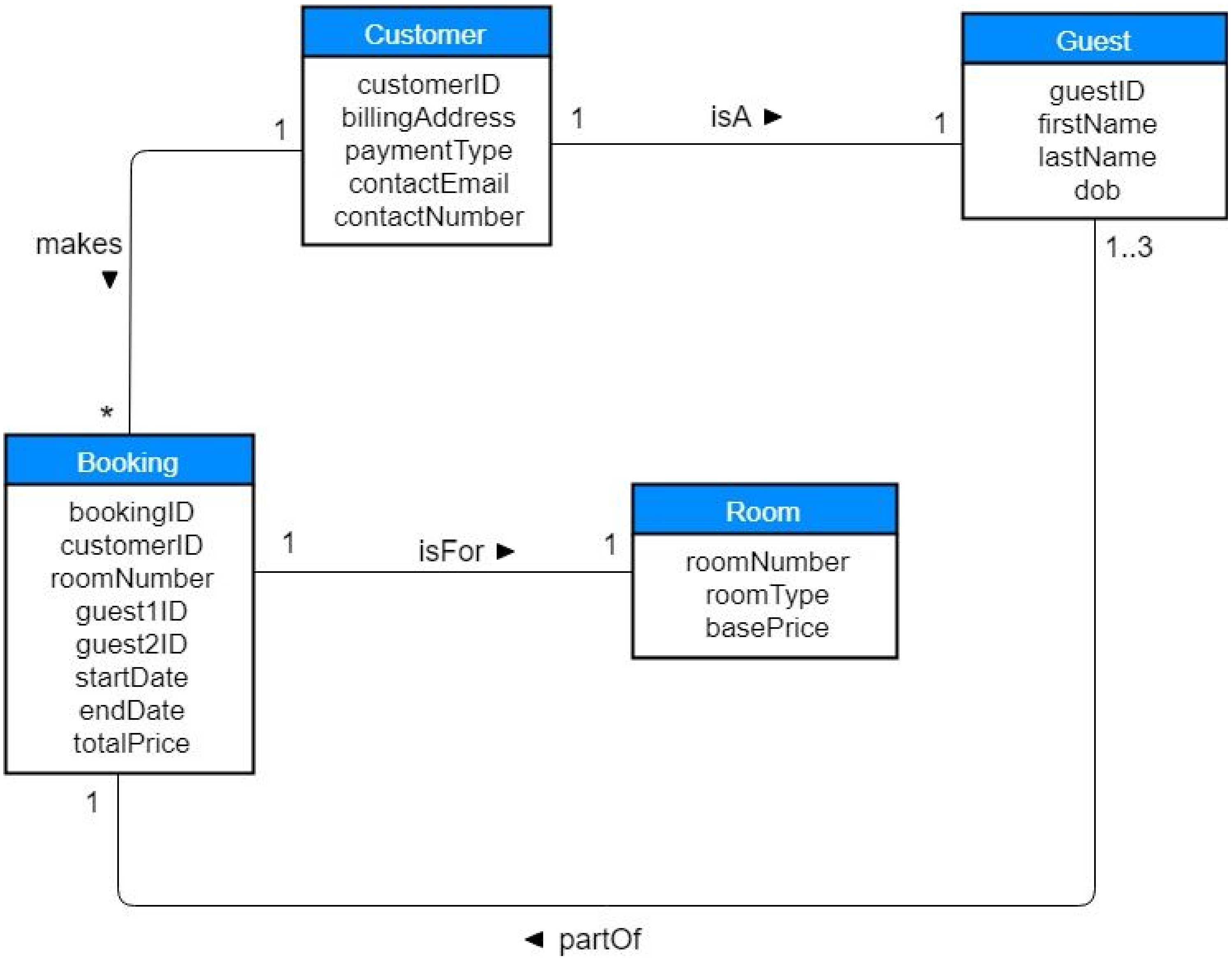
Customers may book accommodation for a particular type of room for a single period or may make several bookings for several visits to the hotel. For each booking they would specify the names of the people who would be staying. Thus a person may wish to make a booking for themselves alone, or with a friend or possibly with a friend and child. In any case, no more than three people may share a room. The hotel assigns these people to an appropriate available room.

Each type of room has a basic price and the cost of the room with several guests is calculated from this basic price, the type of room, and the number of guests sharing the room. Your system must be able to support the storage of the above information, including being able to track the guests included on a particular booking, and the guests allocated to each hotel room.

## Assumptions

- A Customer cannot book more than one room for a single period.
- Bookings cannot overlap; the start date of a second booking by the same customer must be after the end date of the first booking.
- A Customer will stay in the same room as the guests he has booked the room for.
- A Customer is a Guest.
- A Customer's customerID is essentially acting as a 'guest0ID' as they also stay in the same room as their guests but the difference is, they pay for the room.
- If there are less than three guests, the corresponding IDs will be set to null.
- A Customer's customerID will be the same as their guestID as they are a Guest.
- A booking has to be made for the entire day.
- The room type of every room number on one floor will be the same as the other floors.
- Customers can only have one constant billingAddress, contactNumber and contactEmail, these cannot change with various bookings.
- Guests are part of a single booking.
- All customerID's are a guestID's but not all guestID's will have a matching customerID (its only one a one way relation as not all guests will be customers).

**Conceptual Schema (ER-Diagram)**



## Relational Schema

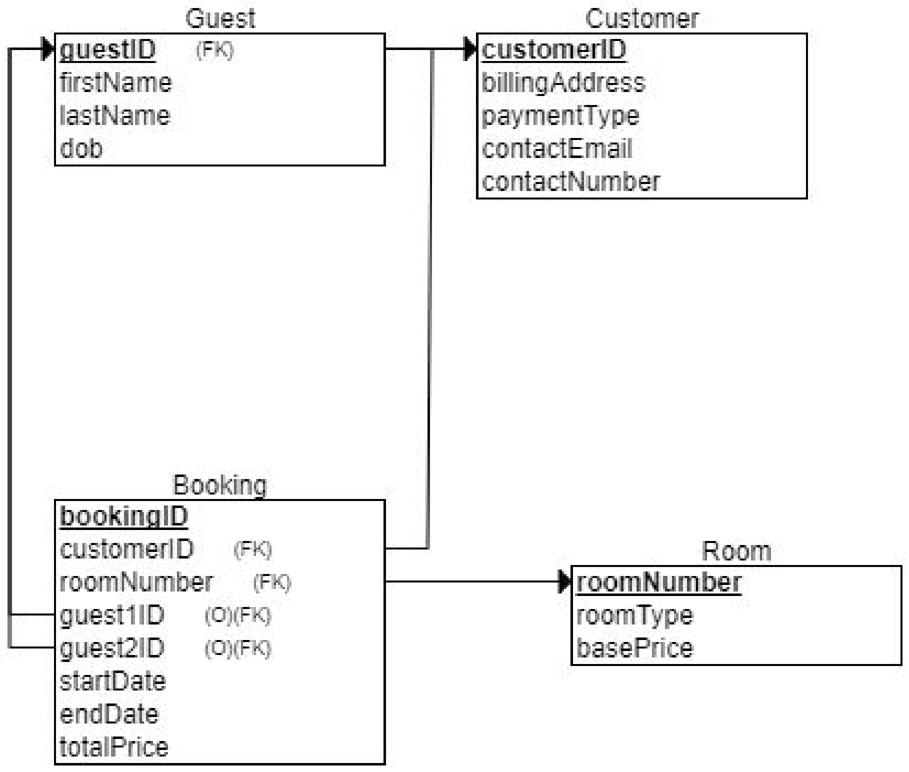
**Key:**
Primary Key = Underline
Foreign Key = *Italics*

**BOOKING**(bookingID, *customerID*, *roomNumber, guest1ID, guest2ID,* startDate, endDate, totalPrice)

**CUSTOMER**(customerID, billingAddress, paymentType, contactEmail, contactNumber)

**GUEST**(guestID, firstName, lastName, dob)

**ROOM**(roomNumber, roomType, basePrice)



Firstly, we identified the primary keys in each entity by determining which attribute can uniquely identify the tuples in the respective relation. We then identified the foreign keys by observing how each entity's attributes may relate to other entity's attributes and also how the primary keys in one table are used in other tables as foreign keys.

**Binary 1-1 Relationships**

In our ER diagram, we had two 1-1 relationships, one between Customer and Guest and another between Booking and Room. For the latter relationship, “roomNumber” is a primary key in table Room and a foreign key in table Booking. For the former, as a customerID is a guestID (as stated in the assumptions list) but not all guestIDs will have a customerID, the field guestID in table Guest is both a primary and a foreign key (to customerID in table Customer).

**Binary 1-N Relationship**

In our ER diagram, we have two 1-N relationships, one between Customer and Booking and the other between Booking and Guest. For the former relationship, the ‘N’ side is the Booking table, which holds the foreign key customerID. For the latter relationship the ‘N’ side is Guest, however, we decided to put the foreign keys inside the Booking entity as putting it in guest would mean each time the guest comes back we would need to change the guest tuple. While inside booking it means that every unique booking will be linked to the guests, so no changes need to be made as all data will represented by adding a new entry to the Booking table.

## Normalisation

### Key:

| | | |
|---|---|---|
| **Primary Key** | ***Foreign Key*** | Normal Table Entry (Entity Attribute) |

We began by making a table headers from each entity from the ER diagram into tables. This is what we produced:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **bookingID** | ***customerID*** | startDate | endDate | ***roomNumber*** | ***Guest1ID*** | ***Guest2ID*** | totalPrice |

| | | | | |
|---|---|---|---|---|
| **customerID** | paymentType | billingAddress | contactNumber | contactEmail |

| | | | |
|---|---|---|---|
| **guestID** | firstName | lastName | dob |

| | | |
|---|---|---|
| **roomNumber** | roomType | basePrice |

### First Normal Form

From this we could see that:

- All our values are atomic (single valued).
- All entries in a given column are of the same type.
- All rows are uniquely identified by primary keys.
- There are no multi-valued attributes.

Hence, we found that are design was already in first normal form, so no changes were required.

### Second Normal Form

From our tables, we could see that:

- Every relation (table) only had a single primary key

This means that every other attribute must be functionally dependent on this primary key, by definition. As a result, this meant it was already in second normal form, so no changes were required.

## Third Normal Form

For third normal form, we need to remove:

- Any transitive dependencies

In our tables, we could see that “basePrice” was dependent on “roomType” which depends on “roomNumber”. To resolve this, we removed “basePrice” from that table and introduced a new table with “roomType” as the primary key and “basePrice” as the other entry. So our table headers now looked like:

| **bookingID** | ***customerID*** | startDate | endDate | ***roomNumber*** | ***Guest1ID*** | ***Guest2ID*** | totalPrice |
|---|---|---|---|---|---|---|---|

| **customerID** | paymentType | billingAddress | contactNumber | contactEmail |
|---|---|---|---|---|

| **guestID** | firstName | lastName | dob |
|---|---|---|---|

| **roomNumber** | ***roomType*** |
|---|---|

| **roomType** | basePrice |
|---|---|

## Boyce-Codd Normal Form

Our tables already :

- Had no partial dependencies and no transitive dependencies
- Had every determinant as a candidate key

So no changes were required.

## Fourth Normal Form

For fourth normal form, we have to eliminate any multivalued dependencies and since in our assumptions we assumed a customer could only have one “billingAddress”, “contactNumber” and “contactEmail”, the only thing they could have multiple of would be “paymentTyp” (cash or card number). So instead of having lots of repeated data, we moved “paymentType” out of the table. We then put it in another table with just “customerID” as the primary key. This avoids data redundancy and spurious information. Our final table design now looks like:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **bookingID** | ***customerID*** | startDate | endDate | ***roomNumber*** | ***Guest1ID*** | ***Guest2ID*** | totalPrice |

| | | | |
|---|---|---|---|
| **customerID** | billingAddress | contactNumber | contactEmail |

| | |
|---|---|
| **customerID** | paymentType |

| | | | |
|---|---|---|---|
| **guestID** | firstName | lastName | dob |

| | |
|---|---|
| **roomNumber** | ***roomType*** |

| | |
|---|---|
| **roomType** | basePrice |