

Project for Database Design:

Phase IV Documentation

Kinjal Basu

kxb170730@utdallas.edu

Piyush Mahatkar

pkm170230@utdallas.edu

Pre-Illumination

In this project report we will follow the requirement of Phase IV directly. In Section 1 we gave problem description copied from Web site; in Section 2 we answered 3 questions listed in the project and justified our solution; in Section 3 we exhibited EER diagram with all assumptions; in Section 4 we showed our relational schema after normalization; in Section 5 we gave all requested SQL statements for both views and queries; and in Section 6 we gave dependency diagram induced from relational schemas. Finally, a short summary is given at the end of this report.

1. Problem Description

Design, develop, and test a database for ABC hotel. The project consists of four parts: conceptual database design (Phase I), logical database design (Phase II), Oracle relational database implementation (Phase III), and final report & demo (Phase IV).

1. The hotel has a group of employees. Each employee has a unique ID number (9-digit number), name, age, address (street number, street name, city, state, zip code), and salary rate. The employees are categorized as discussed below.
2. Employees are categorized based on their job functions: management, reception, dining, housekeeping and concierge, there are also other types (tech support, accountants, etc.) of employees who do not fall into one of the above main types.
3. For each management team member, the system records his/her title (general manager, customer relationship manager, revenue executive, event manager, etc.).
4. For each receptionist, the system tracks a list of languages he/she can speak and read.

5. For each dining staff, the system keeps his/her shift (morning, afternoon, evening, night). Dining staff can be further categorized based on the type of dining they serve: lounge/bar, fine dining restaurant, buffet, catering, etc.
6. For housekeeping staff and concierge, the system tracks their years of experience (integer between 0 to 50). For tech support and accountants, the system tracks the licenses they have obtained.
7. The hotel has two main types of clients: individuals and organizations.
8. The system tracks the information of each individual customer: unique ID (6-digit number), name, sex, phone (may be multiple phone numbers, in the format of '(xxx)xxx-xxxx'), and date of birth (both in the format of 'MM/DD/YYYY').
9. An individual customer may or may not be a member of the hotel's rewards program. Information about the membership is recorded for a customer if he/she has one. A customer may have multiple membership numbers.
10. The hotel tracks the information of hotel rooms. Each room has a unique room number (4-digit number), bed type (twin, king, double queen, etc.), room type (standard, premium, suite, etc.), and per-night price (in USD).
11. Housekeepers clean hotel rooms, the date, time and the housekeeper's ID are recorded every time a room is cleaned.
12. Receptionists help individual customers check in to their hotel rooms. An individual customer can be helped by different receptionists check in to different hotel rooms during multiple stays at the hotel. Each time a customer is checked in, the check-in date ('MM/DD/YYYY'), time ('HH-MM-SS'), key type (card key or digital key), lounge access (Yes/No), and length of the stay are recorded.
13. An individual customer needs to pay his/her bill at check out. Date issued, check-in and checkout date, bill amount in USD are recorded for each bill. A customer may make multiple payments to pay off one bill. The date, time and the payment amount in USD are recorded for each payment.
14. An organization client is uniquely identified by an ID (6-digit number), and may or may not have a direct bill account with the hotel. The account number is recorded if a client has one. A client may have multiple direct account numbers, but the account number itself cannot uniquely identify the account.
15. Organization clients may hold events at the hotel. An organization can hold multiple events, and an event can be held by multiple organizations. For each event, a deposit must be paid to the hotel, and the amount in USD is recorded.
16. The hotel assembles event staff to help serve the events. Four main types of event staff come from management, catering, tech support and accountants. Every event staff member is equipped with an on-call speaker and the on-call number (4-digit number) is recorded in the system. An event staff member can serve multiple events. Each event can be served by a group of event staff, and is uniquely

identified by an event ID (4-digit number). The date and time of an event are also recorded. An event manager from the management team is assigned for each event; the manager's ID (9- digit number) is recorded.

17. An accountant prepares a bill for each event. Each bill has a unique ID (6-digit number), date issued ('MM/DD/YYYY') and total amount in USD. An organization may have multiple bills to pay, and each bill may be paid by multiple organizations. An organization can make multiple payments to a bill. Each time an organization makes a payment, the system records the type of payment (cash, check, credit cards, etc.), amount in USD, date and time of the payment.

2. Three Questions

2.2.1 Is the ability to model super-class / subclass relationships likely to be important in such environment? Why or why not?

Yes, There is a need of having generalization (in case of setting up the group for event staff) and Specialization (in case of Employees with different roles). It is important for situations where only some category (Accountant and technical support) of employees needs to keep a license, whereas there are some common attributes for all kind of employees (Eg Name, Age and so on.). Event staff is a group of employees that is formed, where different subgrouping of employees need to be formed (Combination of management, accountant, catering and tech-support), which needs to be a union set.

2.2.2 Can you think of 5 more rules (other than the one explicitly described above) that are likely to be used in a school environment? Add your rules to the above requirement to be implemented.

- [1] A single customer cannot book, with double entries for booking.
- [2] Constraint on the maximum number of Rooms a customer can book.
- [3] Assignment of housekeepers can be in rotation rather than fixed room cleaning.
- [4] A Receptionist is being assigned to one customer. But the job of receptionist cannot be more than 8 hours. We need to consider another receptionist for hours other than that. Receptionist can have rotation in shifts.
- [5] Separate demands (which involves separate pricing) from the customer, needs to go through accountant (here accountant is handling only organizational event client)

2.3 Justify using a Relational DBMS like Oracle for this project.

Here the data has a simple tabular structure, like an accounting spreadsheet.

Data such as geo-spatial, engineering parts, or molecular modeling, on the other hand, tends to be very complex. It may have multiple levels of nesting and the model can be complicated (which need No-SQL DB), whereas RDBMS is used in case of 2D-row-column data arrangement.

3. EER diagram with all assumptions

Assumptions:

- Receptionist knows at least one language.
- Bill can be paid multiple times within checkout date
- If any event is organized, then at least one event staff crew is required.
- If any employee is a manager then emp_id is equal to manager_id
- An accountant is assigned for an event to record the expenses and prepare the Bill
- Payment is related to only one Organization Client.
- Organization Clients are Individual customer has no overlapping.
- Every individual does booking of at least one room.
- Every individual customer has been assigned at least one receptionist.

Following is the EER Diagram:

4. Relational Schema in Third Normal Form

4.1 Relational Schema

Following page has a Relational Schema.

4.2 Explanation for format design

Rules and assumptions:

- Employee_ID is a 9-Digit Number.
- Years_of_experience for housekeeping_staff is range(0-50)
- Individual_Customer id is a 6 digit number.
- Phone number for customer is a 10 digit number of format (xxx)xxx-xxxx
- Date_of_Birth format is MM/DD/YYYY
- Room number is a 4 digit number.
- check-in date ('MM/DD/YYYY') + time ('HH-MM-SS')
- key type has one of the two values (card key or digital key),
- lounge access has one of the two values (Yes/No)
- organization client is uniquely identified by an ID (6-digit number)
- Every event staff member is assigned on-call number (4-digit number)
- Each event has an event ID (4-digit number).
- Each bill has a unique ID (6-digit number), with date issued format ('MM/DD/YYYY') and total amount in numbers.

Format for Every Relation

Customer Activity:	Data Type
Individual_Customer_ID	Integer (6 digit)
Check_In	MM/DD/YYYY
Check_Out	MM/DD/YYYY
Key_Type	Boolean
Lounge_Access	(Card(0)/Digital(1))
Length_of_Stay	Boolean(Yes/No)
	Integer

CUSTOMER:	Data Type
Customer_Id	Integer (6 Digit)

ORGANIZATION_EVENT_PAY_DETAILS	Data Type
Payment_Id	Integer
Payment_Mode	String <chars(20)>
Amount	Integer
Date	MM/DD/YYYY
Time	HH:MM:SS
Org_Cust_Id	Integer (6 digit)

DIRECT_BILL_ACCOUNT	Data Type
Organizational_Customer_Id	Integer (6 digit)
Account_No	Integer

PAYMENT_THROUGH_DETAILS	Data Type
Payment_Id	Integer
Bill_Id	Integer(6 Digit)

ORGANIZATION_EVENT_BILL_DETAILS	Data Type
BILL_Id	Integer (6 digit)
Amount	Integer
Bill_Date	MM/DD/YYYY
Accountant_ID	Integer
Event_ID	Integer (4 digit)

ORGANIZATIONAL_CLIENT	Data Type
Customer_Id	Integer (6 digit)

INDIVIDUAL_CLIENT	Data Type
Customer_Id	Integer (6 digit)
Name	String (20 Chars)
Sex	M/F
Date_of_Birth	MM/DD/YYYY
Receptionist_Id	Integer

EVENT_HOST	Data Type
Event_Id	Integer (4 digit)
Organizational_Client_Id	Integer (6 digit)
MANAGEMENT	Data Type
Employee_Id	Integer (9 digit)
Manager_Title	String (20 chars)
Event_Staff_Id	Integer

ACCOUNTANT	DATATYPE
Employee_Id	Integer (9 digit)
License_No	Integer
Event_Staff_Id	Integer
Event_Staff_Id	Integer

TECH_SUPPORT	Data Type
Employee_Id	Integer (9 digit)
License_No	Integer
Event_Staff_Id	Integer

DINNING_STAFF	Data Type
Employee_Id	Integer (9 digit)
Shift	string
Dining_Type	string
Event_Staff_Id	Integer

HOUSEKEEPING_CONCIERGE	Data Type
Employee_Id	Integer (9 digit)
Year_of_Experience	Integer (range (0-50))

RECEPTIONIST	Data Type
Employee_Id	Integer (9 digit)
Year_of_Experience	Integer (range (0-50))

EMPLOYEE	Data Type
Employee_Id	Integer (9 digit)
Name	String
Age	Integer
Salary	Integer
Street_No	Integer
Street_Name	String
City	String
State	String
Zip	Integer

EVENT	Data Type
Event_Id	Integer (4 digit)
Time	HH:MM:SS
Date	MM/DD/YYYY
Deposit	Integer
Manager_Id	Integer
Accountant_Id	Integer

EVENT_STAFF	Data Type
Event_Staff_Id	Integer
On_Call_Number	Integer (4
On_Call_Speaker_Number	digit)
	Integer

EVENT_ORGANIZED_BY	Data Type
Event_Id	Integer (4
Event_Staff_Id	digit)
	Integer

INDIVIDUAL_CUSTOMER_BILL_DETAILS	Data Type
Bill_Id	Integer (6 digit)
Bill_Amount	Integer
Customer_Id	Integer (6 digit)

INDIVIDUAL_CLIENT_PHONE_NO	Data Type
Individual_Customer_Id	Integer (6 digit)
Phone_Number	Integer (10 digit)

PAYMENT	Data Type
---------	-----------

Payment_Id	Integer
Payment_Date	MM/DD/YYYY
Payment_Time	HH:MM:SS
Bill_Id	Integer (6 digit)
Individual_Customer_Id	Integer (6 digit)

BOOKING_DETAILS	Data Type
Individual_Customer_Id	Integer (6 digit)
Room_Number	Integer (4 digit)

INDIVIDUAL_CLIENT_REWARD_MEMBER_NO	Data Type
Individual_Customer_Id	Integer (6 digit)
Member_No	Integer

HOTEL_ROOMS	Data Type
Room_No	Integer (4 digit)
Bed_Type	String
Room_Type	String
Price	Integer
Availability	Boolean
Housekeeping_Id	Integer

ACCOUNTANT	Data Type
Employee_Id	Integer (9 digit)

License_No	Integer
Event_Staff_Id	Integer

5. All Requested SQL Statements

5.1 Creation of Database with SQL Statements

5.1.1 Table Creation

1.

```
CREATE TABLE EMPLOYEE (  
  Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
  Manager_Titlte varchar(100),  
  Age NUMBER(3) CHECK (Age >=0),
```

```
  Salary NUMBER(20) CHECK (Salary >=0),  
  Street_no NUMBER(10),  
  Street_name varchar(30),  
  City varchar(20),  
  State varchar(20),  
  Zip NUMBER(32)  
);
```

2.

```
CREATE TABLE RECEPTIONIST (  
  Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
  FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id)  
);
```

3.

```
CREATE TABLE HOUSEKEEPING_CONCIERGE (  
  Employee_Id NUMBER(9) NOT NULL  
  PRIMARY KEY, Years_of_Experience NUMBER(2) CHECK (Years_of_Experience >=0 AND  
  Years_of_Experience<=50), FOREIGN KEY (Employee_Id) REFERENCES  
  EMPLOYEE(Employee_Id));
```

4.

```
CREATE TABLE EVENT_STAFF (  
  Event_Staff_Id NUMBER(9) NOT NULL PRIMARY KEY,  
  On_Call_Number NUMBER(4) NOT NULL,  
  On_Call_Speaker_Number NUMBER(20) NOT NULL  
);
```

5.

```
CREATE TABLE TECH_SUPPORT (  
  Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
  FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id)
```

```
Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
License_No NUMBER(20) NOT NULL,  
Event_Staff_Id NUMBER(9),  
FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id), FOREIGN KEY  
(Event_Staff_Id) REFERENCES EVENT_STAFF(EVENT_STAFF_ID)  
);
```

6.

```
CREATE TABLE ACCOUNTANT (  
Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
License_No NUMBER(20) NOT NULL,  
Event_Staff_Id NUMBER(9),  
FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id), FOREIGN KEY  
(Event_Staff_Id) REFERENCES EVENT_STAFF(EVENT_STAFF_ID)  
);
```

7.

```
CREATE TABLE RECEPTIONIST_LANGUAGES_KNOWN ( Employee_Id NUMBER(9) NOT NULL,  
Language_Name VARCHAR(30) NOT NULL,  
FOREIGN KEY (Employee_Id) REFERENCES RECEPTIONIST(Employee_Id), PRIMARY KEY  
(Employee_Id, Language_Name)  
);
```

8.

```
CREATE TABLE DINING_STAFF (  
Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
Shift VARCHAR(20) NOT NULL CHECK (Shift IN ('morning', 'afternoon', 'evening', 'night')),  
Dining_Type VARCHAR(20) NOT NULL,  
Event_Staff_Id NUMBER(9),  
FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id), FOREIGN KEY  
(Event_Staff_Id) REFERENCES EVENT_STAFF(EVENT_STAFF_ID)  
);
```

9.

```
CREATE TABLE MANAGEMENT (  
Employee_Id NUMBER(9) NOT NULL PRIMARY KEY,  
Manager_Title VARCHAR(20) NOT NULL,  
Event_Staff_Id NUMBER(9),  
FOREIGN KEY (Employee_Id) REFERENCES EMPLOYEE(Employee_Id), FOREIGN KEY  
(Event_Staff_Id) REFERENCES EVENT_STAFF(EVENT_STAFF_ID)  
);
```

10.

```
CREATE TABLE EVENT(  
EVENT_ID NUMBER(4) NOT NULL PRIMARY KEY,  
EVENT_TIME TIMESTAMP NOT NULL,  
EVENT_DATE DATE NOT NULL,  
DEPOSIT NUMBER(20),  
MANAGER_ID NUMBER(9) NOT NULL,
```

ACCOUNTANT_ID NUMBER(9)NOT NULL,

FOREIGN KEY (MANAGER_ID) REFERENCES MANAGEMENT(EMPLOYEE_ID), FOREIGN KEY
(ACCOUNTANT_ID) REFERENCES ACCOUNTANT(EMPLOYEE_ID)
);

11.

CREATE TABLE EVENT_ORGANIZED_BY (
EVENT_ID NUMBER(4) NOT NULL,
EVENT_STAFF_ID NUMBER(9) NOT NULL,
FOREIGN KEY (EVENT_ID) REFERENCES EVENT(EVENT_ID),
FOREIGN KEY (EVENT_STAFF_ID) REFERENCES EVENT_STAFF(EVENT_STAFF_ID), PRIMARY
KEY (EVENT_ID,EVENT_STAFF_ID)
);

12. CREATE TABLE CUSTOMER(
CUSTOMER_ID NUMBER(6) NOT NULL PRIMARY KEY
);

13.

CREATE TABLE INDIVIDUAL_CLIENT(
CUSTOMER_ID NUMBER(6) NOT NULL PRIMARY KEY,
CUST_NAME VARCHAR(30) NOT NULL,
SEX VARCHAR(10) NOT NULL,
DOB DATE NOT NULL,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID)
);

14.

CREATE TABLE ORGANIZATIONAL_CLIENT(
CUSTOMER_ID NUMBER(6) NOT NULL PRIMARY KEY,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID)
);

15.

CREATE TABLE CUSTOMER_ACTIVITY(
INDIVIDUAL_CUSTOMER_ID NUMBER(6) NOT NULL,
CHECK_IN_DATE DATE NOT NULL,
CHECK_IN_TIME TIMESTAMP NOT NULL,
CHECK_OUT_DATE DATE,
CHECK_OUT_TIME TIMESTAMP,
KEY_TYPE VARCHAR(10) NOT NULL CHECK(KEY_TYPE IN ('card key','digital key')),
LOUNGE_ACCESS VARCHAR(5) CHECK(LOUNGE_ACCESS IN ('yes','no')) NOT NULL,
LENGTH_OF_STAY NUMBER(2) CHECK(LENGTH_OF_STAY >=0), RECEPTIONIST_ID
NUMBER(9) NOT NULL,
PRIMARY KEY (INDIVIDUAL_CUSTOMER_ID, CHECK_IN_DATE, CHECK_IN_TIME), FOREIGN
KEY (RECEPTIONIST_ID) REFERENCES RECEPTIONIST(EMPLOYEE_ID),

```
FOREIGN KEY (INDIVIDUAL_CUSTOMER_ID) REFERENCES  
INDIVIDUAL_CLIENT(CUSTOMER_ID)  
);
```

16.

```
CREATE TABLE INDIVIDUAL_CUST_BILL_DETAILS(  
BILL_ID NUMBER(10) NOT NULL PRIMARY KEY,  
BILL_AMOUNT NUMBER(6) NOT NULL CHECK(BILL_AMOUNT >= 0),  
INDIVIDUAL_CUSTOMER_ID NUMBER(6) NOT NULL,  
FOREIGN KEY (INDIVIDUAL_CUSTOMER_ID) REFERENCES  
INDIVIDUAL_CLIENT(CUSTOMER_ID)  
);
```

17.

```
CREATE TABLE INDIVIDUAL_PAYMENT(  
PAYMENT_ID NUMBER(10) NOT NULL PRIMARY KEY,  
PAYMENT_AMOUNT NUMBER(6) NOT NULL CHECK(PAYMENT_AMOUNT >= 0),  
PAYMENT_DATE DATE NOT NULL,  
PAYMENT_TIME TIMESTAMP NOT NULL,  
INDIVIDUAL_CUSTOMER_ID NUMBER(6) NOT NULL,  
BILL_ID NUMBER(10) NOT NULL,  
FOREIGN KEY (INDIVIDUAL_CUSTOMER_ID) REFERENCES  
INDIVIDUAL_CLIENT(CUSTOMER_ID), FOREIGN KEY (BILL_ID) REFERENCES  
INDIVIDUAL_CUST_BILL_DETAILS(BILL_ID)  
);
```

18.

```
CREATE TABLE INDIVIDUAL_CUST_REWARD( INDIVIDUAL_CUSTOMER_ID NUMBER(6) NOT  
NULL, MEMBER_NO NUMBER(10) NOT NULL,  
PRIMARY KEY (INDIVIDUAL_CUSTOMER_ID, MEMBER_NO),  
FOREIGN KEY (INDIVIDUAL_CUSTOMER_ID) REFERENCES  
INDIVIDUAL_CLIENT(CUSTOMER_ID)  
);
```

19.

```
CREATE TABLE HOTEL_ROOMS(  
ROOM_NO NUMBER(4) NOT NULL PRIMARY KEY,  
BED_TYPE VARCHAR(10) NOT NULL,  
ROOM_TYPE VARCHAR(10) NOT NULL,  
PRICE NUMBER(10) NOT NULL CHECK(PRICE >= 0),  
ROOM_AVAILABILITY VARCHAR(5) CHECK(ROOM_AVAILABILITY IN ('yes','no')),  
HOUSEKEEPING_ID NUMBER(9) NOT NULL,  
FOREIGN KEY (HOUSEKEEPING_ID) REFERENCES  
HOUSEKEEPING_CONCIERGE(EMPLOYEE_ID) );
```


20.

```
CREATE TABLE BOOKING_DETAILS (  
  INDIVIDUAL_CUST_ID NUMBER(6) NOT NULL,  
  ROOM_NO NUMBER(4) NOT NULL,  
  FOREIGN KEY (INDIVIDUAL_CUST_ID) REFERENCES INDIVIDUAL_CLIENT(CUSTOMER_ID),  
  FOREIGN KEY (ROOM_NO) REFERENCES HOTEL_ROOMS(ROOM_NO), PRIMARY KEY  
  (INDIVIDUAL_CUST_ID,ROOM_NO)  
);
```

21.

```
CREATE TABLE EVENT_HOST (  
  ORGANIZATIONAL_CLIENT_ID NUMBER(6) NOT NULL,  
  EVENT_ID NUMBER(4) NOT NULL,  
  FOREIGN KEY (ORGANIZATIONAL_CLIENT_ID) REFERENCES  
  ORGANIZATIONAL_CLIENT(CUSTOMER_ID), FOREIGN KEY (EVENT_ID) REFERENCES  
  EVENT(EVENT_ID), PRIMARY KEY (ORGANIZATIONAL_CLIENT_ID,EVENT_ID)  
);
```

22.

```
CREATE TABLE DIRECT_BILL_ACCOUNT ( ORGANIZATIONAL_CLIENT_ID NUMBER(6) NOT  
NULL, ACCOUNT_NO NUMBER(10) NOT NULL,  
  FOREIGN KEY (ORGANIZATIONAL_CLIENT_ID) REFERENCES  
  ORGANIZATIONAL_CLIENT(CUSTOMER_ID), PRIMARY KEY  
  (ORGANIZATIONAL_CLIENT_ID,ACCOUNT_NO)  
);
```

23.

```
CREATE TABLE ORGANIZATION_CLIENT_BILL(  
  BILL_ID NUMBER(10) NOT NULL PRIMARY KEY,  
  AMOUNT NUMBER(10) CHECK(AMOUNT>=0),  
  BILL_DATE DATE NOT NULL,  
  ACCOUNTANT_ID NUMBER(9) NOT NULL,  
  EVENT_ID NUMBER(4) NOT NULL,
```

```
FOREIGN KEY (ACCOUNTANT_ID) REFERENCES ACCOUNTANT(EMPLOYEE_ID), FOREIGN KEY  
(EVENT_ID) REFERENCES EVENT(EVENT_ID)  
);
```

24.

```
CREATE TABLE ORGANIZATIONAL_PAYMENT( PAYMENT_ID NUMBER(10) NOT NULL  
PRIMARY KEY, PAYMENT_MODE VARCHAR(20) NOT NULL,  
PAYMENT_AMOUNT NUMBER(6) NOT NULL CHECK(PAYMENT_AMOUNT >= 0),  
PAYMENT_DATE DATE NOT NULL,  
PAYMENT_TIME TIMESTAMP NOT NULL,  
ORG_CUSTOMER_ID NUMBER(6) NOT NULL,  
FOREIGN KEY (ORG_CUSTOMER_ID) REFERENCES ORGANIZATIONAL_CLIENT(CUSTOMER_ID)  
);
```

25.

```
CREATE TABLE PAYMENT_THROUGH_DETAILS (  
PAYMENT_ID NUMBER(10) NOT NULL,  
BILL_ID NUMBER(10) NOT NULL,  
FOREIGN KEY (PAYMENT_ID) REFERENCES ORGANIZATIONAL_PAYMENT(PAYMENT_ID),  
FOREIGN KEY (BILL_ID) REFERENCES ORGANIZATION_CLIENT_BILL(BILL_ID), PRIMARY KEY  
(PAYMENT_ID,BILL_ID)  
);
```

26.

```
CREATE TABLE INDIVIDUAL_PHONE (  
INDIVIDUAL_CUST_ID NUMBER(6) NOT NULL,  
PHONE_NO NUMBER(10) NOT NULL,  
FOREIGN KEY (INDIVIDUAL_CUST_ID) REFERENCES INDIVIDUAL_CLIENT(CUSTOMER_ID),  
PRIMARY KEY (INDIVIDUAL_CUST_ID,PHONE_NO)  
);
```

5.1.2 A Database State

We insert some values into the database in order to test our SQL create view and query statement. Here we just give one example of insertions as follows:

- Insertion in 'State_details' Table-

```
insert into STATE_DETAILS values (75252, 'Richardson', 'Texas');
insert into STATE_DETAILS values (10000, 'Long beach', 'California');
insert into STATE_DETAILS values (10001, 'San Diego', 'California');
insert into STATE_DETAILS values (20001, 'Las Vegas', 'Nevada');
insert into STATE_DETAILS values (50001, 'NewYork City', 'Newyork');
insert into STATE_DETAILS values (60001, 'New Orleans', 'Florida');
insert into STATE_DETAILS values (70001, 'College Station', 'Texas');
insert into STATE_DETAILS values (30001, 'Cincinnati', 'Ohio');
```

State of the Table –

	ZIP	CITY	STATE
1	75252	Richardson	Texas
2	10000	Long beach	California
3	10001	San Diego	California
4	20001	Las Vegas	Nevada
5	50001	NewYork City	Newyork
6	60001	New Orleans	Florida
7	70001	College Station	Texas
8	30001	Cincinnati	Ohio

Following are the other insertion Queries -

```
insert into EMPLOYEE values (000000001, 'Kinjal', 25, 60000, 7600, 'Plano Street', 75252);
insert into EMPLOYEE values (000000002, 'Rahul', 24, 80000, 7900, 'McCallum blvd', 75252);
insert into EMPLOYEE values (000000003, 'Shubham', 26, 75000, 7600, 'Plano Street', 75252);
insert into EMPLOYEE values (000000004, 'Surya', 21, 50000, 7650, 'Ashwood', 75252);
insert into EMPLOYEE values (000000005, 'Piyush', 24, 80000, 7650, 'Ashwood', 75252);
insert into EMPLOYEE values (000000006, 'Rob', 40, 90000, 7000, 'Palencia', 75252);
insert into EMPLOYEE values (000000007, 'Mary', 18, 30000, 8000, 'Long Beach', 10000);
insert into EMPLOYEE values (000000008, 'Mark', 50, 99000, 9000, 'Irvine', 10001);
insert into EMPLOYEE values (000000009, 'John', 45, 80000, 9600, 'Alpine', 10001);
insert into EMPLOYEE values (000000010, 'Dennis', 60, 75000, 9700, 'Fresno', 10000);
insert into EMPLOYEE values (000000011, 'Prashanth', 22, 55000, 9800, 'Riverside', 10001);
```

```
insert into EMPLOYEE values (000000012,'Pandit',29,102000,9900,'San Bernardino',10000);
insert into EMPLOYEE values (000000013,'Yu lin',25,42000,6000,'Albany',50001);
insert into EMPLOYEE values (000000014,'Dhwani',24,86000,6600,'Delaware',50001);
insert into EMPLOYEE values (000000015,'Noumika',26,95000,5500,'Delaware',50001);
insert into RECEPTIONIST values (000000007);
insert into RECEPTIONIST values (000000013);
insert into RECEPTIONIST values (000000002);
insert into HOUSEKEEPING_CONCIERGE values (000000011,10);
insert into HOUSEKEEPING_CONCIERGE values (000000004,2);
insert into EVENT_STAFF values(999999001,0001,1001);
insert into EVENT_STAFF values(999999002,0002,1002);
insert into EVENT_STAFF values(999999003,0003,1003);
insert into EVENT_STAFF values(999999004,0004,1004);
insert into EVENT_STAFF values(999999005,0005,1005);
insert into EVENT_STAFF values(999999006,0006,1006);
insert into EVENT_STAFF values(999999007,0007,1007);
insert into EVENT_STAFF values(999999008,0008,1008);
insert into EVENT_STAFF values(999999009,0009,1009);
insert into TECH_SUPPORT values(000000001,1111111111, 999999001);
insert into TECH_SUPPORT values(000000003,2222222222, 999999002);
insert into TECH_SUPPORT values(000000014,3333333333, 999999003);
insert into ACCOUNTANT values(000000006,4444444444,999999001);
insert into ACCOUNTANT values(000000008,5555555555,999999002);
insert into ACCOUNTANT values(000000015,6666666666,999999003);
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000007,'English');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000007,'French');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000007,'Mexican');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000013,'English');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000013,'Chinese');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000002,'Urdu');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000002,'English');
insert into RECEPTIONIST_LANGUAGES_KNOWN values(000000002,'Hindi');
insert into DINING_STAFF values(000000009,'morning','Fine Dine',999999001);
insert into DINING_STAFF values(000000010,'night','Casual Dining',999999002);
insert into MANAGEMENT values(000000005,'Vice President',999999001);
insert into MANAGEMENT values(000000012,'General Manager',999999002);
insert into EVENT values (9001,TIMESTAMP '2017-01-25 00:00:00 US/Pacific','25-JAN-2017',5000,000000005,000000006);
insert into EVENT values (9002,TIMESTAMP '2017-02-25 00:00:00 US/Pacific','25-FEB-2017',4000,000000005,000000006);
insert into EVENT values (9003,TIMESTAMP '2017-03-25 00:00:00 US/Pacific','25-MAR-2017',4000,000000012,000000008);
insert into EVENT values (9004,TIMESTAMP '2017-04-25 00:00:00 US/Pacific','25-APR-2017',4000,000000012,000000008);
insert into EVENT values (9005,TIMESTAMP '2017-05-25 00:00:00 US/Pacific','25-MAY-2017',4000,000000012,000000015);
insert into EVENT_ORGANIZED_BY values (9001,999999001);
insert into EVENT_ORGANIZED_BY values (9002,999999001);
```

```
insert into EVENT_ORGANIZED_BY values (9003,999999002);
insert into EVENT_ORGANIZED_BY values (9004,999999002);
insert into EVENT_ORGANIZED_BY values (9005,999999002);
insert into CUSTOMER values (100001);
insert into CUSTOMER values (100002);
insert into CUSTOMER values (100003);
insert into CUSTOMER values (100004);
insert into CUSTOMER values (100005);
insert into CUSTOMER values (100006);
insert into INDIVIDUAL_CLIENT values (100001, 'CustomerA','MALE','05-MAY-1993');
insert into INDIVIDUAL_CLIENT values (100002, 'CustomerB','MALE','11-MAR-1993');
insert into INDIVIDUAL_CLIENT values (100003, 'CustomerC','FEMALE','05-JUN-1991');
insert into ORGANIZATIONAL_CLIENT values (100004);
insert into ORGANIZATIONAL_CLIENT values (100005);
insert into ORGANIZATIONAL_CLIENT values (100006);
insert into HOTEL_ROOMS values (1001,'Single','Single',1000,'yes');
insert into HOTEL_ROOMS values (1002,'Single','Suite',1200,'yes');
insert into HOTEL_ROOMS values (1003,'Double','Luxury',2000,'yes');
insert into HOTEL_ROOMS values (1004,'Single','Suite',1200,'yes');
insert into HOTEL_ROOMS values (1005,'Single','Deluxe',1500,'yes');
insert into HOTEL_ROOMS values (1006,'Double','Luxury',2000,'yes');
insert into CUSTOMER_ACTIVITY values (100001,'01-JAN-2017',TIMESTAMP '2017-01-01 00:00:00 US/Pacific','05-JAN-2017',TIMESTAMP '2017-01-05 00:00:00 US/Pacific',1001,'card','yes',5,000000007);
insert into CUSTOMER_ACTIVITY values (100002,'10-JAN-2017',TIMESTAMP '2017-01-10 00:00:00 US/Pacific','15-JAN-2017',TIMESTAMP '2017-01-15 00:00:00 US/Pacific',1002,'digital','no',5,000000013);
insert into CUSTOMER_ACTIVITY values (100003,'20-JAN-2017',TIMESTAMP '2017-01-20 00:00:00 US/Pacific','25-JAN-2017',TIMESTAMP '2017-01-25 00:00:00 US/Pacific',1003,'card','yes',5,000000013);
insert into CUSTOMER_ACTIVITY values (100003,'20-JAN-2017',TIMESTAMP '2017-01-20 00:00:00 US/Pacific','25-JAN-2017',TIMESTAMP '2017-01-25 00:00:00 US/Pacific',1004,'digital','yes',5,000000002);
insert into CUSTOMER_ACTIVITY values (100003,'20-JAN-2017',TIMESTAMP '2017-01-20 00:00:00 US/Pacific','25-JAN-2017',TIMESTAMP '2017-01-25 00:00:00 US/Pacific',1005,'digital','no',5,000000002);
insert into INDIVIDUAL_CUST_BILL_DETAILS values(1000000001,'01-JAN-2017',5000,100001);
insert into INDIVIDUAL_CUST_BILL_DETAILS values(1000000002,'10-JAN-2017',6000,100002);
insert into INDIVIDUAL_CUST_BILL_DETAILS values(1000000003,'20-JAN-2017',10000,100003);
insert into INDIVIDUAL_CUST_BILL_DETAILS values(1000000004,'20-JAN-2017',6000,100003);
insert into INDIVIDUAL_CUST_BILL_DETAILS values(1000000005,'25-JAN-2017',7500,100003);
insert into INDIVIDUAL_PAYMENT values(9000000001,5000,'01-JAN-2017',TIMESTAMP '2017-01-01 12:00:00 US/Pacific',1000000001);
insert into INDIVIDUAL_PAYMENT values(9000000002,6000,'10-JAN-2017',TIMESTAMP '2017-01-10 12:00:00 US/Pacific',1000000002);
insert into INDIVIDUAL_PAYMENT values(9000000003,10000,'20-JAN-2017',TIMESTAMP '2017-01-20 12:00:00 US/Pacific',1000000003);
insert into INDIVIDUAL_PAYMENT values(9000000005,6000,'20-JAN-2017',TIMESTAMP '2017-01-20 12:00:00 US/Pacific',1000000004);
insert into INDIVIDUAL_PAYMENT values(9000000004,7500,'25-JAN-2017',TIMESTAMP '2017-01-25 12:00:00 US/Pacific',1000000005);
insert into INDIVIDUAL_CUST_REWARD values(100001,99999900001);
insert into INDIVIDUAL_CUST_REWARD values(100002,99999900002);
```

```
insert into House_keeping_activity values('06:00:00',1001,000000011);
insert into House_keeping_activity values('07:00:00',1002,000000011);
insert into House_keeping_activity values('08:00:00',1003,000000011);
insert into House_keeping_activity values('06:00:00',1004,000000004);
insert into House_keeping_activity values('07:00:00',1005,000000004);
insert into House_keeping_activity values('08:00:00',1006,000000004);
insert into EVENT_HOST values(100004,9001);
insert into EVENT_HOST values(100005,9002);
insert into EVENT_HOST values(100006,9003);
insert into EVENT_HOST values(100005,9004);
insert into EVENT_HOST values(100006,9005);
insert into DIRECT_BILL_ACCOUNT values(100004,1234567890);
insert into DIRECT_BILL_ACCOUNT values(100005,1234567891);
insert into DIRECT_BILL_ACCOUNT values(100006,1234567892);
insert into ORGANIZATION_CLIENT_BILL values(1111100001,9000,'01-JAN-2017',000000006,9001);
insert into ORGANIZATION_CLIENT_BILL values(1111100002,9000,'01-FEB-2017',000000006,9002);
insert into ORGANIZATION_CLIENT_BILL values(1111100003,9000,'01-MAR-2017',000000008,9003);
insert into ORGANIZATION_CLIENT_BILL values(1111100004,9000,'01-APR-2017',000000008,9004);
insert into ORGANIZATION_CLIENT_BILL values(1111100005,18000,'01-MAY-2017',000000015,9005);
insert into ORGANIZATIONAL_PAYMENT values(3333333301,'card',9000,'01-JAN-2017',TIMESTAMP '2017-01-01 12:00:00 US/Pacific',100004);
insert into ORGANIZATIONAL_PAYMENT values(3333333302,'digital',9000,'01-FEB-2017',TIMESTAMP '2017-02-01 12:00:00 US/Pacific',100005);
insert into ORGANIZATIONAL_PAYMENT values(3333333303,'card',9000,'01-MAR-2017',TIMESTAMP '2017-03-01 12:00:00 US/Pacific',100006);
insert into ORGANIZATIONAL_PAYMENT values(3333333304,'digital',9000,'01-APR-2017',TIMESTAMP '2017-04-01 12:00:00 US/Pacific',100005);
insert into ORGANIZATIONAL_PAYMENT values(3333333305,'card',8000,'01-MAY-2017',TIMESTAMP '2017-05-01 12:00:00 US/Pacific',100006);
insert into ORGANIZATIONAL_PAYMENT values(3333333306,'digital',10000,'01-JUN-2017',TIMESTAMP '2017-06-01 12:00:00 US/Pacific',100006);
insert into PAYMENT_THROUGH_DETAILS values (3333333301,1111100001);
insert into PAYMENT_THROUGH_DETAILS values (3333333302,1111100002);
insert into PAYMENT_THROUGH_DETAILS values (3333333303,1111100003);
insert into PAYMENT_THROUGH_DETAILS values (3333333304,1111100004);
insert into PAYMENT_THROUGH_DETAILS values (3333333305,1111100005);
insert into PAYMENT_THROUGH_DETAILS values (3333333306,1111100005);
insert into INDIVIDUAL_PHONE values(100001,1234567000);
insert into INDIVIDUAL_PHONE values(100002,1234567001);
insert into INDIVIDUAL_PHONE values(100002,1234567002);
insert into INDIVIDUAL_PHONE values(100003,1234567003);
insert into INDIVIDUAL_PHONE values(100003,1234567004);
insert into INDIVIDUAL_PHONE values(100003,1234567005);
```

5.2 Creation of Views

Available Rooms

```
CREATE VIEW AVAILABLE_ROOMS AS (  
  (select Room_no from Hotel_Rooms) minus  
  (Select Room_no from Customer_Activity where Check_in_Date <= Sysdate and (Check_out_Date  
    = null OR Check_Out_Date > SysDate)))
```

Popular Manager

```
CREATE VIEW POPULAR_MANAGER AS (  
  select distinct Manager_ID from  
  (Select EXTRACT(month from Event_date) as Month, Count(Event_ID) as Event_Count,  
    Manager_ID  
  from Event GROUP BY EXTRACT(month from Event_date), Manager_Id) where Event_Count > 10)
```

Popular Customer

```
CREATE VIEW POPULAR_CUSTOMER AS (  
  select Individual_customer_id from(  
    Select Individual_customer_id, Count(Individual_customer_id) as No_of_CheckIn  
  from Customer_activity where EXTRACT(year from Check_In_date) = '2017' group by  
    Individual_Customer_id  
  where No_of_CheckIn >= 10)
```

Popular Rooms

```
Create View Popular_Rooms as  
(select room_no from (  
  Select Room_no, Count(Room_no) as No_of_CheckIn from Customer_Activity where Extract(year  
  from Check_In_date) = '2017' group by room_no)  
  where No_of_CheckIn >= 30)
```

5.3 Creation of SQL Queries

Retrieve the number of employees who work at the lounge/bar.

```
Select COUNT(Employee_ID) from Dining_staff where Dining_Type ='lounge/bar'
```

	1	2

Retrieve the average salary of the receptionists.


```

INDIVIDUAL_CUSTOMER_ID, SUM(LENGTH_OF_STAY) AS TOTAL_NIGHT FROM
CUSTOMER_ACTIVITY
WHERE EXTRACT(year from CHECK_OUT_DATE) ='2017' GROUP BY(INDIVIDUAL_CUSTOMER_ID)
having SUM(LENGTH_OF_STAY)>=15) On Individual_Client.CUSTOMER_ID =
INDIVIDUAL_CUSTOMER_ID

```

	CUSTOMER_ID	CUST_NAME	SEX	DOB	TOTAL_NIGHT
1	100003	CustomerC	FEMALE	05-JUN-91	15

Retrieve the average age of individual customers who were helped by a receptionist who only speaks Spanish.

```

SELECT AVG(TRUNC(months_between(sysdate, DOB) / 12)) AS "Average_Age" FROM
INDIVIDUAL_CLIENT where customer_id in ( SELECT Customer_id FROM INDIVIDUAL_CLIENT
where Customer_Id in ( Select distinct individual_customer_id from Customer_Activity where
receptionist_id in (select Employee_Id from RECEPTIONIST_LANGUAGES_KNOWN where
Language_name = 'Spanish' and Employee_id in (select Employee_id from
RECEPTIONIST_LANGUAGES_KNOWN group by Employee_id having Count(distinct
Language_Name) = 1))))

```

	Average_Age
1	25

Retrieve the information of the organization that organized at least two events and got bills of over \$2000 in total.

```

Select ORGANIZATIONAL_CLIENT_ID, SUM(AMOUNT) from EVENT_HOST Natural Join
ORGANIZATION_CLIENT_BILL GROUP BY ORGANIZATIONAL_CLIENT_ID having
ORGANIZATIONAL_CLIENT_ID in ( Select ORGANIZATIONAL_CLIENT_ID from EVENT_HOST group
by ORGANIZATIONAL_CLIENT_ID having Count(Event_id) >=2)

```

	ORGANIZATIONAL_CLIENT_ID	SUM(AMOUNT)
1	100005	18000
2	100006	27000

Retrieve the highest amount of bill of the events helped by the most popular event manager.

```

Select * from (
Select EVENT_ID, AMOUNT from ORGANIZATION_CLIENT_BILL where EVENT_ID in (
Select Event_Id from EVENT where Manager_ID =(
Select Manager_Id from (select MANAGER_ID, Count(Event_ID) As No_of_Event from Event
Group by MANAGER_ID) JOIN
(Select Max(No_of_Event) AS Max_Event from (select MANAGER_ID, Count(Event_ID) As
No_of_Event from Event Group by MANAGER_ID))
ON No_of_Event = Max_Event)) Order by Amount desc ) where rownum = 1

```

	EVENT_ID	AMOUNT
1	9005	18000

Retrieve information of the event that each of its organizers pays the highest amount for the event (suppose organizers of the same event pay the bill evenly).

```

CREATE VIEW View1 As (
Select Payment_Id, Payment_amount, Average_Payment, Bill_Id, Count_org from
(
Select * from ORGANIZATIONAL_PAYMENT
Natural Join
(Select * from PAYMENT_THROUGH_DETAILS Natural Join
(Select Bill_ID, (Amount/Count_Org) as Average_Payment, Count_org from (
Select * from ORGANIZATION_CLIENT_BILL Natural Join (
Select event_id, Count_org from ( select Event_Id, Count(ORGANIZATIONAL_CLIENT_ID) as
Count_org , Count(Event_ID) as Count_event from event_host group by EVENT_ID)
where Count_Event > 1))))))
)

Select * from Event where Event_Id in (

```

```

Select Event_ID from ORGANIZATION_CLIENT_BILL
Where Bill_Id in (
Select Bill_Id from (
Select * from (
Select Bill_ID, Count(Bill_ID) as Count_Bill from
(Select * from View1 where Payment_amount = Average_Payment) Group By Bill_id)
Natural Join
(Select distinct Bill_Id, Count_org from View1)) where Count_bill = Count_ORG
))

```

	EVE...	EVENT_TIME	EVENT_DATE	DEPOSIT	MANAGER_ID	ACCOUNTANT_ID
1	9006	25-AUG-17 12.00.00.000000000 AM	25-AUG-17	5000	16	8

Retrieve the date and time the most popular room was last checked in

```

Select * from (
Select Room_No, Check_In_date, Check_in_Time from Customer_Activity where Room_No = (
Select Room_no from
(Select Room_No, Count(Room_no) as No_Of_CheckIn from Customer_Activity Group By
Room_No)
JOIN
(Select Max(No_Of_CheckIn) as Max_checkIn from (
Select Room_No, Count(Room_no) as No_Of_CheckIn from Customer_Activity Group By
Room_No))
on No_Of_CheckIn = Max_checkIn ) order by check_in_date desc) where rownum = 1

```

	ROOM_NO	CHECK_IN_DATE	CHECK_IN_TIME
1	1001	10-JAN-17	10-JAN-17 12.00.00.000000000 AM

6. Dependency Diagram

Following Page contains the dependency diagram.

7. Conclusion

In this final report we summarized all the necessary descriptions and solutions for Hotel Management System database, including process and result of EER diagrams, relational schemas in third normal form, SQL statements to create database, create view and solve corresponding queries, as well as dependency diagram. We also implement the whole database in Oracle and using a database state to test every query. We also explained why we use superclass/subclass relationship to build relational schema, why we choose a Relational DBMS to implement our database, and the additional five business rules shown from implementation.

