CSI2132 – Databases I
Assignment 1 – February 16th 2023
Samuel Harrison – 300 198 650

ER Diagram and Schema

See the Visio PDF at the end of the report.

Main Considerations:

- Emails and Phone numbers are kept in separate tables since PostgreSQL arrays are not a form of normalized data and their use should be limited
- Ratings are kept as attributes for hotels, but an argument can be made that these would be better fit for a separate table in case that this application ever supports customers leaving ratings. However, this would require a DB migration anyways so not much is to be gained from doing this now versus then.
- Managers are kept track of using a Join Table which relates employees to hotels.
- Defects and amenities are stored in their own table, and are associated to rooms using a Join Table
- The view that a room has is standardized using an ENUM containing “sea”, “mountain”, “city”
- All dates are stores as date types rather than timestamp types as the added precision is not needed and this reduces the disk usage by 12 bytes per booking record and 8 bytes per renting record
- Extendable is stored as a Boolean, and the added space from being extendable will not counted in the capacity field
- All of the PK and FK choices are explained below

Primary Keys

In designing the database schema, I’ve chosen to use UUIDs for primary keys instead of SSNs. SSNs are classified as sensitive personal information, and their handling is subject to strict regulatory requirements and privacy laws. These should never be used unless absolutely necessary. As for other composite key choices, it is strongly discouraged to rely on any customer (or employee) facing forms as such fields are inherently risky. This data is prone to typos, duplication, and general human error (malicious or not). Given UUID’s are only 16 bytes and will absolutely assure uniqueness and data integrity, protect the database from 1 in a million events and from human error, it is an absolute no-brainer to use these everywhere.

- HotelChains PK: chainID (uuid)
- Hotels PK: hotelID (uuid)
- Rooms PK: roomID (uuid)
- Rentings PK: rentingID (uuid)
- Bookings PK: bookingID (uuid)
- Customers PK: customerID (uuid)
- Employees PK: employeeID (uuid)
- Manages (Join Table) PK: Composite Key (hotelID, employeeID) (uuids)
- Defects PK: defectID (uuid)
- Amenities PK: amenityID (uuid)
- HasAmenity (Join Table) PK: Composite Key (roomID, amenityID) (uuids)
- HasDefect (Join Table) PK: Composite Key (roomID, defectID) (uuids)
- PhoneNumbers PK: phoneID (uuid)

- EmailAddresses PK: emailID (uuid)

Foreign Keys for Referential Integrity

- *Hotels.chainID* references *HotelChains.chainID*
- *Rooms.hotelID* references *Hotels.hotelID*
- *Rentings.customerID* references *Customers.customerID*
- *Rentings.employeeID* references *Employees.employeeID*
- *Rentings.roomID* references *Rooms.roomID*
- *Bookings.customerID* references *Customers.customerID*
- *Bookings.roomID* references *Rooms.roomID*
- *Manages.employeeID* references *Employees.employeeID*
- *Manages.hotelID* references *Hotels.hotelID*
- *HasAmenity.roomID* references *Rooms.roomID*
- *HasAmenity.amenityID* references *Amenities.amenityID*
- *HasDefect.roomID* references *Rooms.roomID*
- *HasDefect.defectID* references *Defects.defectID*
- *PhoneNumbers.origin* references *HotelChains.chainID or Hotels.hotelID*
- *EmailAddresses.origin* references *HotelChains.chainID or Hotels.hotelID*

To ensure the integrity and consistency of our database, cascading deleting rules that automatically propagate deletions across related entities should be implemented. Specifically, when a hotel chain is deleted, this action triggers the deletion of all its associated hotels, as well as all related emails and phone numbers, thereby maintaining clean and accurate data without orphaned records. Similarly, upon the deletion of a hotel, the database should automatically delete all its associated rooms and remove the corresponding manager entry from the Manages table. It is however mentioned that archives are kept of Bookings and Rentings so these should not be deleted in the cascade even though the room is deleted.

Domain and Attribute Constraints

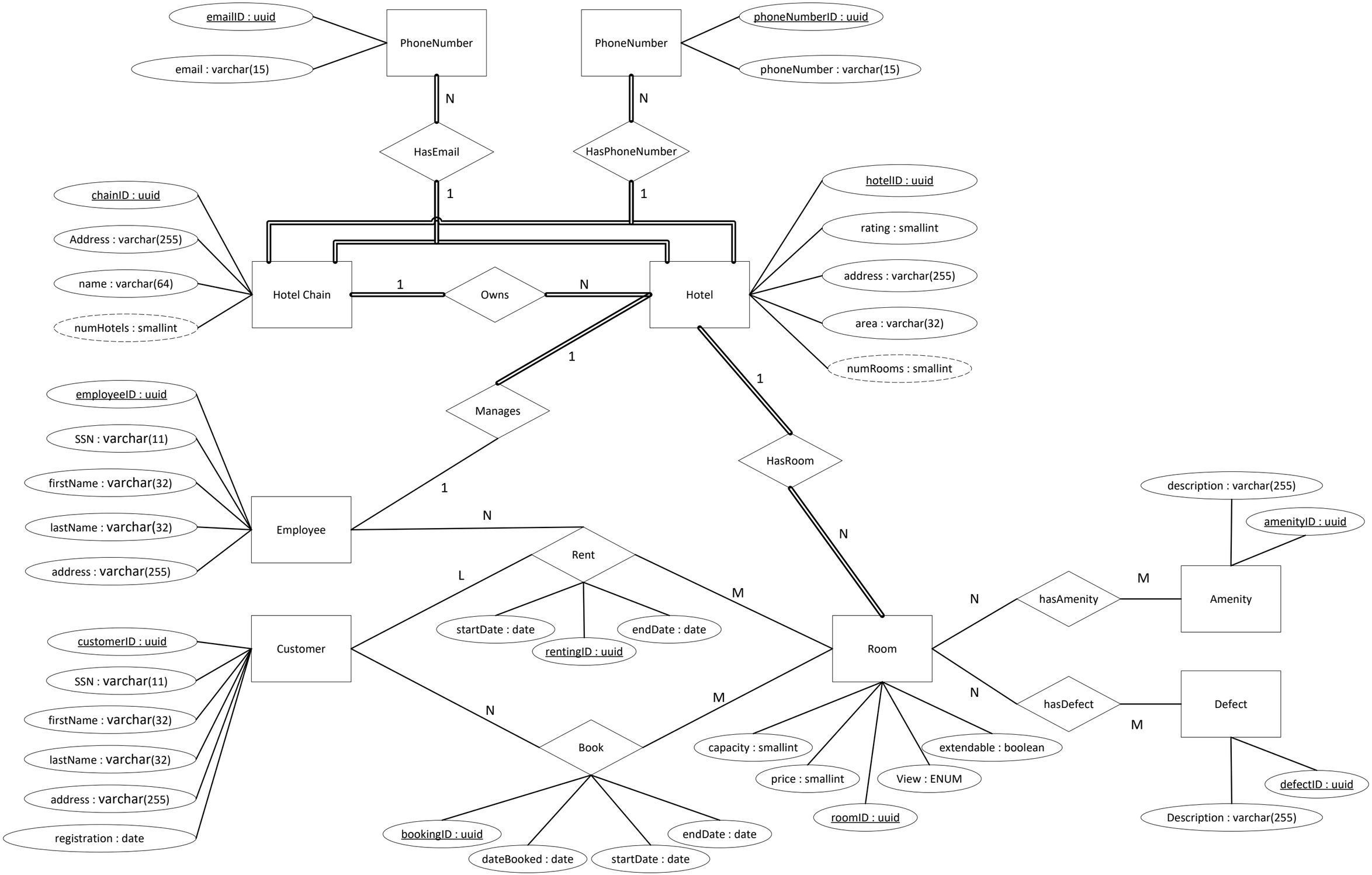
- The following should be greater than or equal to 0
    - HotelChain.numHotels
    - Hotel.numRooms
    - Hotel.price
    - Hotel.capacity
- view ENUM should be restricted to the predefined set of valid views ('sea', 'mountain', 'city')
- SSN should match the pattern of a valid SSN (regex)
- phone should match the pattern of a valid phone number (regex)
- email should match the pattern of a valid email address (regex)

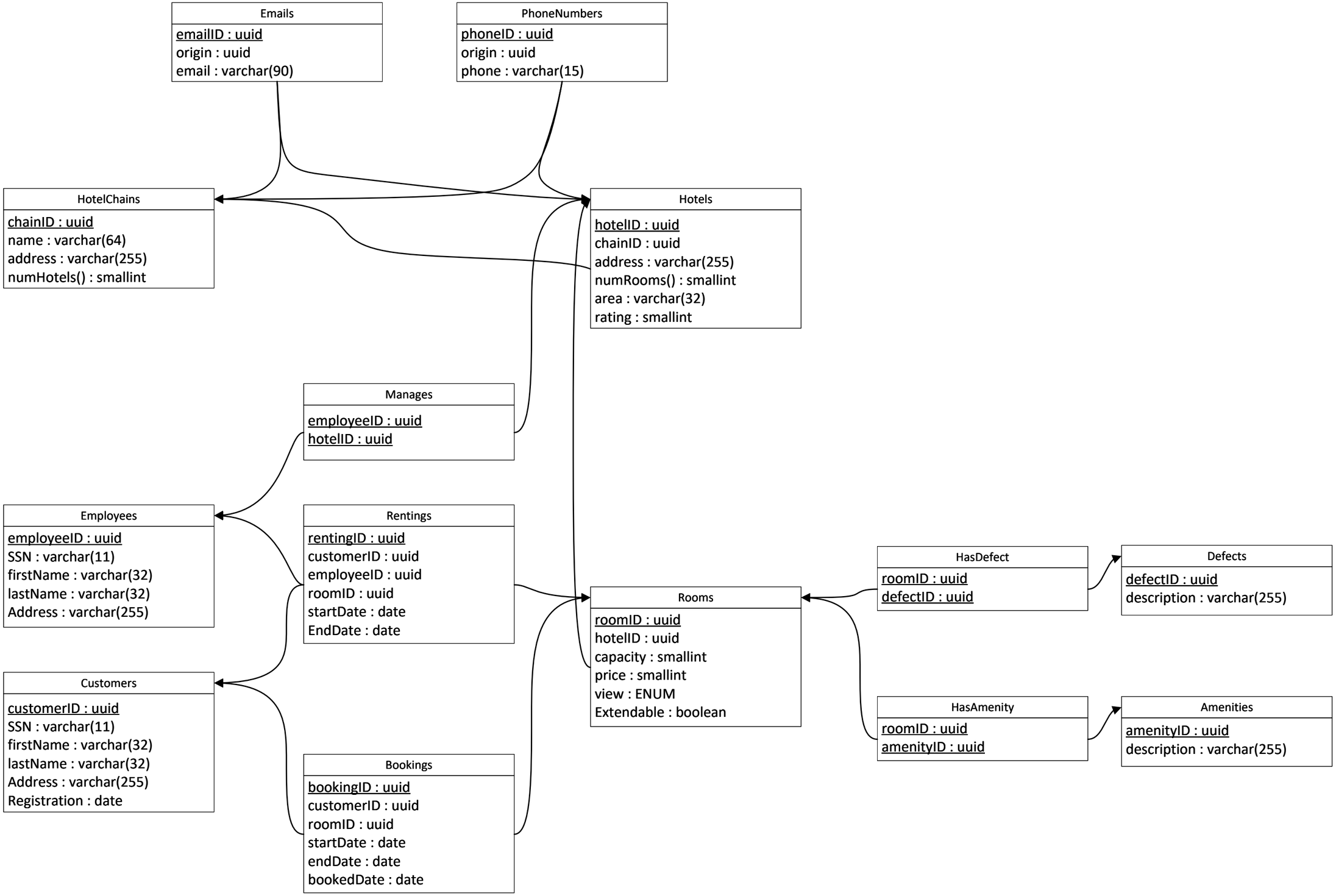
User-Defined Constraints

- An employee can only be assigned as a manager to one hotel at a time
- Every hotel must have at least one manager
- A room cannot have two Rentings with overlapping dates
- A room cannot have two Bookings with overlapping dates

Other Notes

- The length of each varchar() was determined based on heuristics from the PostgreSQL documentation and from various StackOverflow discussions
- A few important attributes were added, but not explicitly asked for
    - Hotel.area : Such that rooms can be filtered by area. Could equally have been stored as geospatial coordinates but this would require some PostgreSQL plugins which seem out of scope for the class
    - Booking.dateBooked : This would be a good piece of information to have a record of and only costs 4 bytes per record. Could be used to see average time booked in advance, and perhaps drive some optimization for price changes as the booking date approaches.


emailID : uuid
email : varchar(15)
PhoneNumber
PhoneNumber
phoneNumberID : uuid
phoneNumber : varchar(15)
N
N
HasEmail
HasPhoneNumber
1
1
chainID : uuid
Address : varchar(255)
name : varchar(64)
numHotels : smallint
Hotel Chain
1
Owns
N
Hotel
hotelID : uuid
rating : smallint
address : varchar(255)
area : varchar(32)
numRooms : smallint
1
1
employeeID : uuid
SSN : varchar(11)
firstName : varchar(32)
lastName : varchar(32)
address : varchar(255)
Manages
1
HasRoom
N
description : varchar(255)
amenityID : uuid
Employee
N
Rent
L
M
N
hasAmenity
M
Amenity
startDate : date
endDate : date
rentingID : uuid
customerID : uuid
SSN : varchar(11)
firstName : varchar(32)
lastName : varchar(32)
address : varchar(255)
registration : date
Customer
Room
N
hasDefect
M
Defect
M
N
Book
capacity : smallint
price : smallint
View : ENUM
extendable : boolean
roomID : uuid
defectID : uuid
Description : varchar(255)
bookingID : uuid
dateBooked : date
startDate : date
endDate : date


Emails
emailID : uuid
origin : uuid
email : varchar(90)
PhoneNumbers
phoneID : uuid
origin : uuid
phone : varchar(15)
HotelChains
chainID : uuid
name : varchar(64)
address : varchar(255)
numHotels() : smallint
Hotels
hotelID : uuid
chainID : uuid
address : varchar(255)
numRooms() : smallint
area : varchar(32)
rating : smallint
Manages
employeeID : uuid
hotelID : uuid
Employees
employeeID : uuid
SSN : varchar(11)
firstName : varchar(32)
lastName : varchar(32)
Address : varchar(255)
Rentings
rentingID : uuid
customerID : uuid
employeeID : uuid
roomID : uuid
startDate : date
EndDate : date
Customers
customerID : uuid
SSN : varchar(11)
firstName : varchar(32)
lastName : varchar(32)
Address : varchar(255)
Registration : date
Bookings
bookingID : uuid
customerID : uuid
roomID : uuid
startDate : date
endDate : date
bookedDate : date
Rooms
roomID : uuid
hotelID : uuid
capacity : smallint
price : smallint
view : ENUM
Extendable : boolean
HasDefect
roomID : uuid
defectID : uuid
Defects
defectID : uuid
description : varchar(255)
HasAmenity
roomID : uuid
amenityID : uuid
Amenities
amenityID : uuid
description : varchar(255)