Hotels.com

(CS6360.002 F19) Database Design Final Project

By

Ali Shariq (sxa190016)

Ashika Hande (axh180061)

Dnyanesh Tarte (dnt190000)

Eric Jonsson School of Engineering and Computer Science

The University of Texas at Dallas

Table of Contents

Chapter 1

Requirements

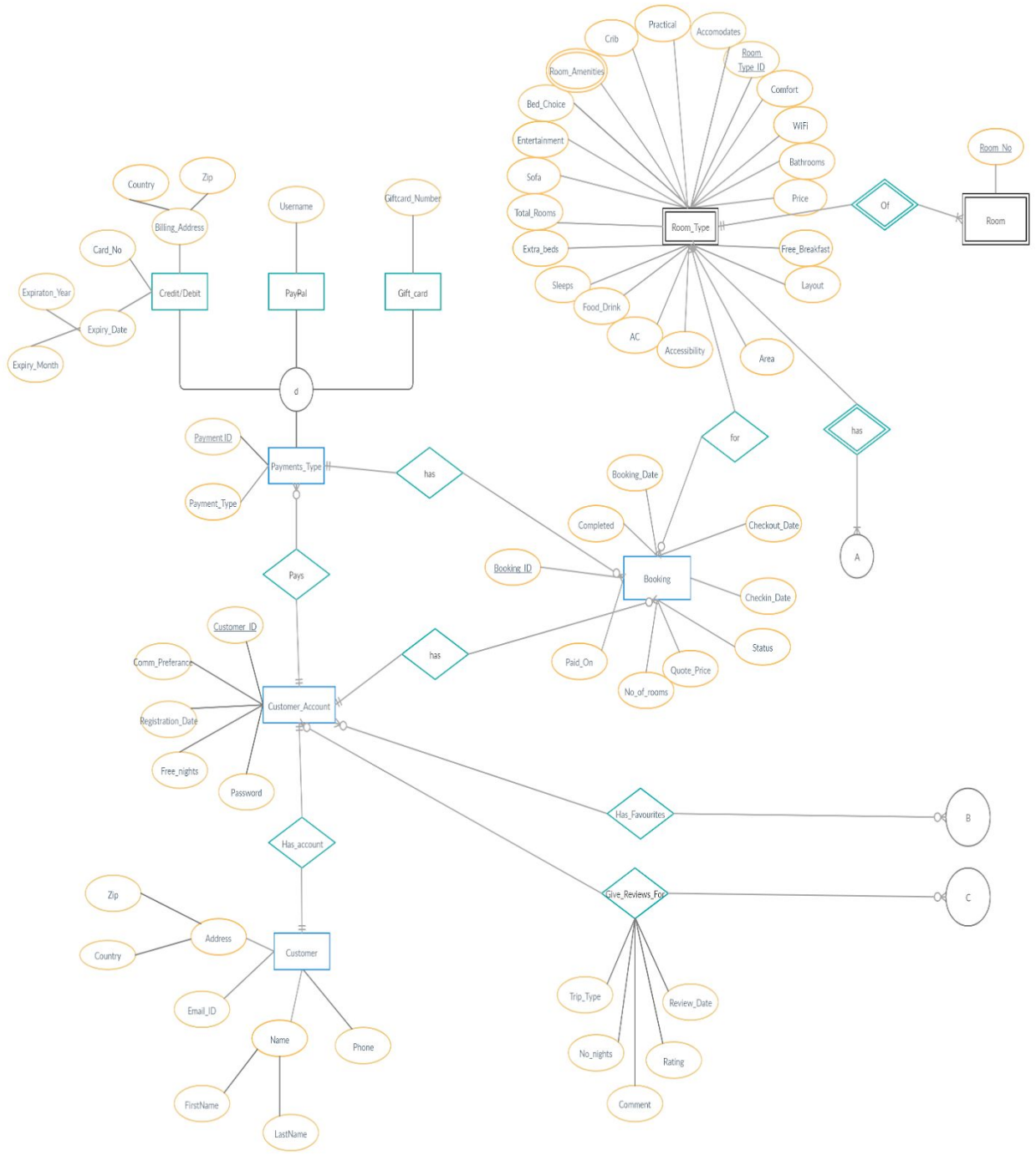
Hotels.com, established in Dallas, Texas in 1991, is a website for booking hotels online all around the world. The company connects customers with hotels in different cities and allows them to choose from a range of hotels. Each hotel has different types of rooms, amenities and places to visit around. Since it is an international company, the website supports multiple languages and currencies along with international payment methods and taxes accordingly. The company also has a robust loyalty program wherein every customer earns 'free nights' upon successful check out from previous hotels.

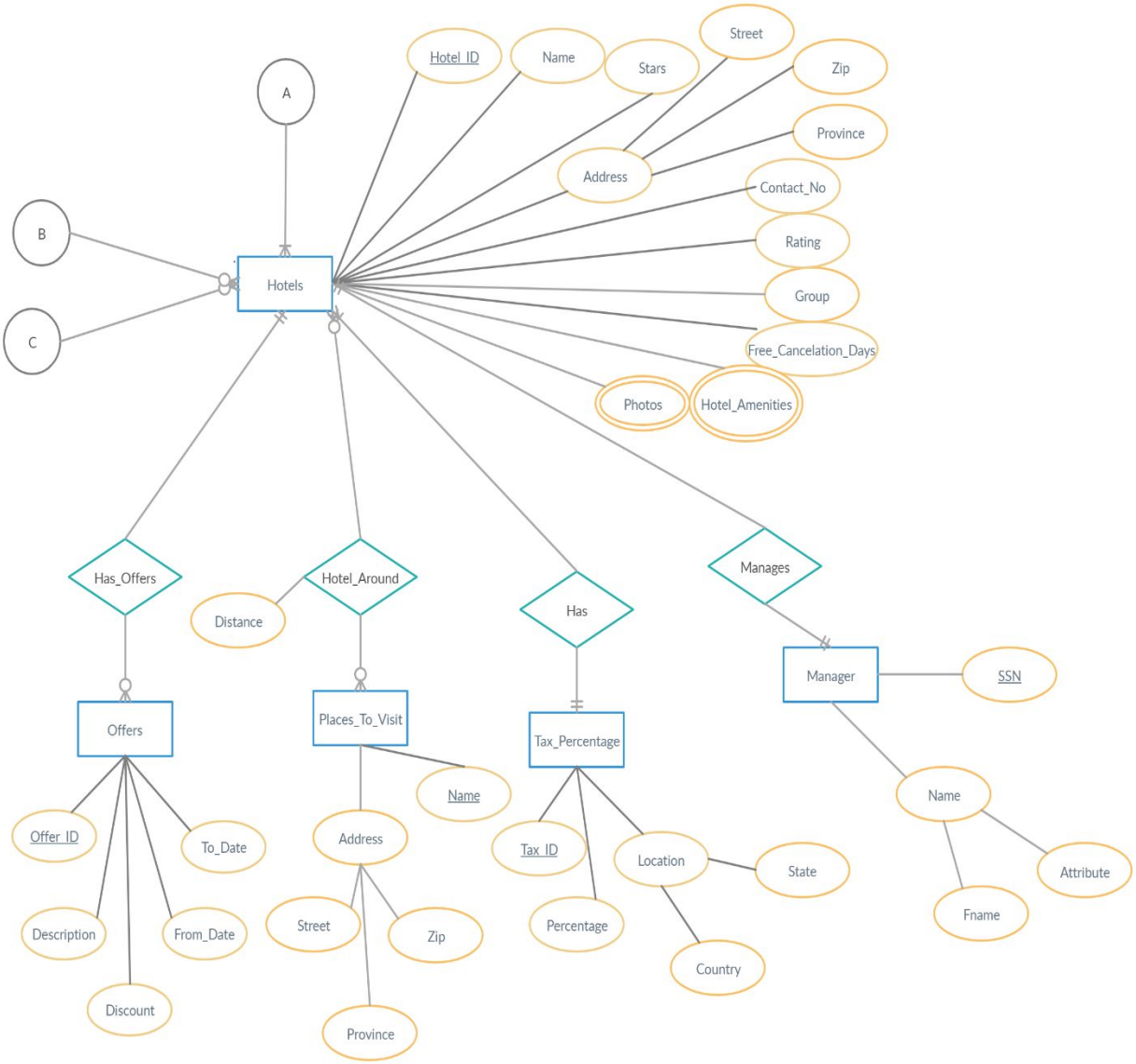
Main entities are:

- Customer: Any individual or a group of individuals who wishes to book a hotel room for a particular date range. A customer can have favorite hotels.
- Hotel: Any individual hotel or a group of hotels in a particular location. Every hotel has a fixed number of rooms of a particular room type. Each room is associated with some amenities particular to that room like A/C, TV etc. There are also some shared amenities like swimming pool, parking, bar etc which are shared by all the customers.
- Payment Type: Since it is an international website, it supports multiple payment methods, credit/debit card, PayPal or gift cards.
- Tax Percentage: The tax depends on the location of the hotel.
- Booking: Stores the booking details by a customer for a hotel. It is connected to the payment type entity to identify the payment method, customer entity to identify the customer and the room type entity to identify the type of room booked in that particular hotel. The customer gets to know the room type but not the room number.
- Amenities: There are 2 types of amenities, hotel amenities like swimming pool, parking, bar etc which are shared by all the customers and room amenities available for a particular room in a hotel like A/C, TV etc.
- Room Type: A hotel can have multiple rooms of multiple kinds.
- Offers: The website gives various offers on hotel bookings which have a start date and an expiry date. An offer is applicable to only one hotel but every hotel can have multiple offers.
- Reviews: A customer can write reviews about one of more hotels.

Chapter 2

ER Diagram




A
B
C
Hotels
Hotel_ID
Name
Stars
Street
Zip
Province
Address
Contact_No
Rating
Group
Free_Cancelation_Days
Photos
Hotel_Amenities
Has_Offers
Hotel_Around
Distance
Has
Manages
Manager
SSN
Offers
Places_To_Visit
Tax_Percentage
Name
Offer_ID
To_Date
Description
From_Date
Discount
Address
Street
Zip
Province
Tax_ID
Percentage
Location
State
Country
Name
Attribute
Fname

Summary of the relationships in the ER:

1. 1:1 Relationships
    a. Customer and Customer_Account
    b. Hotels and Manager

2. 1:N Relationships
    a. Hotels and Offers
    b. Hotels and Tax_percentage
    c. Customer_Account and Payments_Type
    d. Room_Type and Room
    e. Room_Type and Hotels
    f. Room_Type and Booking
    g. Payments_Type and Booking
    h. Customer_Account and Booking

3. M:N Relationships
    a. Hotels and Places_to_visit
    b. Customer_Account and Hotels (Has_Favourites)
    c. Customer_Account and Hotels (Give_reviews_for)

Chapter 3

Relational Schema

Chapter 4

Normalization

Violations for 1 NF:

Hotels (Hotel_id□ Photos, Hotel_Amenities)

Room_Type (Room_Type_id□Room_Amenities)

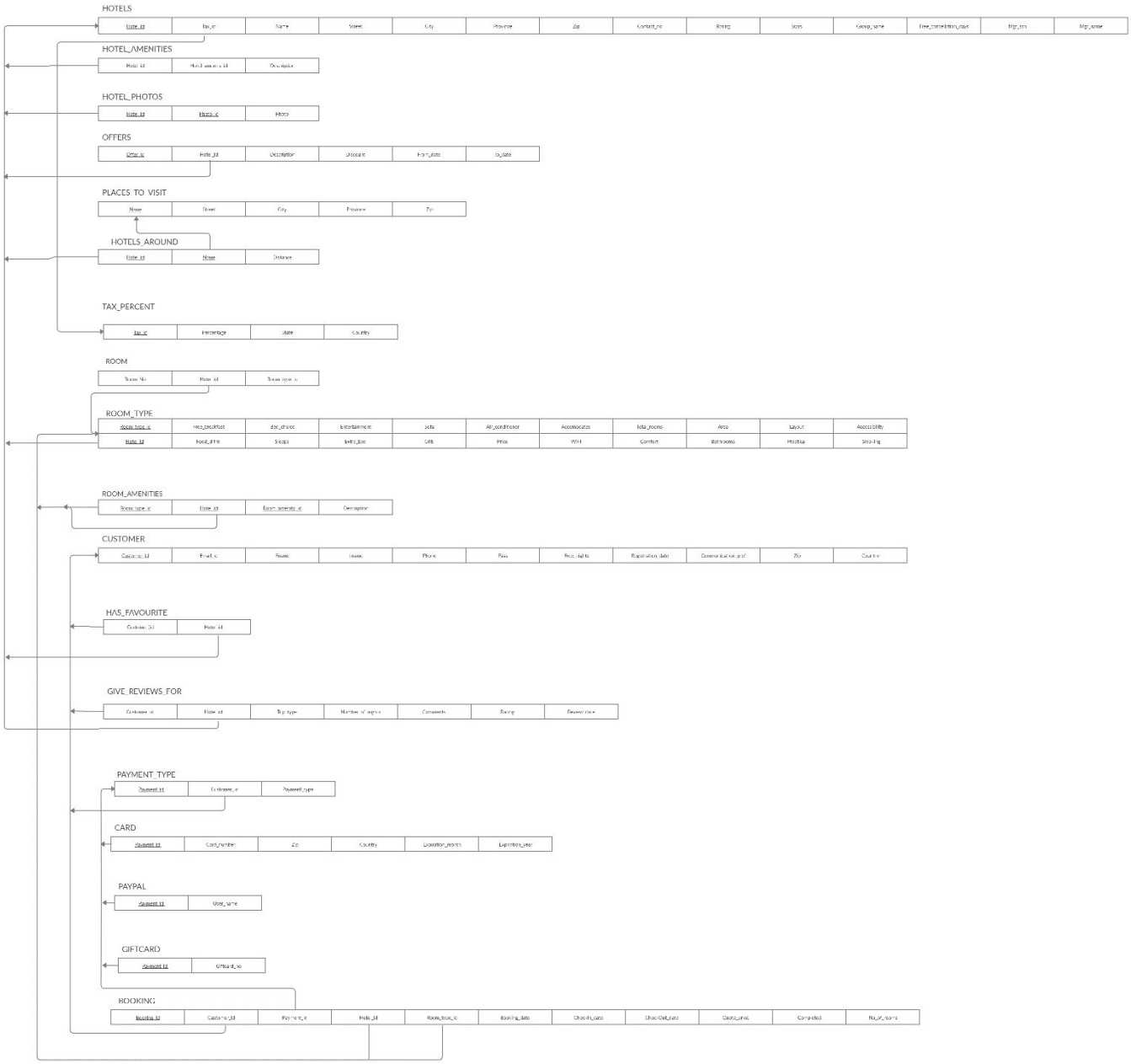
The above attributes have multiple values.

Hence, a separate table for each has been created.

There are no further violations in the relational schema. The following relational schema is in 3NF.

Chapter 5

Relational Schema after Normalization



Chapter 6

SQL Commands to create tables

```
CREATE TABLE Customer(
Customer_id varchar2(10),
Email_id varchar2(20) NOT NULL,
Fname varchar(10) NOT NULL,
Lname varchar2(10) NOT NULL,
Phone int NOT NULL,
Pass varchar2(8) NOT NULL,
Free_Nights int DEFAULT 0,
Registration_Date date NOT NULL,
Communication_Pref varchar2(20),
Zip int NOT NULL, CHECK(zip BETWEEN 10000 AND 99999),
Country varchar2(20) NOT NULL,
PRIMARY KEY(Customer_id)
);
```

```
CREATE TABLE Payment_type(
Payment_id varchar2(10) NOT NULL,
Customer_id varchar2(10) ,
Payment_type varchar2(10) NOT NULL,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Customer_id ) REFERENCES Customer(Customer_id ) ON DELETE
CASCADE
);
```

```
CREATE TABLE Card(
Payment_id varchar2(10) NOT NULL,
Card_number varchar2(16) UNIQUE,
Zip int NOT NULL,
Country varchar2(10) NOT NULL,
Expiration_month int NOT NULL,
Expiration_year int NOT NULL,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Payment_id) REFERENCES Payment_type(Payment_id) ON DELETE
CASCADE
);
```

```
CREATE TABLE Paypal(
Payment_id varchar2(10),
User_name varchar2(20) UNIQUE,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Payment_id) REFERENCES Payment_type(Payment_id) ON DELETE
CASCADE
);

CREATE TABLE Giftcard(
Payment_id varchar2(10) ,
Giftcard_no int UNIQUE,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Payment_id) REFERENCES Payment_type(Payment_id) ON DELETE
CASCADE
);

CREATE TABLE Hotels(
Hotel_id varchar2(10),
Tax_id varchar2(10) NOT NULL,
Name varchar2(10) NOT NULL,
Street varchar2(10) NOT NULL,
City varchar2(10) NOT NULL,
Province varchar2(10) NOT NULL,
Zip varchar2(10) NOT NULL,
Contact int NOT NULL,
Rating int DEFAULT 0,
Stars int NOT NULL,
Group_name varchar2(10) ,
Free_cancellation_days int,
Mgr_ssn varchar2(10) UNIQUE,
Mgr_name varchar2(50),
PRIMARY KEY (Hotel_id),
FOREIGN KEY (Tax_id ) REFERENCES Tax_percentage(Tax_id) ON DELETE SET NULL);

CREATE TABLE Hotel_Amenities(
Hotel_id varchar2(10),
Hotel_amenity_id varchar2(10),
Description varchar2(30),
PRIMARY KEY (Hotel_id, Hotel_amenity_id ),
FOREIGN KEY (Hotel_id) REFERENCES Hotels (Hotel_id) ON DELETE CASCADE
);
```

```
CREATE TABLE Hotel_Photos(
Hotel_id varchar2(10),
Photo_id varchar2(10),
Photo blob,
PRIMARY KEY (Hotel_id, Photo_id),
FOREIGN KEY (Hotel_id) REFERENCES Hotels(Hotel_id) ON DELETE CASCADE
);

CREATE TABLE Has_favourite(
Customer_id varchar2(10),
Hotel_id varchar2(10) ,
PRIMARY KEY (Customer_id  , Hotel_id ),
FOREIGN KEY (Customer_id ) REFERENCES Customer(Customer_id ) ON DELETE
CASCADE ,
FOREIGN KEY (Hotel_id ) REFERENCES Hotels(Hotel_id ) ON DELETE CASCADE
);


CREATE TABLE Room_Type(
Room_type_id varchar2(10),
Hotel_id varchar2(10) NOT NULL,
Accommodates int NOT NULL,
Bed_choice char NOT NULL,
Total_rooms int NOT NULL,
Area int NOT NULL,
Layout varchar2(10) NOT NULL,
Accessibility varchar2(20) NOT NULL,
Free_breakfast char NOT NULL,
Practical varchar(20) NOT NULL,
Bathrooms int NOT NULL,
Comfort varchar2(20) NOT NULL,
WiFi char NOT NULL ,
Price int NOT NULL,
Crib char NOT NULL ,
Entertainment varchar(20) NOT NULL,
Sofa char  NOT NULL,
Extra_bed char NOT NULL ,
Sleeps int NOT NULL,
Food_drink char NOT NULL,
Air_conditioner char NOT NULL,
```

```
Smoking char NOT NULL,
PRIMARY KEY (Room_type_id, Hotel_id),
FOREIGN KEY (Hotel_id) REFERENCES Hotels(Hotel_id) ON DELETE CASCADE
);

CREATE TABLE Room_amenities(
Room_type_id varchar2(10),
Hotel_id varchar(10),
Room_amenity_id varchar2(10),
Description varchar2(50),
PRIMARY KEY (Room_type_id, Hotel_id, Room_amenity_id ),
FOREIGN KEY (Room_type_id, Hotel_id) REFERENCES Room_type (Room_type_id,
Hotel_id) ON DELETE CASCADE
);


CREATE TABLE Room(
Room_no int,
Room_type_id varchar(10),
Hotel_id varchar(10),
PRIMARY KEY (Room_no, Room_type_id, Hotel_id),
FOREIGN KEY (Room_type_id, Hotel_id) REFERENCES Room_type (Room_type_id,
Hotel_id) ON DELETE CASCADE
);


CREATE TABLE Booking(
Booking_id VARCHAR(10),
Customer_id VARCHAR(10),
Hotel_id VARCHAR(10),
Payment_id VARCHAR(10),
Room_type_id VARCHAR(10),
Booking_date DATE NOT NULL,
CheckIn_date DATE NOT NULL,
CheckOut_date DATE NOT NULL,
Quote_price INT NOT NULL,
Completed CHAR NOT NULL,
No_of_rooms int NOT NULL,
Status VARCHAR(10),
PRIMARY KEY (Booking_id),
FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
FOREIGN KEY (Room_type_id, Hotel_id) REFERENCES Room_type (Room_type_id,
Hotel_id) ON DELETE CASCADE,
```

```
FOREIGN KEY (Payment_id) REFERENCES Payment_type(Payment_id) ON DELETE SET
NULL
);


CREATE TABLE Give_reviews_for(
Customer_id varchar2(20),
Hotel_id varchar2(20),
Trip_type varchar2(10),
Number_of_nights int,
Comments varchar2(40),
Rating int CHECK(Rating BETWEEN 1 AND 10) NOT NULL,
Review_date date NOT NULL,
PRIMARY KEY (Customer_id , Hotel_id),
FOREIGN KEY (Customer_id ) REFERENCES Customer(Customer_id) ON DELETE
CASCADE,
FOREIGN KEY (Hotel_id ) REFERENCES Hotels(Hotel_id) ON DELETE CASCADE
);


CREATE TABLE Offers(
Offer_id varchar2(10),
Hotel_id varchar2(10),
Description varchar2(40),
Discount int NOT NULL,
From_date date NOT NULL,
To_date date NOT NULL,
PRIMARY KEY (Offer_id ),
FOREIGN KEY(Hotel_id ) REFERENCES  Hotels(Hotel_id) ON DELETE CASCADE);


CREATE TABLE Places_to_visit(
Name varchar2(20) NOT NULL,
Street varchar2(100) NOT NULL,
City varchar2(10) NOT NULL,
Province varchar2(10) NOT NULL,
Zip int NOT NULL,
PRIMARY KEY (Name)
);


CREATE TABLE Hotels_around(
Hotel_id varchar2(10),
```

```
Name varchar2(20),
Distance float NOT NULL,
PRIMARY KEY (Hotel_id , Name),
FOREIGN KEY (Hotel_id ) REFERENCES  Hotels(Hotel_id ) ON DELETE CASCADE,
FOREIGN KEY (Name ) REFERENCES  Places_to_visit(Name ) ON DELETE CASCADE
);


CREATE TABLE tax_percentage(
Tax_id varchar2(10) NOT NULL,
Percentage float NOT NULL,
State varchar2(20) NOT NULL,
Country varchar2(20) NOT NULL,
PRIMARY KEY (Tax_id )
);
```

Chapter 7

PL\SQL

- **Triggers**

1) Trigger to update the rating of a hotel to the average rating after a customer gives review for the hotel:

```
CREATE or REPLACE TRIGGER update_rating
AFTER INSERT ON Give_reviews_for
FOR EACH ROW
BEGIN
      UPDATE Hotels SET rating= (SELECT AVG(rating) FROM Give_reviews_for WHERE Hotel_ID = :old.Hotel_ID);
END update_rating;
```

Query Result × | Script Output ×

Task completed in 0.205 seconds

Trigger UPDATE_RATING compiled

2) Trigger to calculate the distance of a place from the hotel after a new place has been inserted around the hotel. The following trigger shall make use of google map API. Input to the API function are the location of the hotel and location of the place:

```
CREATE or REPLACE TRIGGER cal_distance
AFTER INSERT ON Hotels_around
FOR EACH ROW
DECLARE
start_add varchar2(100);
end_add varchar2(100);
BEGIN
SELECT Street||city||province||zip FROM hotels INTO start_add WHERE hotel_id = old.hotel_id;
SELECT Street||city||province||zip FROM Places_to_visit INTO end_add WHERE hotel_id = old.name;
UPDATE Hotels_around SET distance =  google_api.find_distance_between(start_add, end_add)
WHERE hotel_id=old.hotel_id AND name=old.name;
END cal_distance;
```

Query Result × | Script Output ×

Task completed in 0.229 seconds

Trigger CAL_DISTANCE compiled

- **Procedures**

1) Procedure to add number of free nights to the customer account 72 hours each new booking. This procedure can be scheduled to be run every hour or every night:

```
CREATE or REPLACE PROCEDURE add_free_nights AS
nights customer.free_nights%TYPE;
checkin booking.checkin_date%TYPE;
checkout booking.checkout_date%TYPE;
CURSOR c1 IS
   SELECT    c.free_nights, b.checkin_date, b.checkout_date
   FROM    customer c, booking b
   WHERE b.customer_id=c.customer_id AND b.completed = 1;
BEGIN
   OPEN c1;
      LOOP
      FETCH c1 INTO nights, checkin, checkout;
            IF checkout <= sysdate-3 THEN
                  UPDATE customer
                  SET free_nights = nights+(checkout-checkin)/10;
            END IF;
         EXIT WHEN (c1%notfound);
      END LOOP;
   CLOSE c1;
END add_free_nights;
```

Query Result × Script Output ×

Task completed in 0.112 seconds

Procedure ADD_FREE_NIGHTS compiled

2) A dummy procedure to imitate initiating refund for a booking which was not successful:

```
CREATE or REPLACE PROCEDURE initiate_refund(booking_id booking.booking_id%TYPE)
as
begin
    DBMS_OUTPUT.put_line('Refund initiated');
End;
```

Script Output

Task completed in 0.033 seconds

```
Procedure INITIATE_REFUND compiled
```

3) Sequence to generate unique booking id for every entry into the booking table. This sequence is required for the below procedure:

```
CREATE SEQUENCE booking_seq
START WITH    1000
INCREMENT BY  1
NOCACHE
NOCYCLE;
```

Query Result | Script Output

Task completed in 0.137 seconds

```
Sequence BOOKING_SEQ created.
```

The below procedure makes a booking if payment is successful and initiates refund if there was a problem with the booking:

Worksheet | Query Builder

```
CREATE or REPLACE PROCEDURE make_booking
(Customer_id_ IN customer.customer_id%TYPE, Payment_id_ IN payment_type.payment_id%TYPE,
Room_type_id_ IN room_type.room_type_id%TYPE, Hotel_id_ IN hotels.Hotel_id%TYPE,
Checkin_ IN booking.checkin_date%TYPE, Checkout_ IN booking.checkout_date%TYPE,
offer_id_ IN offers.offer_id%TYPE, number_of_rooms_ IN booking.no_of_rooms%TYPE)
AS
number_of_days_ int;
discount_ float;
amount_ float;
free_nights_ int;
payment_result_ varchar2(20);
quote_price_ int;
tax_percent_ float;
BEGIN

number_of_days_ := checkout_ - checkin_;

select discount into discount_ from offers o
where o.offer_id=offer_id_ and o.hotel_id=hotel_id_ and o.from_date<=checkin_ and o.to_date>=checkout_;

select floor(free_nights) into free_nights_ from customer
where customer_id=customer_id_;

select price into quote_price_ from room_type
where hotel_id=hotel_id_ and room_type_id=room_type_id_;

select percentage into tax_percent_ from tax_percentage
where tax_id = (select tax_id from hotels where hotel_id = hotel_id_);

amount_ := (((number_of_days_ - free_nights_) * quote_price_) * (1 - discount_)) * (1+tax_percent_);

select make_payment (Payment_id_, amount_) into payment_result_ from dual;
```

Script Output | Task completed in 0.036 seconds

```
Procedure MAKE_BOOKING compiled
```

Worksheet | Query Builder

```
select make_payment (Payment_id_, amount_) into payment_result_ from dual;

if payment_result_ = 'Success'
then
    update customer set free_nights = free_nights-free_nights_ where customer_id=customer_id_;
    insert into booking(customer_id, hotel_id, payment_id, room_type_id, booking_date, checkin_date,
                        checkout_date, quote_price, completed, no_of_rooms, status)
    values (customer_id_, hotel_id_, payment_id_, room_type_id_, sysdate, checkin_,
            checkout_, amount_, null, number_of_rooms_, 'Success');
else
    insert into booking(booking_id, customer_id, hotel_id, payment_id, room_type_id, booking_date,
                        checkin_date, checkout_date, quote_price, completed, no_of_rooms, status)
    values (booking_seq.nextval, customer_id_, hotel_id_, payment_id_, room_type_id_, sysdate,
            checkin_, checkout_, amount_, null, number_of_rooms_, 'Failure');

    initiate_refund(booking_seq.currval);
end if;
END make_booking;
```

Script Output

Task completed in 0.036 seconds

```
Procedure MAKE_BOOKING compiled
```

- **Functions**

1) Function to fetch user readable hotel rating based on average of previous ratings:

```
CREATE or REPLACE FUNCTION hotel_ratings (hotel_id IN hotels.hotel_id %TYPE )
RETURN VARCHAR2 AS
rating_ hotels.rating%TYPE;
user_rating VARCHAR2(20);
BEGIN
  SELECT h.rating INTO rating_ FROM hotels h WHERE h.hotel_id=hotel_id;
  IF rating_<0.0 OR rating_>10.0
  THEN RAISE_APPLICATION_ERROR(-20010, 'Invalid Rating');
  ELSE IF rating_>=9.0
          THEN user_rating := 'Loved by guests';
          ELSE
                  user_rating := 'Enjoyed by guests';
          END IF;
   END IF;
   RETURN user_rating;
END;
```

Query Result × | Script Output ×

Task completed in 0.133 seconds

Function HOTEL_RATINGS compiled

2) A dummy function to imitate making payment for a booking and sending the status of the transaction to the Make_booking procedure:

Worksheet | Query Builder

```
create or replace FUNCTION make_payment (payment_id payment_type.payment_id%TYPE, amount float)
RETURN varchar2 AS
result_ varchar2(20);
BEGIN
    result_ := 'Success';
    RETURN result_;
END;
```

Script Output ×

Task completed in 0.104 seconds

Function MAKE_PAYMENT compiled