Database Systems

Sheradom Project

Fiorentino Nicola - 765448

A.Y. 2021-2022

I. Requirements Collection and Analysis

A. Specification

The company "Sheradom", listed on the Milan Stock Exchange, controls a chain of large hotels distributed throughout the national territory. In order to improve the service offered to customers, it intends to equip itself with a centralized IT system capable of managing both bookings and payment information for the service. To be able to book a room, customers will have to contact the Sheradom call center by telephone, email, or using a specific website. At the time of booking, the customer will specify the number of rooms (single or double) she/he wishes to book, the date of arrival (check-in) and the date of departure (check-out), as well as the number of nights she/he will stay. On the day of departure, the room must be left by 12.00. Obviously, the dates can be different for each room booked. The rooms of the hotel are distinguished in those for smokers and those for non-smokers. In addition, the booking person can specify whether he wants to book a parking space in the hotel garage or not. In the database the data relating to the booking person must be stored, also to identify and contact her/him. To confirm a reservation the customer will be asked to send the information related to his credit card or to make a bank transfer to pay in advance the amount due. When the customer leaves, the system must be able to print the room number in which he/she was hosted, the arrival and departure dates, the number of nights spent in the hotel, the price per night of the room, the total amount, and the balance.

Suppose that the following operations are carried out:
1. Recording a reservation at one of the hotels (50 times a day).
2. Confirmation of a reservation (30 times a day).
3. Request information on a free room for a certain period (100 times a day).
4. Confirmation of the client's arrival (100 times a day).
5. Customer's departure with invoice printing and immediate balance (90 times a day).
6. Print all information related to customers who have not paid yet (once a week).

Take into account that the chain includes 20 large hotels, distributed in 15 cities and that the total number of rooms is about 3000.

B. Structuring of requirements

The specification will be analyzed to remove ambiguities. In this context, booking and reservation represent the same concept: we will use the term booking. Customer and booking person are also synonyms: we will use the term customer. Moreover, the term number associated to the concept of room is ambiguous: it can be referred to the number of rooms booked by a customer or to a code that identifies the room. In the second case we will use the term code in order to avoid ambiguities. We will choose a correct level of abstraction and the sentence structure will be standardized. The final specifications are presented below.

General Phrases
The company "Sheradom", listed on the Milan Stock Exchange, controls a chain of large hotels distributed throughout the national territory. In order to improve the service offered to customers, it intends to implement a centralized IT system capable of managing both bookings and payment information for the service.
Phrases relating to Customer
For each customer we will hold the fiscal code, the name, the date of birth, the city, the sex, the phone number and the email.
Phrases relating to Booking
To make a booking, customers will have to contact the Sheradom call center by telephone, email or website. For each booking we will hold the ID, the rooms booked and if the customer needs a parking space in the hotel garage. For each room booked we will store the date of arrival, the date of departure and the number of nights. The dates can be different for each room booked. To confirm a booking the customer has to provide his credit card number or he has to pay in advance the amount due.
Phrases relating to Rooms
For each room in a hotel we will hold the code and the price per night. The rooms are divided into single and double. Moreover, they are distinguished in those for smokers and those for non-smokers.
Phrases relating to Departures
On the day of departure, the room must be left by 12.00. The system must be able to print the room code, the arrival and departure dates, the number of nights spent in the hotel, the price per night of the room, the total amount and the balance.

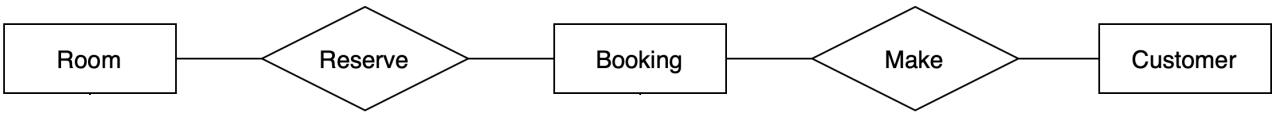
C. Glossary of terms

Term	Description	Synonyms	Connections
Customer	User who can book hotel rooms	Booking person	Booking
Hotel	Building containing multiple rooms		Room
Room	Hotel room that can be booked by a customer		Hotel
Booking	Reservation of hotel rooms made by a user	Reservation	Customer, Room
Invoice	Receipt indicating the amount paid by a customer		Booking, Room

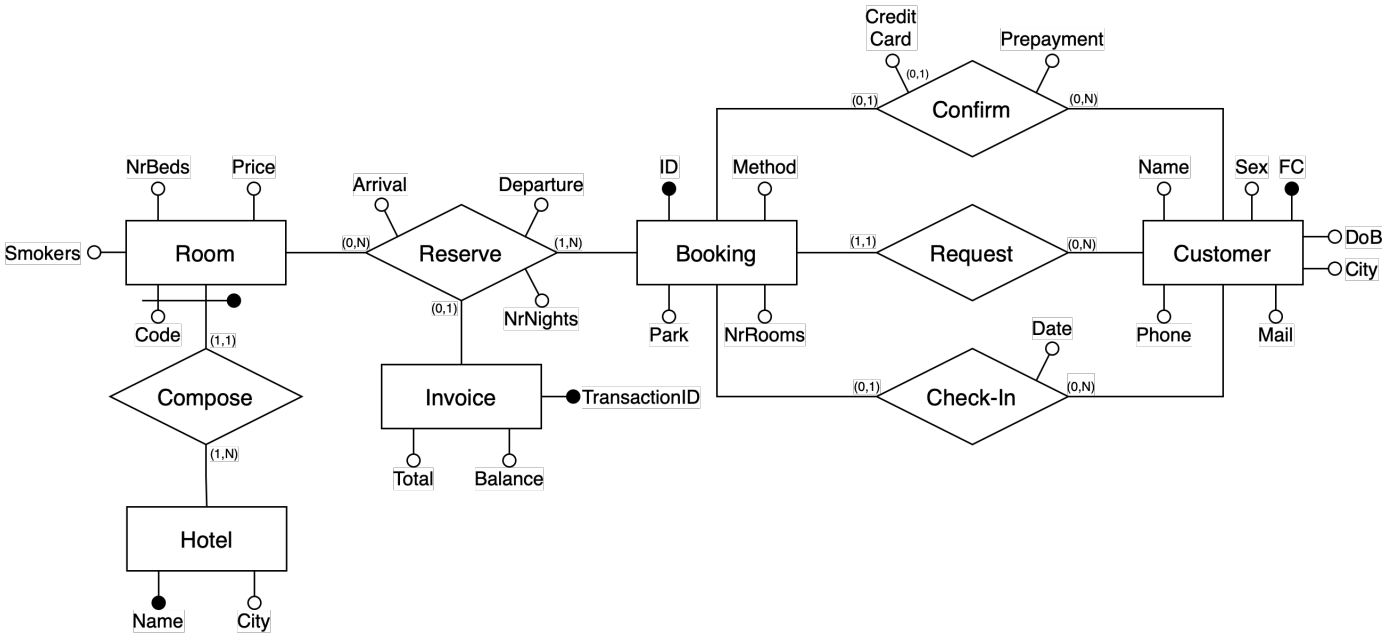
II. Conceptual Design

For the conceptual design we will use a hybrid approach. First a skeleton scheme is defined and then the individual concepts are modeled separately. The final scheme is obtained by integrating the different subschemes.

A. Skeleton scheme



B. Complete scheme



C. Business rules

Constraints:

- Values for sex are M, F.
- A customer can only confirm a requested booking.
- The credit card is provided for bookings without prepayment.
- Check-in requires a confirmed booking.
- Values for booking method are phone, mail, website.
- The departure date must follow the arrival date.
- Prepayment, park and smokers have boolean values.
- A room cannot have multiple bookings for the same day.
- The number of beds in a room can be one or two.
- Prices are expressed in euros.

Derivations:

- The number of rooms in a booking is obtained by adding the rooms booked.
- The number of nights is obtained by subtracting date of departure and date of arrival.
- The total of an invoice is obtained by multiplying price per night and number of nights and adding any additional costs.

III. Logical Design

A. Table of volumes

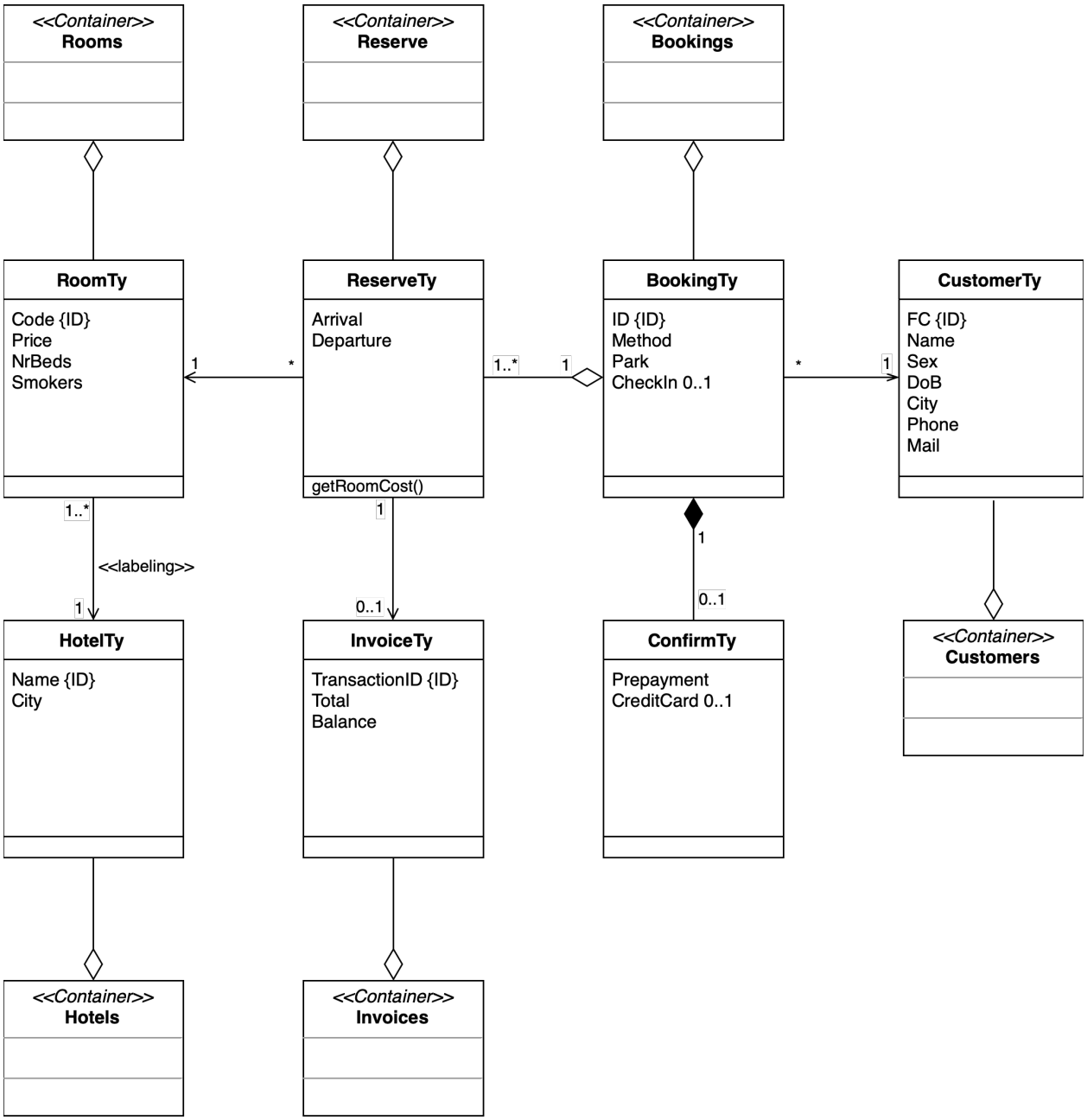
Concept	Type	Volume	Notes
Customer	E	50.000	Suppose the Sheradom chain has 50.000 customers
Request	R	150.000	Suppose each customer makes 3 bookings on average
Booking	E	150.000	There are 50.000*3 bookings in total
Confirm	R	135.000	Suppose 10% of bookings are not confirmed
Check-In	R	128.000	Suppose 15% of check-ins don't take place
Reserve	R	225.000	Suppose each booking reserves 1.5 rooms on average
Invoice	E	203.000	Invoices are 135.000*1.5
Room	E	3.000	Given by the specifications
Compose	R	150	Each hotel has 3000/20 rooms on average
Hotel	E	20	Given by the specifications

B. Redundancy analysis

- The number of rooms in a booking can be derived from an occurrence count. Since this attribute is not used in any post-booking operation, it can be removed without loss of performance.

- The number of nights spent in a room can be derived from a simple subtraction between attributes of the same concept. If we ignore the cost of arithmetic operations in main memory, the redundant attribute can be eliminated without loss of performance.

C. Logical scheme



D. Implementation

We will follow the object-relational model to implement the system. The DBMS used is Oracle.

Creation of User

```
-- USER SQL
CREATE USER "SHERADOM" IDENTIFIED BY "sheradom"
DEFAULT TABLESPACE "USERS";

-- QUOTAS
ALTER USER "SHERADOM" QUOTA UNLIMITED ON "USERS";

-- ROLES
GRANT "CONNECT" TO "SHERADOM" ;

-- SYSTEM PRIVILEGES
GRANT DROP ANY TRIGGER TO "SHERADOM" ;
GRANT CREATE TRIGGER TO "SHERADOM" ;
GRANT CREATE ANY PROCEDURE TO "SHERADOM" ;
GRANT CREATE ANY INDEX TO "SHERADOM" ;
GRANT CREATE TABLE TO "SHERADOM" ;
GRANT DROP ANY TABLE TO "SHERADOM" ;
GRANT CREATE TYPE TO "SHERADOM" ;
GRANT DROP ANY TYPE TO "SHERADOM" ;
GRANT EXECUTE ANY PROCEDURE TO "SHERADOM" ;
GRANT DROP ANY INDEX TO "SHERADOM" ;
GRANT INSERT ANY TABLE TO "SHERADOM" ;
GRANT DROP ANY PROCEDURE TO "SHERADOM" ;
GRANT CREATE ANY TRIGGER TO "SHERADOM" ;
GRANT CREATE ANY TABLE TO "SHERADOM" ;
GRANT CREATE ANY TYPE TO "SHERADOM" ;
```

We will assume that each booking cannot reserve more than ten rooms.

Definition of Types

```
CREATE OR REPLACE TYPE CustomerTy AS OBJECT(
FC CHAR(16),
CName VARCHAR2(30),
Sex CHAR(1),
DoB DATE,
City VARCHAR2(20),
Phone INTEGER(10),
Mail VARCHAR2(40));
/
CREATE OR REPLACE TYPE HotelTy AS OBJECT(
HName VARCHAR2(20),
City VARCHAR2(20));
/
CREATE OR REPLACE TYPE RoomTy AS OBJECT(
Code INTEGER(3),
Price NUMBER(6,2),
NrBeds INTEGER(1),
Smokers CHAR(1),
Hotel REF HotelTy);
/
CREATE OR REPLACE TYPE InvoiceTy AS OBJECT(
TransactionID INTEGER(10),
Total NUMBER(7,2),
Balance VARCHAR2(40));
/
CREATE OR REPLACE TYPE ReserveTy AS OBJECT(
Arrival DATE,
Departure DATE,
Room REF RoomTy,
Invoice REF InvoiceTy,
MEMBER FUNCTION getRoomCost RETURN NUMBER);
/
CREATE OR REPLACE TYPE BODY ReserveTy AS
MEMBER FUNCTION getRoomCost RETURN NUMBER IS
price NUMBER(6,2);
BEGIN
SELECT DEREF(Room).Price INTO price FROM DUAL;
RETURN (Departure - Arrival) * price;
END getRoomCost;
END;
/
CREATE OR REPLACE TYPE ReserveArray AS VARRAY(10) OF REF ReserveTy;
/
CREATE OR REPLACE TYPE ConfirmTy AS OBJECT(
Prepayment CHAR(1),
CreditCard INTEGER(16));
/
CREATE OR REPLACE TYPE BookingTy AS OBJECT(
BookingID INTEGER(6),
BookMethod VARCHAR2(7),
Park CHAR(1),
CheckIn DATE,
Customer REF CustomerTy,
Confirm ConfirmTy,
ReserveList ReserveArray);
/
```

Creation of Tables

```
CREATE TABLE Customers OF CustomerTy(
FC PRIMARY KEY,
CName NOT NULL,
Sex NOT NULL,
DoB NOT NULL,
City NOT NULL,
Phone NOT NULL);
/
CREATE TABLE Hotels OF HotelTy(
HName PRIMARY KEY,
City NOT NULL);
/
CREATE TABLE Rooms OF RoomTy(
Code NOT NULL,
Price NOT NULL,
NrBeds NOT NULL,
Smokers NOT NULL,
Hotel NOT NULL);
/
CREATE TABLE Invoices OF InvoiceTy(
TransactionID PRIMARY KEY,
Total NOT NULL,
Balance NOT NULL);
/
CREATE TABLE Reserve OF ReserveTy(
Arrival NOT NULL,
Departure NOT NULL,
Room NOT NULL);
/
CREATE TABLE Bookings OF BookingTy(
BookingID PRIMARY KEY,
BookMethod NOT NULL,
Park NOT NULL,
Customer NOT NULL,
ReserveList NOT NULL);
/
```

IV. Constraint management

The following constraints can be managed using triggers.

- A customer must provide a credit card number to confirm a booking without prepayment.
- It is not possible to register a check-in for unconfirmed bookings.
- The departure date must follow the arrival date.
- A room cannot have multiple bookings for the same day.
- When a booking is canceled, the rooms become available again.

Definition of triggers

```
CREATE OR REPLACE TRIGGER creditCardCheck
BEFORE INSERT OR UPDATE OF Confirm ON Bookings
FOR EACH ROW
WHEN (NEW.Confirm IS NOT NULL)
BEGIN
IF :NEW.Confirm.CreditCard IS NULL AND :NEW.Confirm.Prepayment<>'T' THEN
    RAISE_APPLICATION_ERROR(-20999, 'Pay for your booking or provide a credit card
number');
END IF;
END;
/
CREATE OR REPLACE TRIGGER validateCheckIn
BEFORE UPDATE OF CheckIn ON Bookings
FOR EACH ROW
BEGIN
IF :NEW.Confirm IS NULL THEN
    RAISE_APPLICATION_ERROR(-20998, 'Unconfirmed booking');
END IF;
END;
/
CREATE OR REPLACE TRIGGER bookingCheck
BEFORE INSERT ON Reserve
FOR EACH ROW
DECLARE
x INTEGER;
BEGIN
IF :NEW.Arrival >= :NEW.Departure THEN
    RAISE_APPLICATION_ERROR(-20997, 'Invalid dates');
END IF;
SELECT COUNT(*) INTO x FROM Reserve WHERE :NEW.Room = Room AND :NEW.Arrival < Departure
AND :NEW.Departure > Arrival;
IF x > 0 THEN
    RAISE_APPLICATION_ERROR(-20996, 'Room not available');
END IF;
END;
/
CREATE OR REPLACE TRIGGER makeRoomsAvailable
AFTER DELETE ON Bookings
FOR EACH ROW
DECLARE
i INTEGER;
BEGIN
FOR i IN 1..:OLD.ReserveList.COUNT LOOP
    DELETE FROM Reserve r WHERE REF(r)=:OLD.ReserveList(i);
END LOOP;
END;
/
```

V. Procedures

Procedures for performing frequent operations are defined below.

Recording a reservation at one of the hotels

```
CREATE OR REPLACE PROCEDURE recordBooking(bookID INTEGER, met VARCHAR2, park CHAR, custFC CHAR,
arr DATE, dep DATE, hotelName VARCHAR2, room INTEGER) AS
res REF ReserveTy;
BEGIN
INSERT INTO Reserve r VALUES (arr, dep, (SELECT REF(r) FROM Rooms r WHERE Code=room AND
DEREF(Hotel).HName=hotelName), NULL) RETURNING REF(r) INTO res;
INSERT INTO Bookings VALUES (bookID, met, park, NULL, (SELECT REF(c) FROM Customers c WHERE
FC=custFC), NULL, ReserveArray(res));
END;
/
```

Confirmation of a reservation

```
CREATE OR REPLACE PROCEDURE confirmBooking(bookID INTEGER, prepayment CHAR, card INTEGER) AS
BEGIN
UPDATE Bookings SET Confirm=ConfirmTy(prepayment, card) WHERE BookingID=bookID;
END;
/
```

Request information on a free room for a certain period

```
CREATE OR REPLACE FUNCTION roomAvailability(hotelName VARCHAR2, room INTEGER, startDate DATE,
endDate DATE) RETURN VARCHAR2 AS
i INTEGER;
d DATE := startDate;
r REF RoomTy;
results VARCHAR2(30000);
BEGIN
SELECT REF(ro) into r FROM Rooms ro WHERE Code=room AND DEREF(Hotel).HName=hotelName;
WHILE d < endDate LOOP
    SELECT COUNT(*) INTO i FROM Reserve WHERE Room=r AND d >= Arrival AND d < Departure;
    IF i = 0 THEN
        results := results || d || ': Available' || CHR(10);
    ELSE
        results := results || d || ': Not Available' || CHR(10);
    END IF;
    d := d + 1;
END LOOP;
RETURN results;
END;
/
```

Confirmation of the client's arrival

```
CREATE OR REPLACE PROCEDURE confirmArrival(bookID INTEGER) AS
BEGIN
UPDATE Bookings SET CheckIn=SYSDATE WHERE BookingID=bookID;
END;
/
```

Customer's departure with invoice printing and immediate balance

```
CREATE OR REPLACE PROCEDURE printInvoice(bookID INTEGER, hotelName VARCHAR2, roomCode INTEGER,
transID INTEGER, additional NUMBER) AS
i INTEGER;
x INTEGER;
rList ReserveArray;
roomCost Invoices.Total%TYPE;
inv InvoiceTy;
iRef REF InvoiceTy;
BEGIN
SELECT ReserveList INTO rList FROM Bookings WHERE BookingID=bookID;
FOR i IN 1..rList.COUNT LOOP
    SELECT COUNT(*) INTO x FROM DUAL WHERE DEREF(DEREF(rList(i)).Room).Code=roomCode AND
DEREF(DEREF(DEREF(rList(i)).Room).Hotel).HName=hotelName;
    IF x <> 0 THEN
        SELECT DEREF(rList(i)).getRoomCost() INTO roomCost FROM DUAL;
        inv := InvoiceTy(transID, roomCost + additional, 'Room: ' || roomCost || ' Additional: '
|| additional);
        INSERT INTO Invoices v VALUES inv RETURNING REF(v) INTO iRef;
        UPDATE Reserve res SET Invoice=iRef WHERE REF(res)=rList(i);
        DBMS_OUTPUT.PUT_LINE(hotelName || ':' || roomCode || ' ' || inv.Balance);
    END IF;
END LOOP;
IF inv IS NULL THEN
    RAISE_APPLICATION_ERROR(-20995, 'Wrong data');
END IF;
END;
/
```

Print all information related to customers who have not paid yet

```
CREATE OR REPLACE PROCEDURE unpaidCustomers AS
CURSOR cur IS SELECT * FROM Bookings WHERE Confirm IS NULL;
c CustomerTy;
BEGIN
FOR b in cur LOOP
    SELECT DEREF(b.Customer) INTO c FROM DUAL;
    DBMS_OUTPUT.PUT_LINE(c.FC || ' ' || c.CName || ' ' || c.DoB);
END LOOP;
END;
/
```

VI. Database population

The database will be populated according to the defined volumes. In this operation, dummy data will be used.

Customers

```
DECLARE
i INTEGER;
nCustomers INTEGER := 50000;
type nameArray IS VARRAY(12) OF Customers.CName%TYPE;
type sexArray IS VARRAY(2) OF Customers.Sex%TYPE;
type dateArray IS VARRAY(5) OF Customers.DoB%TYPE;
type cityArray IS VARRAY(10) OF Customers.City%TYPE;
cNames nameArray :=
nameArray('John','Klaire','Luke','Marie','Mike','Lucy','Rick','Nicole','George','Andrea','Karl','E
lizabeth');
sexes sexArray := sexArray('M', 'F');
dates dateArray := dateArray('23-MAY-1998','10-FEB-1980','12-JUL-1990','26-DEC-1991','6-
SEP-1996');
cities cityArray := cityArray('Rome', 'Milan', 'Florence', 'Bari', 'Turin', 'Venice', 'Verona',
'Palermo', 'Naples', 'Genova');
BEGIN
FOR i IN 1..nCustomers LOOP
    INSERT INTO Customers VALUES (dbms_random.string('X', 16), cNames(1 + MOD(i, 12)), sexes(1 +
MOD(i, 2)), TO_DATE(TRUNC(DBMS_RANDOM.VALUE(2451545, 5373484)), 'J'),
cities(DBMS_RANDOM.VALUE(1,10)), DBMS_RANDOM.VALUE(3000000000,4000000000), dbms_random.string('l',
8) || '@mail.com');
END LOOP;
END;
/
```

Hotels

```
DECLARE
i INTEGER;
nHotels INTEGER := 20;
type cityArray IS VARRAY(15) OF Hotels.City%TYPE;
cities cityArray := cityArray('Rome', 'Milan', 'Florence', 'Bari', 'Turin', 'Venice', 'Verona',
'Palermo', 'Naples', 'Genova', 'Ancona', 'Pisa', 'Lecce', 'Cagliari', 'Aosta');
BEGIN
FOR i IN 1..nHotels LOOP
    INSERT INTO Hotels VALUES ('SH' || i, cities(DBMS_RANDOM.VALUE(1,15)));
END LOOP;
END;
/
```

Rooms

```
DECLARE
i INTEGER;
j INTEGER;
nHotels INTEGER := 20;
nRooms INTEGER := 150;
hotel Rooms.Hotel%TYPE;
type smokersArray IS VARRAY(2) OF Rooms.Smokers%TYPE;
smokers smokersArray := smokersArray('T', 'F');
BEGIN
FOR i IN 1..nHotels LOOP
    SELECT REF(h) into hotel from Hotels h WHERE HName = 'SH' || i;
    FOR j IN 1..nRooms LOOP
        INSERT INTO Rooms VALUES (j, TRUNC(DBMS_RANDOM.VALUE(50, 1500),2), DBMS_RANDOM.VALUE(1,2),
smokers(DBMS_RANDOM.VALUE(1,2)), hotel);
    END LOOP;
END LOOP;
END;
/
```

Bookings - Reserve - Invoices

```
ALTER TRIGGER bookingCheck DISABLE;
/
SET SERVEROUTPUT OFF;
DECLARE
i INTEGER;
nBookings INTEGER := 150000;
type methodArray IS VARRAY(3) OF Bookings.BookMethod%TYPE;
type parkArray IS VARRAY(2) OF Bookings.Park%TYPE;
met methodArray := methodArray('Phone', 'Mail', 'Website');
park parkArray := parkArray('T', 'F');
d DATE;
cust Customers.FC%TYPE;
room Rooms.Code%TYPE;
hotel INTEGER;
BEGIN
FOR i IN 1..nBookings LOOP
    SELECT FC INTO cust FROM Customers SAMPLE(1) FETCH NEXT 1 ROWS ONLY;
    d := TO_DATE(TRUNC(DBMS_RANDOM.VALUE(2451545, 2460000)), 'J');
    hotel := DBMS_RANDOM.VALUE(1,20);
    room := DBMS_RANDOM.VALUE(1,150);
    recordBooking(100000 + i, met(DBMS_RANDOM.VALUE(1,3)), park(DBMS_RANDOM.VALUE(1,2)), cust, d,
d + 2, 'SH' || hotel, room);
    IF (DBMS_RANDOM.VALUE(1,10) <= 9) THEN
        confirmBooking(100000 + i, 'T', NULL);
        printInvoice(100000 + i, 'SH' || hotel, room, 1000000000 + i, TRUNC(DBMS_RANDOM.VALUE(0,
100),2));
    END IF;
END LOOP;
END;
/
ALTER TRIGGER bookingCheck ENABLE;
/
```

VII. Query Optimization

The extensive use of references avoids expensive join operations. Therefore we will look for secondary indexes that allow to optimize selection operations. For this analysis, the tables Rooms (3000) and Hotels (20) are too small to be considered. The frequent selections on Customers and Bookings always concern the primary key which is already indexed. So the most relevant table is Reserve which contains the rooms associated with each booking. This table is queried whenever the user requests information on the availability of a room. Furthermore, before a booking is recorded in the system, the bookingCheck trigger consults this table to check for possible conflicting bookings.
The table attributes involved in selection operations are *Room*, *Arrival* and *Departure*. Since *Room* is a reference attribute, it cannot be indexed. So the candidate attributes are *Arrival* and *Departure*. The chosen index allows to reduce the cost of the test query by approximately 95%. Details are shown below.

Test query

```
SELECT COUNT(*) FROM Reserve WHERE Room=(SELECT REF(r) FROM Rooms r WHERE Code=125 AND
DEREF(Hotel).HName='SH8') AND '01-JAN-2018' >= Arrival AND '10-JAN-2018' < Departure;
```

Index

```
CREATE INDEX dateIndex ON Reserve(Arrival DESC, Departure DESC);
```

Cost without index

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	663
SORT		AGGREGATE	1	
TABLE ACCESS	RESERVE	FULL	258	650

Cost with index

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	34
SORT		AGGREGATE	1	
TABLE ACCESS	RESERVE	BY INDEX ROWID BATCHED	258	21

VIII. GUI for Room Availability

We will implement a graphical interface that allows easy checking of room availability. For this purpose we will use Java language and Oracle JDBC. The source code of the project can be found in the CheckRoom folder. The user specifies hotel, room, check-in and check-out dates. The application will provide room availability for each selected day. The following string can be used to run the application. Note that JavaFX must be installed on the machine.

```
java --module-path PATH_TO_JAVAFX_LIB --add-
modules=javafx.controls,javafx.fxml -jar PATH_TO_CheckRoom.jar
```

An example of the application use is shown below.