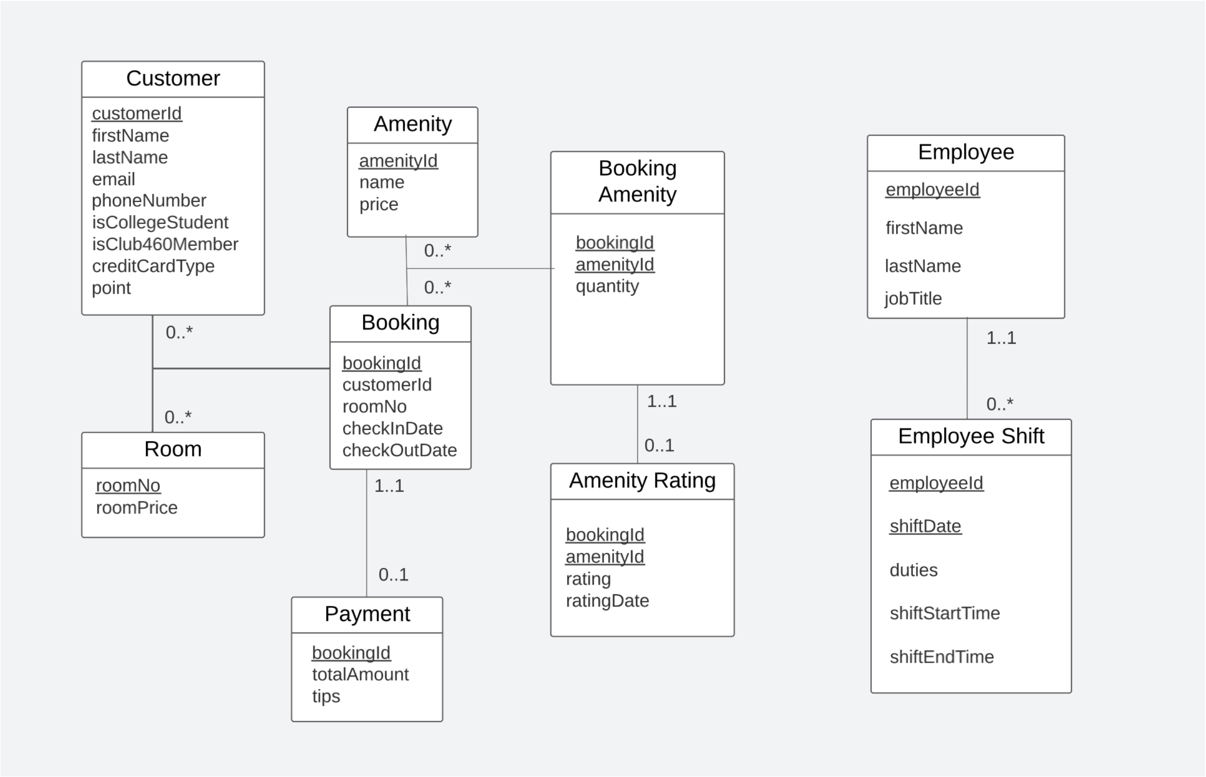
Minh Duong & Soumay Agarwal - CSC 460

Design Documentation

1. Conceptual Schema

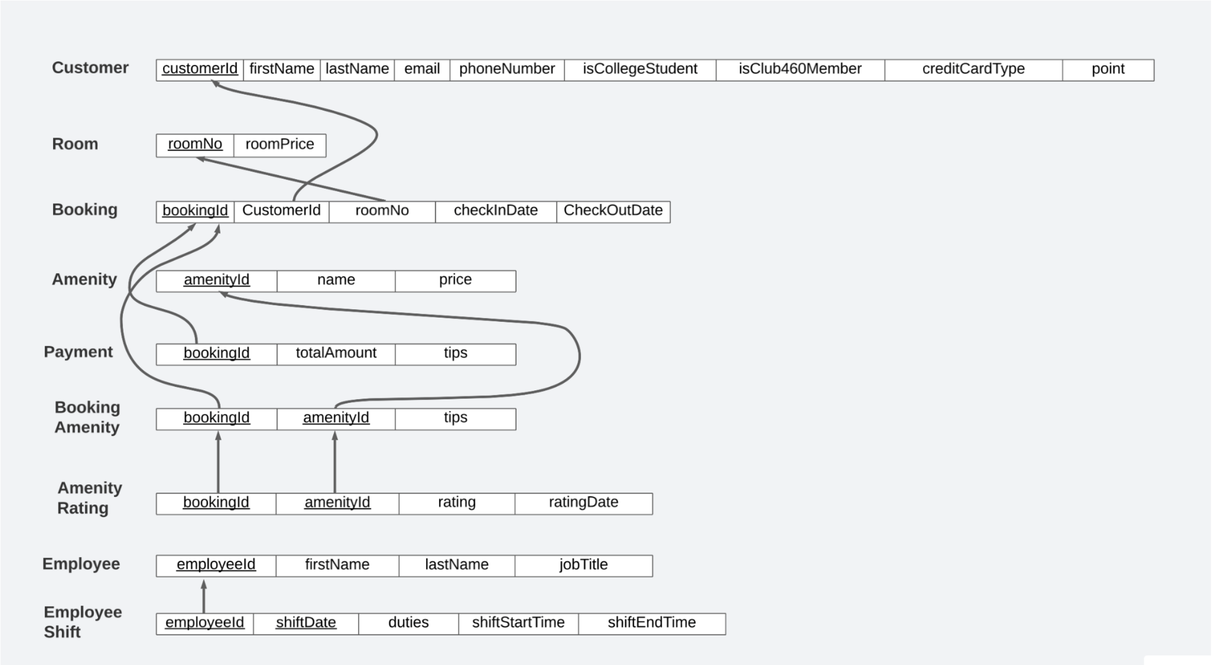


Here we can notice that EmployeeShift, AmenityRating and Payment are all **weak entities** since each of these relations are dependent on Employee, BookingAmenity and Booking respectively.

Additionally, the rating and ratingDate of AmenityRating will also not be able to be null, as if there is a rating then there must be a ratingDate, and vice versa. If both rating and ratingDate is null, then such tuple of AmenityRating should not exist.

Regarding the Booking, we initially went with a primary key of {roomNo, customerId, checkInDate}. However, this set of attributes also do not identify the booking: the customer can change the check in date, while it is still the same booking. Even if we go with checkInDate being uneditable (which makes changing booking unnecessarily complicated), then these three attributes also do not follow the 2nd Normal Form: the checkOutDate is fully functionally determined by {roomNo, checkInDate}. Thus we believe that having an artificial Primary Key here is necessary, thus we gave the artificial bookingId be the primary key for the relation.

2. Relational Schema (next page)



Customer: customerId | firstName | lastName | email | phoneNumber | isCollegeStudent | isClub460Member | creditCardType | point

Room: roomNo | roomPrice

Booking: bookingId | CustomerId | roomNo | checkInDate | CheckOutDate

Amenity: amenityId | name | price

Payment: bookingId | totalAmount | tips

Booking Amenity: bookingId | amenityId | tips

Amenity Rating: bookingId | amenityId | rating | ratingDate

Employee: employeeId | firstName | lastName | jobTitle

Employee Shift: employeeId | shiftDate | duties | shiftStartTime | shiftEndTime

3. Normalization Analysis

a. Customer

- Functional Dependencies:
  + customerId → firstName
  + customerId → lastName
  + customerId → email
  + customerId → phoneNumber
  + customerId → isCollegeStudent
  + customerId → isClub460Member
  + customerId → creditCardType
  + customerId → point
- All atributes are not set-valued (1st Normal Form)
- All non-prime attributes (every attributes except for customerId) are fully functionally dependent on the customerId, which is the Primary Key and the only Candidate Key (2nd Normal Form)
- All Functional Dependencies have the left-hand side being a set of only customerId, which is a superset (3rd Normal Form and Boyce-Codd Normal Form)

b. Room

- Functional Dependencies
  + roomNo → roomPrice
- All atributes are not set-valued (1st Normal Form)
- The only non-prime attribute, roomPrice, is fully functionally dependent on the roomNo, which is the Primary Key and also the only Candidate Key (2nd Normal Form)
- The only functional dependency we have has the left-hand side being roomNo, which is the Primary Key and it is a superset (3rd Normal Form and Boyce-Codd Normal Form).

c. Booking

- Functional Dependencies:

  - bookingId → roomNo
  - bookingId → customerId
  - bookingId → checkInDate
  - bookingId → checkOutDate
- All attributes are not set-valued (1st Normal form)
- All non-prime attributes (every attributes except for bookingId) are fully functionally dependent on the bookingId, which is the Primary Key and the only Candidate Key (2nd Normal Form)
- All Functional Dependencies have the left-hand side being a set of only bookingId, which is a superset (3rd Normal Form and Boyce-Codd Normal Form)

d. Payment
- Functional Dependencies:
  - bookingId → totalAmount
  - bookingId → tips
- All attributes are not set-valued (1st Normal form)
- All non-prime attributes (every attributes except for bookingId) are fully functionally dependent on the bookingId, which is the Primary Key and the only Candidate Key (2nd Normal Form)
- All Functional Dependencies have the left-hand side being a set of only bookingId, which is a superset (3rd Normal Form and Boyce-Codd Normal Form)
- Note: This relation is a weak entity with the “parent” entity of Booking.

e. Amenity
- Functional Dependencies:
  - amenityId → amenityName
  - amenityId → price
- All attributes are not set-valued (1st Normal form)
- All non-prime attributes (every attributes except for amenityId) are fully functionally dependent on the bookingId, which is the Primary Key and the only Candidate Key (2nd Normal Form)
- All Functional Dependencies have the left-hand side being a set of only amenityId, which is a superset (3rd Normal Form and Boyce-Codd Normal Form)

f. BookingAmenity
- Functional Dependencies:
  - {bookingId, amenityId} → quantity
- All attributes are not set-valued (1st Normal form)
- The only non-prime attribute of quantity is fully functionally dependent on {bookingId, amenityId}, which is the only Candidate Key (2nd Normal form)
- The only functional dependency we have has the left-hand side containing all attributes of the Primary Key, which means it is a superset (3rd Normal Form and Boyce-Codd Normal Form)

g. AmenityRating
- Functional Dependencies:
  - {bookingId, amenityId} → rating
  - {bookingId, amenityId} → ratingDate
- All attributes are not set-valued (1st Normal form)
- The two non-prime attributes of rating and ratingDate are both fully functionally dependent on {bookingId, amenityId}, which is the set of all Candidate Key (2nd Normal form)

- Both of the functional dependencies we have has the left-hand side containing all attributes of the Primary Key, which means it is a superset (3rd Normal Form and Boyce-Codd Normal Form)
- Note: This relation is a weak entity with the "parent" entity of BookingAmenity.

h. Employee
- Functional Dependencies:
  + employeeId → firstName
  + employeeId → lastName
  + employeeId → jobTitle
- All attributes are not set-valued (1st Normal form)
- All oof non-prime attributes are fully functionally dependent on employeeId, which is the only Primary Key (and Candidate Key) (2nd Normal form)
- All functional dependencies we have has the left-hand side of employeeId, which is the only Primary Key and thus a superset (3rd Normal Form and Boyce-Codd Normal Form)

i. EmployeeShift
- Functional Dependencies:
  + {employeeId, shiftDate} → duties
  + {employeeId, shiftDate} → shiftStartTime
  + {employeeId, shiftDate} → shiftStartTime
- All attributes are not set-valued (1st Normal form)
- All oof non-prime attributes are fully functionally dependent on the set of {employeeId, shiftDate}, which is the Candidate Key (2nd Normal Form)
- All functional dependencies we have has the left-hand side of {employeeId, shiftDate}, which is a superset (3rd Normal Form and Boyce-Codd Normal Form)
- Note: This relation is a weak entity with the "parent" entity of Employee

In conclusion, from the Normalization Analysis, we can observe that all of the relations reach Boyce-Codd Normal Form.

4. Query Description

For the custom query, we go with "Finding the first names, last names, and the shift duties of all employees in a single (given) day, at a specific time frame (the start time and end time are given by the user)".

Utility: This query can be very helpful if there is any incident that happens in a particular day in the past, and the hotel manager needs to check for all of the employees in the hotel at that time and what they are doing.

Technical Aspect: This query uses both Employee and EmployeeShift relations. To support this query, we store the time information (of the day) as an integer, which is the value of the number of seconds from the start of the day. The reason is because Oracle does not have any specific type for the time of day, and if we use varchar then we would not be able to do support this query of time value. The frontend will take care of translating the input into time value stored in the database, and vice versa. The query then becomes a range query.