CSC 440 Course Project

WolfVillas Hotel Database

Grayson Jones - gcjones

Rodolphe Njiwa Sandjong - rnjiwas

Julie Persinger - jcpersin

Tsu-Hsin Yeh - tyeh3

CSC 440

Professor Chirkova

Fall 2016

Assumptions

1. The customer can only process check-in through a front desk representative (no online purchases, only over the phone or in person with a front desk representative).
2. When the customer is checked in, the start/end date, check-in/check-out time should be specified (customer cannot change their mind for staying longer or leave earlier).
3. Given room occupancy and room type, there is only one nightly rate associated with room of these type combination.
4. If a group of customers check in, then only one customer needs to be associated with check-in information and billing account. Meanwhile, only one customer is responsible for the payment.
5. Total expenses during a stay = phone bill + laundry bill + restaurant bill + (nightly rate * number of days staying)
6. Front desk staff will assign catering staff and service staff to a specific set of check in information if customers request to stay in a presidential room.
7. Service records are associated with a specific check-in instead of the customer they are serving (If person A, B and C check in as a group, the services availed for A or B or C will generate service records for check-in which represents the entire group).
8. For both customer and staff relations, we decided to make ssn unique but not a primary key because we think ssn is sensitive user information and should not be used for condition and join. A uniquely generated id is used in place of ssn as the primary key.
9. The payment method customers choose is either cash or credit card.
10. A hotel can only have one phone number and a phone number can only be for one hotel location.
11. Customers can share phone and address. (e.g. family members or roommates may reveal common information to the hotel)

1) Database Schema

hotel(id, name, address, phone)

1. **id -> name, address, phone** holds because every hotel has a name, address, and phone number that can be identified by the id, the primary key.
2. **address -> id, address -> name,** and **address -> phone** hold because it is not possible to have more than one hotel at the same address.
3. **phone -> address, phone -> name, and phone -> id** hold because a phone number can only be used at one hotel
4. **name -> id, name -> phone, or name -> address** does not hold because it is possible to have multiple hotels with the same name.

This relation is in BCNF (and thus 3NF) because the left hand side of every Functional Dependency that holds is a super key.

staff(id, ssn, name, age, gender, title, department, phone, address, hotel_id)

1. **id -> ssn, name, age, gender, title, department, phone, address, hotel_id** holds because id serves to uniquely identify every staff and thus id determine all personal attributes. (personal attributes refers to ssn, name, age, gender, title, department, phone, address, hotel_id)
2. **ssn -> id, name, age, gender, title, department, phone, address, hotel_id** holds because ssn can uniquely identified every staff and thus ssn determine all personal attributes.
3. **name -> id, ssn, age, gender, title, department, phone, address, hotel_id** does not hold because different staff can have same name and thus name cannot determine any personal attributes.
4. **age -> id, ssn, name, gender, title, department, phone, address, hotel_id** does not hold because different staff can have same age and thus age cannot determine any personal attributes.
5. **gender-> id, ssn, name, age, title, department, phone, address, hotel_id** does not hold because different staff can have same gender and thus gender cannot determine any personal attributes.
6. **title -> id, ssn, name, age, gender, department, phone, address, hotel_id** does not hold because different staff can have same title and thus title cannot determine any personal attributes.
7. **department -> id, ssn, name, age, gender, title, phone, address, hotel_id** does not hold because different staff can have same department and thus department cannot determine any personal attributes.
8. **phone -> id, ssn, name, age, gender, title, department, address, hotel_id** does not hold because different staff can have the same phone number (we assume that staff can share phone number) and phone cannot uniquely determine all personal attributes.
9. **address -> id, ssn, name, age, gender, title, department, phone, hotel_id** does not hold because different staff can have the same address (we assume that staff can share address) and thus address cannot uniquely determine all personal attributes.
10. **hotel_id -> id, ssn, name, age, gender, title, department, phone, address** does not hold because different staff can have same hotel they work in and thus hotel_id cannot determine any personal attributes.

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

manager(id, hotel_id)

id -> hotel_id holds because each manager can take charge of only one hotel.

The relation with two attributes is in BCNF (and thus 3NF).

frontdesk_staff(id)

id -> id

The relation with only keys as its attribute is in BCNF (and thus 3NF).

billing_staff(id)

id -> id

The relation with only keys as its attribute is in BCNF (and thus 3NF).

catering_staff(id)

id -> id

The relation with only keys as its attribute is in BCNF (and thus 3NF).

service_staff(id)

id -> id

The relation with only keys as its attribute is in BCNF (and thus 3NF).

room(room_number, category, occupancy, availability, hotel_id)

1. **room_number , hotel_id -> category, occupancy, availability** holds because each room is uniquely identify by a combination of room_number and the id of the hotel in which that room is located.
2. **room_number->category, room_number->occupancy, room_number->availability** does not hold because two rooms with the same number that are located in two different hotels do not necessarily have the same occupancy or availability.
3. **category->occupancy, category->availability** does not hold because rooms with the same category do not necessarily have the same occupancy or availability.

4. **occupancy->category, occupancy->availability** does not hold because rooms with the same occupancy do not necessarily have the same category or availability.
5. **availability->category, availability->occupancy** does not hold because rooms with the same availability do not necessarily have the same category or occupancy.

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

room_type(<u>category</u>, <u>occupancy</u>, nightly_rate)

1. **category, occupancy -> nightly_rate holds** because the room category and room occupancy determine the nightly rate for the room.
2. **nightly_rate -> category or nightly_rate -> occupancy** does not hold because you need to know both the category and the occupancy in order to know the nightly rate.
3. **category -> nightly_rate** does not hold because you need to also know the occupancy in order to know the nightly rate.
4. **occupancy -> nightly_rate** does not hold because you need to also know the room category in order to know the nightly rate.

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

checkin(<u>id</u>, current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)

1. **id -> current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** holds because id serves to uniquely identify every check-in information entry and thus id determines all check-in attributes. (check-in attributes refer to current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
2. **current_occupancy -> id, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same current occupancy and thus current occupancy cannot determine any check-in attributes.
3. **start_date -> id, current_occupancy, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same start date and thus start date cannot determine any check-in attributes.

4. **end_date -> id, current_occupancy, start_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same end date and thus end date cannot determine any check-in attributes.
5. **start_time -> id, current_occupancy, start_date, end_date, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same start time and thus start time cannot determine any check-in attributes.
6. **end_time -> id, current_occupancy, start_date, end_date, start_time, cstaff_id, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same end time and thus end time cannot determine any check-in attributes.
7. **cstaff_id -> id, current_occupancy, start_date, end_date, start_time, end_time, sstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same catering staff associated with it and thus cstaff_id cannot determine any check-in attributes.
8. **sstaff_id -> id, current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, fdstaff_id, customer_id** does not hold because different check-in information entries can have the same service staff associated with it and thus sstaff_id cannot determine any check-in attributes.
9. **fdstaff_id -> id, current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, customer_id** does not hold because different check-in information entries can have the same front desk staff associated with it and thus cstaff_id cannot determine any check-in attributes.
10. **customer_id -> id, current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, customer_id** does not hold because different check-in information entries can have the same customer associated with it and thus customer_id cannot determine any check-in attributes.

This relation is in BCNF (and thus 3NF) because the left hand side of every functional dependency that holds is a super key.

service_record(<u>id</u>, type, amount, checkin_id)

1. **id -> type, amount, checkin_id** holds because each service record has a type, amount, and checkin_id which can all be uniquely identified by id.
2. **type -> amount, checkin_id, id** does not hold because different service records can have the same type and different amount and check-in associated with it.
3. **amount -> type, checkin_id, id** does not hold because different service records can have same amount and different type and check-in information associated with it.

4. **checkin_id -> type, amount, id** does not hold because different service records can have same check-in information associated and have different type or amount.
5. **type, amount -> checkin_id** does not hold because the same checkin_id can have multiples of the same type and amount

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

restaurant_record(id, cstaff_id)

1. **id -> cstaff_id** holds because each restaurant record is uniquely identified by id.
2. **cstaff_id -> id** does not hold because same catering staff can generate different restaurant records and thus different id.

The relation with two attributes is in BCNF (and thus 3NF).

phone_record(id, sstaff_id)

1. **id -> sstaff_id** holds because each phone record is uniquely identified by id.
2. **sstaff_id -> id** does not hold because same service staff can generate different phone records and thus different id.

The relation with two attributes is in BCNF (and thus 3NF).

laundry_record(id, sstaff_id)

1. **id -> sstaff_id** holds because each laundry record is uniquely identified by id.
2. **sstaff_id -> id** does not hold because same service staff can generate different laundry records and thus different id.

The relation with two attributes is in BCNF (and thus 3NF).

customer(id, name, ssn, gender, phone, email, address)

1. **id -> name, ssn, gender, phone, email, address** holds because each customer is uniquely identified by id and thus id determines all customer attributes (customer attributes refer to name, ssn, gender, phone, email, address).
2. **ssn -> id, name, gender, phone, email, address** holds because ssn can uniquely identify every customer and thus ssn determine all customer attributes.

3. **name -> id, ssn, gender, phone, email, address** does not hold because different customer can have same name and thus name cannot determine customer attributes.
4. **gender -> id, name, ssn, phone, email, address** does not hold because different customer can have same gender and thus gender cannot determine customer attributes.
5. **phone -> id, name, ssn, gender, email, address** does not hold because different customer can have same phone and thus phone cannot determine customer attributes.
6. **email -> id, name, ssn, gender, phone, address** does not hold because different customer can have same email and thus email cannot determine customer attributes.
7. **address -> id, name, ssn, gender, phone, email** does not hold because different customer can have same address and thus address cannot determine customer attributes.

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

billing(id, billing_addr, payment_method, card_number, customer_id, bstaff_id)

1. **id->billing_addr, payment_method, card_number, customer_id, bstaff_id** holds because each bill is uniquely identified by its id and is associated with billing_addr, payment_method, card_number, customer_id**.**
2. **customer_id->id, billing_addr, payment_method, card_number, bstaff_id** holds because each customer is assigned a billing with a billing_id, since billing_id functionally determine billing_addr, payment_method, and card_number, so does **customer_id;**
3. No combination of **billing_addr, payment_method, or card_number** can functionally determine (billing) id because two Billings might be from customer living at the same address, having the same payment method (e.g. credit card) and using the same card_number (e.g.: joint credit card account).
4. No combination of **billing_addr, payment_method, or card_number** can functionally determine customer_id because two customers might live at the same address (e.g. a couple), have the same payment method (e.g. credit card) and use the same card_number (e.g. joint credit card account).
5. **billing_addr->payment_method, and billing_addr->card_number** does not hold because customers can have the same address but not necessarily the same payment_method (debit/credit) or card_number.
6. **payment_method->billing_addr, and payment_method->card_number** doesn't hold because customers can have the same payment method but not necessarily the same billing_address (debit/credit) or card_number.

7. **card_number->billing_addr, and card_number->payment_method** does not hold because customers can use the same card (same card_number) but use different billing address and different payment methods (some cards can be used as credit or debit).

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

reserve_for(hotel_id, room_number, checkin_id)

1. **checkin_id -> hotel_id, room_number** holds because only one room reservation can be made in one hotel per check-in reservation.
2. **hotel_id - > checkin_id, room_number** does not hold because different check-in information can be associated with same hotel and different rooms can be reserved for the same hotel.
3. **room_number -> hotel_id, checkin_id** does not hold because different hotel with the same room number can be reserved and same room number within different hotel can be associated with different check-in information.
4. **hotel_id, room_number -> checkin_id** does not hold since the same room for specific hotel will be checked in/out multiple times and thus associated with different check-in id.

This relation is in BCNF (and thus 3NF) because for the left hand side for every FDs that hold is a super key.

2) Design Decisions

The technical approach to convert E/R entity to relation is described as following

1. For each entity, key(s) and attributes associated with the entity should be recorded in the relation.
2. For weak entity (specifically Room entity), the relation includes information to describe in part 1 plus the additional key(s) from the supporting relationship.
3. For many-many relationships, the key(s) for the entity associated with the relationship is included in the relation.
4. For many-one relationships (many-one relationship R from E to F), we can combine source entity and relationship into one relation that contains all the attributes and key(s) from source entity(E), attributes associated with relationship(R) and key(s) from target entity(F).

The method that converts subclass into relation follows the E/R viewpoint. The relation for subclass should contain its unique attributes as well as key(s) from its parent.

***hotel*(<u>id</u>, name, address, phone)**

- id (primary key) - unique identifier
- name (not null) - identification purpose
- address (not null) - every hotel requires an address
- phone (not null) - every hotel requires a phone number

***staff*(<u>id</u>, ssn, name, age, gender, title, department, phone, address, hotel_id)**

- id (primary key) - unique identifier
- ssn (not null) - unique to identify ssn for the staff
- name (not null) - to identify name for the staff
- age (not null) - to identify age for the staff
- gender (not null) - to identify the gender for the staff
- title (not null) - to identify the title for the staff
- department (not null) - to identify the department of the staff
- phone (not null) - to identify phone number of the staff
- address (not null) - to identify address of the staff

- hotel_id (not null, referential integrity) - refers to other entities within the database (***hotel***)

***manager*(id, hotel_id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities within the database (***staff***)
- hotel_id (not null, referential integrity) - refers to other entities within the database (***hotel***)

***frontdesk_staff*(id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***staff***)

***billing_staff*(id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***staff***)

***catering_staff*(id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***staff***)

***service_staff*(id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***staff***)

***room*(number, category, occupancy, availability, hotel_id)**

- number (primary key) - part of unique identifier and the number of the room
- hotel_id (primary key, referential integrity) - part of unique identifier and refers to other entities in the database (***hotel***)
- category (not null, referential integrity) - the category of the room and refers to other entities in the database (***room_type***)
- occupancy (not null, referential integrity) - maximum number people that can live in the room and refers to other entities in the database (***room_type***)

- availability (not null) - the status of whether the room is available

***room_type*(<u>category</u>, <u>occupancy</u>, nightly_rate)**

- category (primary key) the type of room(economy, presidential suite, etc.). Uniquely identifies the room type in tandem with the occupancy
- occupancy (primary key) the number of occupants that can be assigned to the room. Uniquely identifies the room type in tandem with the category
- nightly_rate (not null) the amount charged to the customer per night

***checkin*(<u>id</u>, current_occupancy, start_date, end_date, start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)**

- id (primary key) - unique identifier
- current_occupancy (not null) - the number of the customer stay in the room associated with check in
- start_date (not null) - the start date for check-in record
- end_date (not null) - the end date for check-in record
- start_time (not null) - the start time for check-in record
- end_time (not null) - the end time for check-in record
- cstaff_id (referential integrity) - the catering staff assigned to the check-in record and refers to other entities in the database (***catering_staff***). It can be null if the room associated with the record is not presidential suite
- sstaff_id (referential integrity) - the service staff assigned to the check-in record and refers to other entities in the database (***service_staff***). It can be null if the room associated with the record is not presidential suite
- fdstaff_id (not null, referential integrity) - the front desk staff who generates this check-in record and refers to other entities in the database (***frontdesk_staff***)
- customer_id (not null, referential integrity) - the representative customer associated with the check-in record and refers to other entities in the database (***customer***)

***service_record*(<u>id</u>, type, amount, checkin_id)**

- id (primary key) - unique identifier

- type (not null) - the type of the service record (restaurant, phone or laundry)
- amount (not null) - the amount of expense associated with the service record
- checkin_id (not null, referential integrity) - the check-in record associated with the service record and refers to other entities in the database (***checkin***)

***restaurant_record*(id, cstaff_id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***service_record***)
- cstaff_id (referential integrity) - catering staff who generates restaurant record and refers to other entities in the database (***catering_staff***)

***phone_record*(id, sstaff_id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***service_record***)
- sstaff_id (referential integrity) - service staff who generates phone record and refers to other entities in the database (***service_staff***)

***laundry_record*(id, sstaff_id)**

- id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***service_record***)
- sstaff_id (referential integrity) - service staff who generates laundry record and refers to other entities in the database (***service_staff***)

***customer*(id, name, ssn, gender, phone, email, address)**

- id (primary key) - unique identifier
- name (not null) - to identify the name of the customer
- ssn (not null) - to identify the ssn for every customer and ssn should be unique
- gender (not null) - to identify the gender for the customer
- phone (not null) - to identify the phone number for the customer
- email (not null) - to identify the email for the customer
- address (not null) - to identify the address for the customer

***billing*(id, billing_addr, payment_method, card_number, customer_id, bstaff_id)**

- id (primary key) - unique identifier
- billing_addr (not null) - to identify the billing address for billing information
- payment_method (not null) - to identify how customer pay the bill (cash or card)
- card_number - to identify card number associated with the billing information. It can be null if the customer decides to pay in cash
- customer_id (not null, referential integrity) - to identify the customer associated with the billing information and refers to other entities in the database (***customer***). ustomer_id must be unique.
- bstaff_id (not null, referential integrity) - to identify the billing staff who generates this billing information and refers to other entities in the database (***billing_staff***)

***reserve_for*(hotel_id, room_number, checkin_id)**

- hotel_id (not null, referential integrity) - to identify which hotel the room belongs to is associated with specific check-in record and refers to other entities in the database (***hotel***)
- room_number (not null, referential integrity) - to identify which room in the hotel is associated with specific check-in record and refers to other entities in the database (***room***).
- checkin_id (primary key, referential integrity) - unique identifier and refers to other entities in the database (***checkin***)

3) SQL Statements

Drop Table

```
DROP TABLE hotel CASCADE CONSTRAINTS;

DROP TABLE staff CASCADE CONSTRAINTS;

DROP TABLE manager CASCADE CONSTRAINTS;

DROP TABLE frontdesk_staff CASCADE CONSTRAINTS;

DROP TABLE billing_staff CASCADE CONSTRAINTS;

DROP TABLE catering_staff CASCADE CONSTRAINTS;

DROP TABLE service_staff CASCADE CONSTRAINTS;
```

```
DROP TABLE room CASCADE CONSTRAINTS;
DROP TABLE room_type CASCADE CONSTRAINTS;
DROP TABLE checkin CASCADE CONSTRAINTS;
DROP TABLE service_record CASCADE CONSTRAINTS;
DROP TABLE restaurant_record CASCADE CONSTRAINTS;
DROP TABLE phone_record CASCADE CONSTRAINTS;
DROP TABLE laundry_record CASCADE CONSTRAINTS;
DROP TABLE customer CASCADE CONSTRAINTS;
DROP TABLE billing CASCADE CONSTRAINTS;
DROP TABLE reserve_for CASCADE CONSTRAINTS;
```

Drop Sequence

```
DROP SEQUENCE hotel_seq;
DROP SEQUENCE staff_seq;
DROP SEQUENCE checkin_seq;
DROP SEQUENCE service_record_seq;
DROP SEQUENCE cutomer_seq;
DROP SEQUENCE billing_seq;
```

Create Tables

```
CREATE TABLE hotel (
      id INT PRIMARY KEY,
      name CHAR(32) NOT NULL,
      address VARCHAR(255) NOT NULL,
      phone VARCHAR(20) NOT NULL
);

CREATE TABLE customer(
      id INT PRIMARY KEY,
      name CHAR(32) NOT NULL,
      ssn VARCHAR(11) NOT NULL UNIQUE,
```

```
    gender CHAR(1) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(64) NOT NULL,
    address VARCHAR(255) NOT NULL,
    CHECK(gender IN ('f', 'm'))
);


CREATE TABLE staff (
    id INT PRIMARY KEY,
    ssn VARCHAR(11) NOT NULL UNIQUE,
    name CHAR(32) NOT NULL,
    age INT NOT NULL,
    gender CHAR(1) NOT NULL,
    title VARCHAR(32) NOT NULL,
    department VARCHAR(32) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    address VARCHAR(255) NOT NULL,
    hotel_id INT NOT NULL,
    CHECK(gender IN ('f', 'm')),
    CONSTRAINT staff_hotel_id_fk FOREIGN KEY(hotel_id) REFERENCES
hotel(id)
    ON DELETE CASCADE
);

CREATE TABLE manager (
    id INT PRIMARY KEY,
    hotel_id INT NOT NULL UNIQUE,
    CONSTRAINT manager_fk FOREIGN KEY(id) REFERENCES staff(id) ON
DELETE CASCADE,
    CONSTRAINT hotel_fk FOREIGN KEY(hotel_id) REFERENCES hotel(id)
```

```
);

CREATE TABLE frontdesk_staff (
     id INT PRIMARY KEY,
     CONSTRAINT frontdesk_staff_fk FOREIGN KEY(id) REFERENCES
staff(id) ON DELETE CASCADE
);

CREATE TABLE billing_staff (
     id INT PRIMARY KEY,
     CONSTRAINT billing_staff_fk FOREIGN KEY(id) REFERENCES
staff(id) ON DELETE CASCADE
);

CREATE TABLE catering_staff (
     id INT PRIMARY KEY,
     CONSTRAINT catering_staff_fk FOREIGN KEY(id) REFERENCES
staff(id) ON DELETE CASCADE
);

CREATE TABLE service_staff (
     id INT PRIMARY KEY,
     CONSTRAINT service_staff_fk FOREIGN KEY(id) REFERENCES
staff(id) ON DELETE CASCADE
);

CREATE TABLE room_type(
     category VARCHAR(20) NOT NULL,
     occupancy INT NOT NULL,
     nightly_rate INT NOT NULL,
     PRIMARY KEY (category, occupancy)
);
```

```
CREATE TABLE room (
      room_number INT,
      category VARCHAR(20) NOT NULL,
      occupancy INT NOT NULL,
      availability INT NOT NULL,
      hotel_id INT NOT NULL,
      PRIMARY KEY (hotel_id, room_number),
      CONSTRAINT room_hotel_id_fk FOREIGN KEY(hotel_id) REFERENCES
hotel(id),
      CONSTRAINT room_type_fk FOREIGN KEY(category, occupancy)
REFERENCES room_type(category, occupancy)
);


CREATE TABLE checkin(
      id INT PRIMARY KEY,
      current_occupancy INT NOT NULL,
      start_date DATE NOT NULL,
      end_date DATE NOT NULL,
      start_time CHAR(20) NOT NULL,
      end_time CHAR(20) NOT NULL,
      cstaff_id INT,
      sstaff_id INT,
      fdstaff_id INT NOT NULL,
      customer_id INT NOT NULL,
      CONSTRAINT checkin_cstaff_id_fk FOREIGN KEY(cstaff_id)
REFERENCES catering_staff(id) ON DELETE CASCADE,
      CONSTRAINT checkin_sstaff_id_fk FOREIGN KEY(sstaff_id)
REFERENCES service_staff(id) ON DELETE CASCADE,
      CONSTRAINT checkin_fdstaff_id_fk FOREIGN KEY(fdstaff_id)
REFERENCES frontdesk_staff(id) ON DELETE CASCADE,
      CONSTRAINT checkin_customer_id_fk FOREIGN KEY(customer_id)
REFERENCES customer(id) ON DELETE CASCADE
```

```
);

CREATE TABLE service_record(
      id INT PRIMARY KEY,
      type VARCHAR(16) NOT NULL,
      amount INT NOT NULL,
      checkin_id INT NOT NULL,
      CONSTRAINT checkin_id_fk FOREIGN KEY(checkin_id) REFERENCES
checkin(id) ON DELETE CASCADE
);

CREATE TABLE restaurant_record(
      id INT PRIMARY KEY,
      cstaff_id INT NOT NULL,
      CONSTRAINT restaurant_cstaff_id_fk FOREIGN KEY(cstaff_id)
REFERENCES catering_staff(id) ON DELETE CASCADE,
      CONSTRAINT restaurant_record_id_fk FOREIGN KEY(id) REFERENCES
service_record(id) ON DELETE CASCADE
);

CREATE TABLE phone_record(
      id INT PRIMARY KEY,
      sstaff_id INT NOT NULL,
      CONSTRAINT phone_sstaff_id_fk FOREIGN KEY(sstaff_id) REFERENCES
service_staff(id) ON DELETE CASCADE,
      CONSTRAINT phone_record_id_fk FOREIGN KEY(id) REFERENCES
service_record(id) ON DELETE CASCADE
);

CREATE TABLE laundry_record(
      id INT PRIMARY KEY,
      sstaff_id INT NOT NULL,
```

```
     CONSTRAINT laundry_sstaff_id_fk FOREIGN KEY(sstaff_id)
REFERENCES service_staff(id) ON DELETE CASCADE,

     CONSTRAINT laundry_record_id_fk FOREIGN KEY(id) REFERENCES
service_record(id) ON DELETE CASCADE

);


CREATE TABLE billing(

     id INT PRIMARY KEY,

     billing_addr VARCHAR(255) NOT NULL,

     payment_method CHAR(4) NOT NULL,

     card_number VARCHAR(16),

     customer_id INT NOT NULL UNIQUE,

     bstaff_id INT NOT NULL,

     CONSTRAINT billing_customer_id_fk FOREIGN KEY(customer_id)
REFERENCES customer(id) ON DELETE CASCADE,

     CONSTRAINT bstaff_id_fk FOREIGN KEY(bstaff_id) REFERENCES
billing_staff(id) ON DELETE CASCADE

);


CREATE TABLE reserve_for(

     hotel_id INT,

     room_number INT,

     checkin_id INT PRIMARY KEY,

     CONSTRAINT reserve_room_number_fk FOREIGN KEY(room_number,
hotel_id) REFERENCES room(room_number, hotel_id) ON DELETE CASCADE,

     CONSTRAINT reserve_checkin_id_fk FOREIGN KEY(checkin_id)
REFERENCES checkin(id) ON DELETE CASCADE

);
```

Sequence

```
CREATE SEQUENCE hotel_seq MINVALUE 0 START WITH 0;

CREATE SEQUENCE  staff_seq MINVALUE 0 START WITH 0;
```

```
CREATE SEQUENCE checkin_seq MINVALUE 0 START WITH 0;

CREATE SEQUENCE service_record_seq MINVALUE 0 START WITH 0;

CREATE SEQUENCE cutomer_seq MINVALUE 0 START WITH 0;

CREATE SEQUENCE billing_seq MINVALUE 0 START WITH 0;
```

Inserts

```
INSERT INTO hotel(id, name, address, phone) VALUES
(hotel_seq.nextval, '4 Seasons', '1010 Laney Dr., Raleigh, NC 27612',
'909-909-9090');

INSERT INTO hotel(id, name, address, phone) VALUES
(hotel_seq.nextval, 'Hyatt', '2020 Drivey Ln., San Francisco, CA
94102', '808-808-8080');

INSERT INTO hotel(id, name, address, phone) VALUES
(hotel_seq.nextval, 'Marriot', '3030 Boulevard Circle, Boston, MA
02108', '707-707-7070');

INSERT INTO hotel(id, name, address, phone) VALUES
(hotel_seq.nextval, 'Microhotel', '4040 Circle Blvd., Seattle, WA
98101', '606-606-6060');


INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
('Economy', 2, 80);

INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
('Economy', 1, 60);

INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
('Presidential Suite', 8, 300);

INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
('Deluxe', 4, 120);

INSERT INTO room_type(category, occupancy, nightly_rate) VALUES
('Executive Suite', 4, 200);


INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Gurgen Schneider', '111111111', 'm',
'121-121-1212', 'gurgen.schneider@gmail.com', 'USA 1');

INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Alena Ng', '222222222', 'f',
'131-131-1313', 'alena.ng@gmail.com', 'USA 2');
```

```
INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Matt Mutton', '333333333', 'm',
'141-141-1414', 'matt.mutton@gmail.com', 'USA 3');

INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Tobias Cantu', '444444444', 'm',
'151-151-1515', 'tobias.cantu@gmail.com', 'USA 4');

INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Sarah Rademakers', '555555555', 'f',
'161-161-1616', 'sara.rademakers@gmail.com', 'USA 5');

INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Hilda Milligan', '666666666', 'm',
'171-171-1717', 'hilda.milligan@gmail.com', 'USA 6');


INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (301, 'Economy', 2, 0, 0);

INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (401, 'Deluxe', 4, 0, 1);

INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (804, 'Presidential Suite', 8, 1, 2);

INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (601, 'Executive Suite', 4, 1, 3);


INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '123456789',
'Jane Doe', 45, 'f', 'sous chef', 'catering', '555-555-5555', '3414
Lane Rd., Raleigh, NC 27606', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '987654321',
'Joe Smith', 32, 'm', 'front desk staff', 'front desk',
'444-444-4444', '8765 Lane Rd., Raleigh, NC 27606', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '121234345',
'Mary Lamb', 89, 'f', 'billing staff', 'billing', '333-333-3333',
'5678 Road Ln., San Francisco, CA 94102', 1);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '212143435',
'Muffin Man', 76, 'm', 'service staff', 'service', '222-222-2222',
'9090 Road Ln., San Francisco, CA 94102', 1);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '676789890',
```

```
'Shoe Lady', 53, 'f', 'manager', 'management', '111-111-1111', '7384
Creek Dr., Boston, MA 02108', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '767698980',
'Peter Piper', 19, 'm', 'manager', 'management', '999-999-9999',
'4567 Creek Dr., Boston, MA 02108', 1);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '192837465',
'Black Sheep', 61, 'f', 'manager', 'management', '888-834-8888',
'9190 Circle Blvd., Seattle, WA 98101', 2);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '918273645',
'Bob Sheep', 62, 'm', 'manager', 'management', '888-888-8888', '9190
Circle Blvd., Seattle, WA 98101', 3);


INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '918542345',
'Peter Sheep', 34, 'm', 'fish chef', 'catering', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '912363645',
'Shu Sheep', 23, 'f', 'chicken chef', 'catering', '888-123-8888',
'9190 Circle Blvd., Seattle, NA 98101', 3);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '910000645',
'Taylor Win', 41, 'm', 'pork chef', 'catering', '888-888-8888', '9190
Circle Blvd., Seattle, WA 98101', 1);


INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '435673645',
'Taylor Winner', 31, 'm', 'billing staff', 'billing', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '939473645',
'Taylor Winson', 21, 'm', 'billing staff', 'billing', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 2);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '444473645',
'Taylor Winsy', 26, 'f', 'billing staff', 'billing', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 3);
```

```
INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '049583645',
'Taylor Winton', 24, 'f', 'service', 'service staff', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 2);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '986753645',
'Taylor Sam', 33, 'm', 'service', 'service staff', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 1);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '993343645',
'Taylor Tom', 30, 'm', 'service', 'service staff', '888-888-8888',
'9190 Circle Blvd., Seattle, WA 98101', 3);


INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '172383645',
'Taylor Yeh', 22, 'm', 'front desk staff', 'front desk',
'888-888-8888', '9190 Circle Blvd., Seattle, WA 98101', 0);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '203943645',
'Taylor Zhang', 35, 'f', 'front desk staff', 'front desk',
'888-888-8888', '9190 Circle Blvd., Seattle, WA 98101', 1);

INSERT INTO staff(id, ssn, name, age, gender, title, department,
phone, address, hotel_id) VALUES (staff_seq.nextval, '758393645',
'Taylor Yi', 29, 'm', 'front desk staff', 'front desk',
'888-888-8888', '9190 Circle Blvd., Seattle, WA 98101', 3);



INSERT INTO catering_staff(id) VALUES (0);

INSERT INTO catering_staff(id) VALUES (8);

INSERT INTO catering_staff(id) VALUES (9);

INSERT INTO catering_staff(id) VALUES (10);

INSERT INTO frontdesk_staff(id) VALUES (1);

INSERT INTO frontdesk_staff(id) VALUES (17);

INSERT INTO frontdesk_staff(id) VALUES (18);

INSERT INTO frontdesk_staff(id) VALUES (19);

INSERT INTO billing_staff(id) VALUES (2);
```

```
INSERT INTO billing_staff(id) VALUES (11);

INSERT INTO billing_staff(id) VALUES (12);

INSERT INTO billing_staff(id) VALUES (13);

INSERT INTO service_staff(id) VALUES (3);

INSERT INTO service_staff(id) VALUES (14);

INSERT INTO service_staff(id) VALUES (15);

INSERT INTO service_staff(id) VALUES (16);

INSERT INTO manager(id, hotel_id) VALUES (4, 0);

INSERT INTO manager(id, hotel_id) VALUES (5, 1);

INSERT INTO manager(id, hotel_id) VALUES (6, 2);

INSERT INTO manager(id, hotel_id) VALUES (7, 3);


INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
VALUES (checkin_seq.nextval, 2, '05-OCT-2016', '07-OCT-2016',
'12:12:12', '12:12:12', null, null, 1, 0);

INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
VALUES (checkin_seq.nextval, 8, '20-NOV-2016', '25-NOV-2016',
'12:12:13', '12:12:14', 0, 3, 1, 1);

INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
VALUES (checkin_seq.nextval, 4, '24-NOV-2016', '26-NOV-2016',
'12:12:13', '12:12:14', null, null, 1, 2);

INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
VALUES (checkin_seq.nextval, 4, '25-NOV-2016', '28-NOV-2016',
'12:12:13', '12:12:14', null, null, 1, 3);


INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 1', 'CARD',
'1234123412341234', 0, 2);

INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 2', 'CASH',
NULL, 1, 2);
```

```
INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 3', 'CARD',
'2345234523452345', 2, 2);

INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 4', 'CASH',
NULL, 3, 2);

INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 5', 'CASH',
NULL, 4, 2);

INSERT INTO billing(id, billing_addr, payment_method, card_number,
customer_id, bstaff_id) VALUES (billing_seq.nextval, 'USA 6', 'CASH',
NULL, 5, 2);


INSERT INTO reserve_for(hotel_id, room_number, checkin_id) VALUES (0,
301, 0);

INSERT INTO reserve_for(hotel_id, room_number, checkin_id) VALUES (1,
401, 2);

INSERT INTO reserve_for(hotel_id, room_number, checkin_id) VALUES (2,
804, 1);

INSERT INTO reserve_for(hotel_id, room_number, checkin_id) VALUES (3,
601, 3);


INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'phone', 32, 0);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'restaurant', 12, 1);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'laundry', 20, 0);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'phone', 14, 1);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'restaurant', 24, 3);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'restaurant', 16, 2);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'restaurant', 90, 1);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'phone', 23, 2);
```

```
INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'phone', 5, 3);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'laundry', 5, 3);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'laundry', 9, 1);

INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'laundry', 15, 2);


INSERT INTO restaurant_record(id, cstaff_id) VALUES (1, 0);

INSERT INTO restaurant_record(id, cstaff_id) VALUES (4, 0);

INSERT INTO restaurant_record(id, cstaff_id) VALUES (5, 0);

INSERT INTO restaurant_record(id, cstaff_id) VALUES (6, 0);


INSERT INTO phone_record(id, sstaff_id) VALUES (0, 3);

INSERT INTO phone_record(id, sstaff_id) VALUES (3, 3);

INSERT INTO phone_record(id, sstaff_id) VALUES (7, 3);

INSERT INTO phone_record(id, sstaff_id) VALUES (8, 3);


INSERT INTO laundry_record(id, sstaff_id) VALUES (2, 3);

INSERT INTO laundry_record(id, sstaff_id) VALUES (9, 3);

INSERT INTO laundry_record(id, sstaff_id) VALUES (10, 3);

INSERT INTO laundry_record(id, sstaff_id) VALUES (11, 3);
```

Selects

Note: All result sets from select statements have been formatted for readability.

```
SELECT * from hotel;

ID    NAME        ADDRESS                                        PHONE

-------------------------------------------------------------------------------
-------------------------------------
```

0	4 Seasons	1010 Laney Dr., Raleigh, NC 27612	909-909-9090
1	Hyatt	2020 Drivey Ln., San Francisco, CA 94102	808-808-8080
2	Marriot	3030 Boulevard Circle, Boston, MA 02108	707-707-7070
3	Microhotel	4040 Circle Blvd., Seattle, WA 98101	606-606-6060

SQL> SELECT * FROM STAFF;

ID	SSN	NAME	AGE	G	TITLE	DEPARTMENT	PHONE	ADDRESS	HOTEL_ID
0	123456789	Jane Doe	45	f	sous chef	catering	555-555-5555	3414 Lane Rd., Raleigh, NC 27606	0
1	987654321	Joe Smith	32	m	front desk staff	front desk	444-444-4444	8765 Lane Rd., Raleigh, NC 27606	0
2	121234345	Mary Lamb	89	f	billing staff	billing	333-333-3333	5678 Road Ln., San Francisco, CA 94102	1
3	212143435	Muffin Man	76	m	service st	service	222-222-2222	9090 Road Ln., San Francisco, CA 94102	1
4	676789890	Shoe Lady	53	f	manager	management	111-111-1111	7384 Creek Dr., Boston, MA 02108	0
5	767698980	Peter Piper	19	m	manager	management	999-999-9999	4567 Creek Dr., Boston, MA 02108	1
6	192837465	Black Sheep	61	f	manager	management	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	2
7	918273645	Bob Sheep	62	m	manager	management	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	3
8	918542345	Peter Sheep	34	m	fish chef	catering	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	0
9	912363645	Shu Sheep	23	f	chicken chef	catering	888-123-8888	9190 Circle Blvd., Seattle, NA 98101	3
10	910000645	Taylor Win	41	m	pork chef	catering	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	1
11	435673645	Taylor Winner	31	m	billing staff	billing	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	0
12	939473645	Taylor Winson	21	m	billing staff	billing	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	2
13	444473645	Taylor Winsy	26	f	billing staff	billing	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	3
14	049583645	Taylor Winton	34	f	service	service staff	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	2
15	986753645	Taylor Sam	33	m	service	service staff	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	1
16	993343645	Taylor Tom	30	m	service	service staff	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	3
17	172383645	Taylor Yeh	22	m	front desk staff	front desk	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	0
18	203943645	Taylor Zhang	35	f	front desk staff	front desk	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	1
19	758393645	Taylor Yi	29	m	front desk staff	front desk	888-888-8888	9190 Circle Blvd., Seattle, WA 98101	3

SELECT * from room_type;

CATEGORY	OCCUPANCY	NIGHTLY_RATE
---	---	---

Economy	2	80
Economy	1	60
Presidential Suite	8	300
Deluxe	4	120
Executive Suite	4	200

SELECT * from customer;

ID	NAME	SSN	G	PHONE	EMAIL	ADDRESS
0	Gurgen Schneider	111111111	m	121-121-1212	gurgen.schneider@gmail.com	USA 1
1	Alena Ng	222222222	f	131-131-1313	alena.ng@gmail.com	USA 2
2	Matt Mutton	333333333	m	141-141-1414	matt.mutton@gmail.com	USA 3
3	Tobias Cantu	444444444	m	151-151-1515	tobias.cantu@gmail.com	USA 4
4	Sarah Rademakers	555555555	f	161-161-1616	sara.rademakers@gmail.com	USA 5
5	Hilda Milligan	666666666	m	171-171-1717	hilda.milligan@gmail.com	USA 6

SQL> select * from room;

ROOM_NUMBER	CATEGORY	OCCUPANCY	AVAILABILITY	HOTEL_ID
301	Economy	2	0	0
401	Deluxe	4	0	1
804	Presidential Suite	8	1	2
601	Executive Suite	4	1	3

SELECT * FROM catering_staff;

ID

```
----------
         0
         8
         9
        10

SELECT * FROM frontdesk_staff;

        ID
----------
         1
        17
        18
        19

SELECT * FROM billing_staff;

        ID
----------
         2
        11
        12
        13

SELECT * FROM service_staff;

        ID
----------
```

3

14

15

16

SELECT * from manager;

ID	HOTEL_ID
4	0
5	1
6	2
7	3

*SQL> select * from reserve_for;*

HOTEL_ID	ROOM_NUMBER	CHECKIN_ID
0	301	0
1	401	2
2	804	1
3	601	3

SELECT * FROM checkin;

ID	CURRENT_OCCUPANCY	START_DAT	END_DATE	START_TIME	END_TIME	CSTAFF_ID	SSTAFF_ID	FDSTAFF_ID	CUSTOMER_ID
0	2	05-OCT-16	07-OCT-16	12:12:12	12:12:12			1	0
1	8	20-NOV-16	25-NOV-16	12:12:13	12:12:14	0	3	1	1
2	4	24-NOV-16	26-NOV-16	12:12:13	12:12:14			1	2
3	4	25-NOV-16	28-NOV-16	12:12:13	12:12:14			1	3

```
SQL> SELECT * FROM SERVICE_RECORD
ID          TYPE         AMOUNT CHECKIN_ID
---------- ----------------    ----------    ----------
0         phone     32         0
1         restaurant       12         1
2         laundry          20         0
3         phone     14         1
4         restaurant       24         3
5         restaurant       16         2
6         restaurant       90         1
7         phone     23         2
8         phone     5          3
9         laundry          5          3
10        laundry          9          1
11        laundry          15         2
```

SQL> select * from restaurant_record;

ID	CSTAFF_ID
1	0
4	0
5	0
6	0

SQL> select * from phone_record;

ID	SSTAFF_ID
0	3
3	3
7	3

```
    8       3
SQL> select * from laundry_record;

    ID  SSTAFF_ID
---------- ----------
     2       3
     9       3
    10       3
    11       3
```

```
SQL> SELECT * FROM billing;

    ID BILLING_ADDR    PAYM          CARD_NUMBER       CUSTOMER_ID  BSTAFF_ID
---------- ---------------------------------------------------- ---- ---------------- ----------- ----------
     0       USA 1          CARD          1234123412341234        0                    2
     1       USA 2          CASH                                  1                    2
     2       USA 3          CARD          2345234523452345        2                    2
     3       USA 4          CASH                                  3                    2
     4       USA 5          CASH                                  4                    2
     5       USA 6          CASH                                  5                    2
```

4) Interactive SQL Queries

Note: All result sets from select statements have been formatted for readability.

4.1 Queries for tasks and operations

The following commands assume that no other actions have been taken on the database, and that the following commands are executed chronologically.

1.Information processing

Enter basic information about staff

```
SQL> INSERT INTO staff(id, ssn, name, age, gender, title,
department, phone, address, hotel_id) VALUES (staff_seq.nextval,
'913857668', 'Josh Doe', 19, 'm', 'Waiter', 'catering',
'565-565-5656', '4321 Road St., Raleigh, NC 27607', 1);


1 row created.
```

Update basic information about staff

```
SQL> UPDATE staff SET phone='919-191-9191', title = 'Head Waiter',
address = 'canada road, Canada' WHERE id=8;


1 row updated.
```

Delete basic information about staff

```
SQL> DELETE FROM staff WHERE id=8;


1 row deleted.
```

Enter basic information about room

```
INSERT INTO room(room_number, category, occupancy, availability,
hotel_id) VALUES (302, 'Economy', 2, 0, 0);


1 row created.
```

Update basic information about room

```
UPDATE room SET occupancy = 1 WHERE(room_number = 302 AND hotel_id =
0);


1 row updated.
```

Delete basic information about room

```
DELETE FROM room WHERE(room_number = 302 AND hotel_id = 0);
```

```
1 row deleted.
```

Enter basic information about customer

```
INSERT INTO customer(id, name, ssn, gender, phone, email, address)
VALUES (cutomer_seq.nextval, 'Ebenezer Scruge', '777777777', 'm',
'312-231-3113', 'scruginator@gmail.com', '7 real street, USA');


1 row created.
```

Update basic information about customer

```
UPDATE customer SET name = 'Ebenezer Scruge JR.', ssn = '777777787',
phone = '313-231-3113', email = 'scruginator2@gmail.com', address =
'7 real street, USA' WHERE id = 6;


1 row updated.
```

Delete basic information about customer

```
DELETE FROM customer WHERE id=6;


1 row deleted.
```

Check available room

```
SELECT * FROM room WHERE availability = 1;


ROOM_NUMBER    CATEGORY                   OCCUPANCY    AVAILABILITY  HOTEL_ID
-----------  --------------------  ----------  ------------
804          Presidential Suite         8            1         2
601          Executive Suite            4            1         3
```

Assign room to customers according to their request

```
SELECT * FROM room WHERE(availability = 1 AND category = 'Executive
Suite');

INSERT INTO checkin(id, current_occupancy, start_date, end_date,
start_time, end_time, cstaff_id, sstaff_id, fdstaff_id, customer_id)
```

```
VALUES (checkin_seq.nextval, 4, '12-OCT-2016', '18-OCT-2016',
'10:10:10', '10:10:10', null, null, 1, 1);

INSERT INTO reserve_for(hotel_id, room_number, checkin_id) VALUES (3,
601, 4);

UPDATE room SET availability = 0 WHERE(room_number = 601 AND hotel_id
= 3);

ROOM_NUMBER CATEGORY          OCCUPANCY AVAILABILITY   HOTEL_ID
----------- -------------------- ---------- ------------ ----------
      601 Executive Suite           4          1     3


1 row created.

1 row created.

1 row updated.
```

Releasing room

```
UPDATE room SET availability = 1 WHERE(room_number = 601 AND hotel_id
= 3);

DELETE FROM reserve_for WHERE(room_number = 601 AND hotel_id = 3 AND
checkin_id = 4);


1 row updated.

1 row deleted.
```

2.Maintaining Service Availed Records

Enter record for phone bill

```
SQL> INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'phone', 32, 0);


1 row created.

SQL> INSERT INTO phone_record(id, sstaff_id) VALUES (12, 3);


1 row created.
```

Update record for phone bill

```
SQL> UPDATE service_record SET amount = 300 WHERE(id = 12 AND type =
'phone');


1 row updated.
```

Enter record for laundry bill

```
SQL> INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'laundry', 10, 0);


1 row created.

SQL> INSERT INTO laundry_record(id, sstaff_id) VALUES (13, 3);


1 row created.
```

Update record for laundry bill

```
SQL> UPDATE service_record SET amount = 20 WHERE (id = 13 AND type =
'laundry');


1 row updated.
```

Enter record for restaurant bill

```
SQL> INSERT INTO service_record(id, type, amount, checkin_id) VALUES
(service_record_seq.nextval, 'restaurant', 108, 0);

1 row created.

SQL> INSERT INTO laundry_record(id, sstaff_id) VALUES (14, 3);


1 row created.
```

Update record for laundry bill

```
SQL> UPDATE service_record SET amount = 100 WHERE (id = 14 AND type =
'restaurant');


1 row updated.
```

3.Maintaining Billing `accounts`

Generate billing account for every customer

```
INSERT INTO customer(id, name, ssn, gender, phone, email, address) VALUES
(cutomer_seq.nextval, 'Bill billingC', '6666646', 'm', '171-171-1717',
'hilda.milligan@gmail.com', 'USA 6');

UPDATE room SET availability = 1 WHERE hotel_id = 0 AND room_number = 301;

INSERT INTO billing  (id, billing_addr, payment_method, card_number, customer_id,
bstaff_id) SELECT billing_seq.nextval, 'USA 1', 'CARD', '1234123412341234', 6, 2 FROM
dual WHERE EXISTS (SELECT room_number FROM room r WHERE r.category = 'Economy' AND
r.availability=1 AND hotel_id = 0);


1 row created.

1 row updated.

1 row created.
```

Update billing account for every customer

```
UPDATE billing  SET billing_addr = 'CANADA1', payment_method =
'CARD', card_number = '1112223333888999' WHERE customer_id = 0;


1 row updated.
```

4.Reports

Report occupancy by room type

```
SELECT category, SUM(current_occupancy) FROM checkin c, room r,
reserve_for f WHERE (c.id = f.checkin_id AND f.room_number =
r.room_number AND f.hotel_id = r.hotel_id AND c.start_date >=
'18-OCT-16') GROUP BY r.category;

CATEGORY              SUM(CURRENT_OCCUPANCY)
--------------------- ----------------------
Deluxe                                     4
Presidential Suite                         8
Executive Suite                                 4
```

Report occupancy by date range

```
SELECT SUM(current_occupancy) FROM checkin WHERE ('20-OCT-16' <=
start_date AND '26-NOV-16' >= end_date);

SUM(CURRENT_OCCUPANCY)

----------------------

                    12
```

Report occupancy by hotel

```
SELECT hotel_id, SUM(current_occupancy) FROM checkin c,
reserve_for r WHERE (c.id = r.checkin_id AND '20-OCT-16' <=
c.start_date AND '26-NOV-16' >= c.end_date) GROUP BY r.hotel_id;

  HOTEL_ID SUM(CURRENT_OCCUPANCY)
---------- ----------------------
         1                      4
         2                      8
```

Report occupants

```
SELECT c.* FROM customer c, checkin k WHERE(c.id = k.customer_id
AND '20-OCT-16' <= k.start_date AND '26-NOV-16' >= k.end_date);

ID NAME       SSN       G PHONE        EMAIL                 ADDRESS
---------- ------------- ----------  ---- ----------    --------------
1 Alena Ng    222222222 f 131-131-1313 alena.ng@gmail.com     USA 2

2 Matt Mutton 333333333 m 141-141-1414  matt.mutton@gmail.com  USA 3
```

Report percentage of rooms occupied

```
SELECT COUNT(r.room_number)/COUNT(m.room_number)*100 AS
percent_occupied FROM room r, room m, reserve_for f, checkin c
WHERE (m.hotel_id = 1 AND r.hotel_id = m.hotel_id AND
r.availability = 0 AND f.room_number = m.room_number AND
f.hotel_id = m.hotel_id AND f.checkin_id = c.id AND '20-OCT-16'
<= c.start_date AND '26-NOV-16' >= c.end_date);

PERCENT_OCCUPIED
----------------
             100
```

Return information on staff grouped by their role

```
SELECT * FROM staff ORDER BY title;

ID  SSN         NAME      AGE G  TITLE      DEPARTMENT  PHONE        ADDRESS                      HOTEL_ID
---------------------------------------------------------------------------------------------------------------------------------------------------------------
13 444473645  Taylor Winsy   26   f  billing staff    billing         888-888-8888  9190 Circle Blvd., Seattle, WA 98101       3
2 121234345   Mary Lamb      89   f  billing staff    billing         333-333-3333  5678 Road Ln., San Francisco, CA 94102  1
```

12 939473645 Taylor Winson 21 m billing staff billing 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 2

11 435673645 Taylor Winner 31 m billing staff billing 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 0

9 912363645 Shu Sheep 23 f chicken chef catering 888-123-8888 9190 Circle Blvd., Seattle, NA 98101 3

8 918542345 Peter Sheep 34 m fish chef catering 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 0

19 758393645 Taylor Yi 29 m front desk staff front desk 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 3

1 987654321 Joe Smith 32 m front desk staff front desk 444-444-4444 8765 Lane Rd., Raleigh, NC 27606 0

18 203943645 Taylor Zhang 35 f front desk staff front desk 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 1

17 172383645 Taylor Yeh 22 m front desk staff front desk 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 0

4 676789890 Shoe Lady 53 f manager management 111-111-1111 7384 Creek Dr., Boston, MA 02108 0

6 192837465 Black Sheep 61 f manager management 888-834-8888 9190 Circle Blvd., Seattle, WA 98101 2

5 767698980 Peter Piper 19 m manager management 999-999-9999 4567 Creek Dr., Boston, MA 02108 1

7 918273645 Bob Sheep 62 m manager management 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 3

10 910000645 Taylor Win 41 m pork chef catering 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 1

16 993343645 Taylor Tom 30 m service service staff 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 3

15 986753645 Taylor Sam 33 m service service staff 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 1

14 049583645 Taylor Winton 24 f service service staff 888-888-8888 9190 Circle Blvd., Seattle, WA 98101 2

3 212143435 Muffin Man 76 m service staff service 222-222-2222 9090 Road Ln., San Francisco, CA 94102 1

0 123456789 Jane Doe 45 f sous chef catering 555-555-5555 3414 Lane Rd., Raleigh, NC 27606 0

20 rows selected.

Return information on all the customers of a given catering staff

SELECT c.* FROM customer c, checkin k WHERE (c.id = k.customer_id AND k.cstaff_id = 0);

ID NAME SSN G PHONE EMAIL ADDRESS

---------- ---------- ---------- ---- ---------- ----------

1 Alena Ng 222222222 f 131-131-1313 alena.ng@gmail.com USA 2

Return information on all the customers of a given service staff

SELECT c.* FROM customer c, checkin k WHERE (c.id = k.customer_id AND k.sstaff_id = 3);

ID NAME SSN G PHONE EMAIL ADDRESS

---------- ---------- ---------- ---- ---------- ----------

1 Alena Ng 222222222 f 131-131-1313 alena.ng@gmail.com USA 2

4.2 Autotrace and Indexes for two tables

The first index was added to customer based on the "name" column.

SQL> SET AUTOTRACE ON;
SP2-0618: Cannot find the Session Identifier. Check PLUSTRACE role is enabled

```
SP2-0611: Error enabling STATISTICS report
SQL> SELECT * FROM customer WHERE name = 'Gurgen Schneider';

ID NAME                 SSN    G PHONE          EMAIL                          ADDRESS
------------------------------------------------------------------------------
 0 Gurgen Schneider          111111111          m 121-121-1212 gurgen.schneider@gmail.com USA 1


Execution Plan
----------------------------------------------------------
Plan hash value: 2844954298

-------------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |     1 |   130 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| CUSTOMER |     1 |   130 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("NAME"='Gurgen Schneider')

Note
-----
   - 'PLAN_TABLE' is old version
   - dynamic sampling used for this statement (level=2)

SQL> CREATE INDEX customer_name ON customer(name);

Index created.
SQL> SELECT * FROM customer WHERE name = 'Gurgen Schneider';

ID NAME                 SSN                  G    PHONE         EMAIL                          ADDRESS
------------------------------------------------------------------------------
0 Gurgen Schneider       111111111     m    121-121-1212 gurgen.schneider@gmail.com   USA 1


Execution Plan
----------------------------------------------------------
Plan hash value: 3906886354

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     1 |   130 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMER      |     1 |   130 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | CUSTOMER_NAME |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("NAME"='Gurgen Schneider')

Note
-----
   - 'PLAN_TABLE' is old version
   - dynamic sampling used for this statement (level=2)

SQL>
```

The second index was added to room based on the "room_number" column.

```
SQL> SET AUTOTRACE ON;

SQL> select * from room where room_number>301;

ROOM_NUMBER CATEGORY         OCCUPANCY    AVAILABILITY   HOTEL_ID
----------- ------------------- ---------- ------------ ----------
        401  Deluxe                      4                  0            1
        804  Presidential Suite          8                  1                    2
        601  Executive Suite             4                  1                    3


Execution Plan
----------------------------------------------------------
Plan hash value: 640342614

-------------------------------------------------------------------------------
| Id   | Operation                          | Name   | Rows  | Bytes  | Cost (%CPU) | Time     |
-------------------------------------------------------------------------------
|   0  | SELECT STATEMENT   |            |     3     |   192  |     3   (0)          | 00:00:01 |
|*  1 |  TABLE ACCESS FULL  | ROOM |     3     |   192  |     3   (0)          | 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("ROOM_NUMBER">301)

Note
-----
   - 'PLAN_TABLE' is old version
   - dynamic sampling used for this statement (level=2)

SQL> create index room_number_id on room(room_number);

Index created.

SQL> select * from room where room_number>301;

ROOM_NUMBER                CATEGORY         OCCUPANCY   AVAILABILITY   HOTEL_ID
----------- -------------------- ---------- ------------ ----------
        401                        Deluxe                 4                  0                        1
        601                        Executive Suite        4                  1                        3
        804                        Presidential Suite     8                  1                      2


Execution Plan
----------------------------------------------------------
Plan hash value: 48694694

-------------------------------------------------------------------------------------------------
| Id   | Operation                                    | Name                       | Rows   | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0  | SELECT STATEMENT                     |                                |     3     |   192  |     2   (0)        | 00:00:01 |
|   1  |  TABLE ACCESS BY INDEX ROWID  | ROOM                       |     3     |   192  |     2   (0)        | 00:00:01 |
|*  2  |   INDEX RANGE SCAN                   | ROOM_NUMBER_ID  |     1     |          |     1   (0)        | 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("ROOM_NUMBER">301)

Note
-----
```

```
  - 'PLAN_TABLE' is old version
  - dynamic sampling used for this statement (level=2)
SQL>
```

4.3 Query correctness proofs

(1)SELECT c.* FROM (customer c INNER JOIN checkin k ON c.id = k.customer_id) WHERE (k.cstaff_id = 0);

1. Relational algrebra:

   $\pi_{ID, NAME, SSN, GENDER, PHONE, EMAIL, ADDRESS}(\sigma_{k.cstaff\_id=0}(\rho_c(customer) \bowtie_{c.id = k.customer\_id} P_k(checkin)))$

2. Specification: return all information on all customers who has the catering staff with id 0;
3. Explanation: Suppose c is any tuple in the customer relation and k is any tuple in the checkin relation such that the value c.id is the same as the value k.customer_id and the value of k.cstaff_id is 0. Each combination of tuples (c, k) gives information about one customer together with the information on the checkin for that customer if the checkin cstaff_id is 0. For each such combination (c,k) the query returns all the customer information. These values are the customer id, name, ssn, gender, phone number, email, and address of the customer. This is what the query should return.

Erroneous solutions:

1. SELECT * FROM customer WHERE cstaff_id = 0; Forgot cstaff_id was not in customer and is not linked to customer without using a join.
2. SELECT c.* FROM customer c, checkin k WHERE (c.id = k.customer_id AND k.cstaff_id = 0); Works but does not use a join.

(2)SELECT hotel_id, SUM(current_occupancy) FROM (checkin c INNER JOIN reserve_for r ON c.id = r.checkin_id) WHERE ('20-OCT-16' <= c.start_date AND '26-NOV-16' >= c.end_date) GROUP BY r.hotel_id;

1. Relational algrebra:
   $\Upsilon_{hotel\_id,SUM(current\_occupancy)}(\sigma_{c.start\_date>='20\text{-}OCT\text{-}16' \text{ AND } c.end\_date<='26\text{-}NOV\text{-}16'}(\rho_c(checkin) \bowtie_{c.id = r.checkin\_id} \rho_r(reserve\_for)))$
2. Specification: return the hotel id and total number of occupants for each hotel inside a particular date range;
3. Explanation: suppose c is any tuple in the checkin relation and r is any tuple in the reserve_for relation such that the value of c.id is the same as the value of

r.checkin_id and the c start and end dates falls in the range of October 20, 2016 and November 26, 2016. Each combination of tuples (c, r) gives information about the data for one checkin together with the information about the reservation for that checkin if the checkin falls in the correct date range. For each such combination (c, r) the query adds the current occupancy to the total occupancy for that hotel and then returns each hotel id with its associated occupancy. This is what the query should return.

Erroneous solutions:

1. '20-OCT-16' >= c.start_date AND '26-NOV-16' <= c.end_date got the wrong date range.
2. SELECT hotel_id, SUM(current_occupancy) FROM checkin c, reserve_for r WHERE (c.id = r.checkin_id AND '20-OCT-16' <= c.start_date AND '26-NOV-16' >= c.end_date) GROUP BY r.hotel_id; Works but does not use a join.