|  |  |
| --- | --- |
| Project for Database Design—Phase III | Nov 27, 2017 |
|  |  |

Project for Database Design

**Phase III. Implementation**

Anand K Rajagopalan Kartikey Gupta

akr170430@utdallas.edu kxg173430@utdallas.edu

(November 27, 2017)

**0. Pre-Illumination**

For clearly describing the implementation of our database, we separate this report into four sections. In Section 1 we normalize the original relational schema into third normal form and changed part of our relational schema because of some requirement from Phase III. We then explained what are changed. In Section 2 we drew a dependency diagram for each relation table one by one. In Section 3 we began our process of building a database in Oracle using SQL statements, which contains three parts. Part one is the creation of database, including tables, all other structures as well as data type and format, Part two is the creation of views corresponding to five distinct requirements from Question d, and Part three is the creation of Queries to satisfy 14 requirements from Question e. Finally, a short summary is given at the end of this report.

**1. Modified Relational Schema**

Firstly, according to the requirement of phase III and with purpose to simplify the relation model for this database, we changed a couple of things respect to original relational models. We will list them as follows.

1. We normalized the EMPLOYEE table and broke it down further into EMPLOYEE, ZIPCODE\_CITY and CITY\_STATE tables. This was done to convert the EMPLOYEE table in third normal form.
2. We removed the event\_staff\_id foreign key from EVENT table since the information is redundant and helps to convert the EVENT table directly into 3NF.

We have put the modified Relation Schema Diagram in the figure on the last page of the report.

**2. Dependency Diagram**

We now draw a dependency diagram for each table from the Relation schema as follows

EMPLOYEE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Emp\_ID | Name | Age | Salary\_rate | Street\_no | Street\_name | Zip\_code |

ZIPCODE\_CITY

|  |  |
| --- | --- |
| Zip\_code | City |

CITY\_STATE

|  |  |
| --- | --- |
| City | State |

HOUSE\_KEEPING

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

CONCIERGE

|  |  |
| --- | --- |
| Emp\_ID | Yrs\_experience |

RECEPTION

|  |
| --- |
| Emp\_ID |

LICENSED\_EMP

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Licensed\_emp\_type |

MANAGEMENT

|  |  |  |
| --- | --- | --- |
| Emp\_ID | Event\_staff\_ID | Title |

DINING

|  |  |
| --- | --- |
| Emp\_ID | Shift\_type |

LANGUAGES

|  |  |
| --- | --- |
| Emp\_ID | License\_obtained |

LICENSES\_OBTAINED

|  |  |
| --- | --- |
| Emp\_ID | Language |

BUFFET

|  |
| --- |
| Emp\_ID |

FINE\_DINING

|  |
| --- |
| Emp\_ID |

LOUNGE/BAR

|  |
| --- |
| Emp\_ID |

EVENT\_STAFF

|  |  |
| --- | --- |
| Event\_staff\_ID | On-Call\_no |

CATERING

|  |  |
| --- | --- |
| Emp\_ID | Event\_staff\_ID |

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

SERVES

|  |  |
| --- | --- |
| Event\_staff\_ID | Event\_ID |

ROOM

|  |  |  |  |
| --- | --- | --- | --- |
| Room\_no | Price\_per\_night | Room\_type | Bed\_type |

CLEANS

|  |  |  |  |
| --- | --- | --- | --- |
| Housekeeping\_ID | Room\_no | Date | Time |

CHECKIN

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Room\_no | Checkin\_date | Bill\_ID | Receptionist\_ID | Individual\_ID | Check\_out\_date | Lounge\_access | Key\_type | Time |

CLIENT

|  |
| --- |
| Client\_ID |

INDIVIDUAL

|  |  |  |  |
| --- | --- | --- | --- |
| Client\_ID | D\_O\_B | Sex | Name |

PHONE\_NO

|  |  |
| --- | --- |
| Client\_ID | Phone\_no |

MEMBERSHIP\_NO

|  |  |
| --- | --- |
| Client\_ID | MEMBERSHIP\_NO |

ORGANIZATION

|  |
| --- |
| Client\_ID |

BILL

|  |  |  |
| --- | --- | --- |
| Bill\_id | Bill\_amount | Date\_issued |

PAYMENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bill\_ID | Payment\_ID | Client\_ID | date | Time | Amount | Type |

PREPARES

Bill\_ID Accountant\_ID Event\_ID

ACCOUNT

|  |  |  |
| --- | --- | --- |
| Client\_ID | Account\_No | Account\_type |

CONTAINS

Client\_ID Account\_no Bill\_ID

HOLDS

Event\_ID Organizational\_ID Deposit

EVENT

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Manager\_ID | Time | Date |

HOLDS

Event\_ID Organizational\_ID Deposit

**3. Implementation of Database**

**3.1 Creation of Database with SQL Statements**

After normalizing every relational schema into third normal form and modifying some details, it is the time to implement our database using SQL languages into Oracle.

**3.1.1 Table Creation**

Using SQL statement, we created tables as follows:

CITY\_STATE TABLE:

create table CITY\_STATE

(city varchar(30),

state\_ varchar(30),

primary key(city));

ZIPCODE\_CITY TABLE:

create table ZIPCODE\_CITY

(zip\_code INT,

city varchar(30),

primary key(zip\_code),

foreign key(city) references CITY\_STATE(city));

EMPLOYEE TABLE:

create table EMPLOYEE

(Emp\_id INT,

Name varchar(20),

Age INT,

Salary\_rate Float,

Street\_no INT,

Street\_name varchar(30),

zip\_code INT,

primary key(Emp\_id),

foreign key(zip\_code) references ZIPCODE\_CITY(zip\_code));

HOUSE\_KEEPING TABLE:

create table HOUSE\_KEEPING

(Emp\_id INT,

yrs\_experience INT check(yrs\_experience >=0 and yrs\_experience<=50),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

CONCIERGE TABLE:

create table CONCIERGE

(Emp\_id INT,

yrs\_experience INT check(yrs\_experience >=0 and yrs\_experience<=50),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

LICENSED\_EMP TABLE:

create table LICENSED\_EMP

(Emp\_id INT,

Event\_staff\_id INT,

Licensed\_Employee\_type varchar(30) check( Licensed\_Employee\_type in ('ACCOUNTANT', 'TECH\_SUPPORT', 'Accountant', 'Tech\_support')),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

EVENT\_STAFF TABLE:

create table EVENT\_STAFF

(Event\_staff\_id INT,

oncall\_no INT,

primary key(Event\_staff\_id));

RECEPTION TABLE:

create table RECEPTION

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

MANAGEMENT TABLE:

create table MANAGEMENT

(Emp\_id INT,

Event\_staff\_id INT,

Title varchar(20),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

EVENT TABLE:

create table EVENT

(Event\_ID INT,

Manager\_ID INT,

Time\_ TIMESTAMP,

Date\_ DATE default(Sysdate),

primary key(Event\_ID),

foreign key (Manager\_ID) references MANAGEMENT(Emp\_id));

DINING TABLE:

create table DINING

(Emp\_id INT,

Shift\_type char(1) check(Shift\_type in ('m', 'a', 'e', 'n')),

primary key(Emp\_id),

foreign key(Emp\_id) references EMPLOYEE(Emp\_id));

LOUNGE\_BAR TABLE:

create table LOUNGE\_BAR

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

FINE\_DINING TABLE:

create table FINE\_DINING

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

BUFFET TABLE:

create table BUFFET

(Emp\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id));

CATERING TABLE:

create table CATERING

(Emp\_id INT,

Event\_staff\_id INT,

primary key(Emp\_id),

foreign key(Emp\_id) references DINING(Emp\_id),

foreign key(Event\_staff\_id) references EVENT\_STAFF(Event\_staff\_id));

SERVES TABLE:

create table SERVES

(Event\_staff\_id INT,

Event\_ID INT,

primary key(Event\_staff\_id,Event\_ID),

foreign key(Event\_staff\_id) references EVENT\_STAFF,

foreign key(Event\_ID) references EVENT(Event\_ID));

LANGUAGES TABLE:

create table LANGUAGES

(Emp\_id INT,

language varchar(20),

primary key(Emp\_id, language),

foreign key(Emp\_id) references RECEPTION(Emp\_ID));

LICENSES\_OBTAINED TABLE:

create table LICENSES\_OBTAINED

(Emp\_id INT,

License\_obtained varchar(20),

primary key (Emp\_id, License\_obtained),

foreign key(Emp\_id) references LICENSED\_EMP(Emp\_id));

ROOM TABLE:

create table ROOM

(Room\_no INT,

price\_per\_night float,

Room\_type varchar(20),

Bed\_type varchar(20),

primary key(Room\_no));

CLEANS TABLE:

create table CLEANS

(Housekeeping\_id INT,

Room\_no INT,

date\_ DATE default(sysdate),

time\_ Timestamp,

Primary key(Housekeeping\_id, Room\_no),

foreign key(Housekeeping\_id) references HOUSE\_KEEPING(Emp\_id),

foreign key(Room\_no) references ROOM(Room\_no));

CLIENT TABLE:

create table CLIENT

(client\_id INT,

primary key(client\_id));

BILL TABLE:

create table BILL

(Bill\_id INT,

Bill\_amount float,

Date\_issued Date,

primary key(Bill\_id));

PREPARES TABLE:

create table PREPARES

(Bill\_id INT,

Account\_id INT,

Event\_id INT,

primary key(Bill\_id),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(Account\_id) references LICENSED\_EMP(Emp\_id),

foreign key(Event\_id) references EVENT(Event\_id));

ORGANIZATION TABLE:

create table ORGANIZATION

(client\_id INT,

primary key(client\_id),

foreign key(client\_id) references CLIENT(client\_id));

ACCOUNT TABLE:

create table ACCOUNT

(client\_id INT,

Account\_no INT,

Account\_type varchar(20),

primary key(client\_id, Account\_no),

foreign key(client\_id) references ORGANIZATION(client\_id));

HOLDS TABLE:

create table HOLDS

(Event\_id INT,

Organization\_id INT,

Deposit float,

primary key(Event\_id, Organization\_id),

Foreign key(Organization\_id) references ORGANIZATION(client\_id),

foreign key(Event\_id) references EVENT(Event\_id));

CONTAINS TABLE:

create table CONTAINS

(client\_id INT,

Account\_no INT,

Bill\_id INT,

primary key(client\_id, Account\_no, Bill\_id),

foreign key(client\_id, Account\_no) references ACCOUNT(client\_id, Account\_no),

foreign key(Bill\_id) references BILL(Bill\_id));

PAYMENT TABLE:

create table PAYMENT

(bill\_id INT,

Payment\_id INT,

client\_id INT,

date\_ Date,

time\_ Timestamp,

amount float,

type varchar(20),

primary key(Bill\_id, Payment\_id),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(client\_id) references CLIENT(client\_id));

INDIVIDUAL TABLE:

create table INDIVIDUAL

(client\_id INT,

D\_O\_B date,

sex varchar(20),

name varchar(20),

primary key(client\_id),

foreign key(client\_id) references CLIENT(client\_id));

CHECKIN TABLE:

create table CHECKIN

(Room\_no INT,

checkin\_date DATE,

Bill\_id INT,

Receptionist\_id INT,

Individual\_id INT,

checkout\_date DATE,

lounge\_access INT,

Key\_type varchar(10) check(Key\_type in('CARD','DIGITAL')),

Time\_ Timestamp,

primary key(Room\_no, checkin\_date),

foreign key(Room\_no) references ROOM(Room\_no),

foreign key(Bill\_id) references BILL(Bill\_id),

foreign key(Receptionist\_id) references RECEPTION(Emp\_id),

foreign key(Individual\_id) references INDIVIDUAL(client\_id));

PHONE\_NO TABLE:

create table PHONE\_NO

(client\_id INT,

Phone\_no INT,

primary key(client\_id, Phone\_no),

foreign key(client\_id) references CLIENT(client\_id));

MEMBERSHIP\_NO TABLE:

create table MEMBERSHIP\_NO

(client\_id INT,

Membership\_no INT,

primary key(client\_id, Membership\_no),

foreign key(client\_id) references CLIENT(client\_id));

**3.2 Creation of Views**

**3.2.1 Available rooms**

CREATE VIEW AVAILABLE\_ROOMS

AS (SELECT R.Room\_no

FROM ROOM R)

MINUS

(SELECT R1.Room\_no

FROM ROOM R1, CHECKIN C

WHERE R1.Room\_no = C.Room\_no AND C.Checkin\_date < SYSDATE AND C.Checkout\_date > SYSDATE);

**3.2.2 Popular event manager**

CREATE VIEW POPULAR\_EVENT\_MANAGER

AS SELECT EMP.\*, E.N\_events

FROM EMPLOYEE EMP,

(SELECT EVE.Manager\_id AS Manager\_id, COUNT(\*) AS N\_events

FROM EVENT EVE

GROUP BY EVE.Manager\_id, EXTRACT(month FROM EVE.DATE\_)) E

WHERE E.Manager\_id = EMP.Emp\_id AND E.N\_events > 10;

**3.2.3 Frequent customers**

CREATE VIEW FREQUENT\_CUSTOMERS

AS SELECT IND.\*, I.N\_checkin

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, COUNT(\*) AS N\_checkin

FROM CHECKIN C

WHERE EXTRACT(year FROM C.TIME\_) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.N\_checkin >= 10;

**3.2.4 Popular rooms**

CREATE VIEW POPULAR\_ROOMS

AS SELECT R.\*, R1.N\_checkin

FROM ROOM R,

(SELECT C.Room\_no AS Room\_no, COUNT(\*) AS N\_checkin

FROM CHECKIN C

WHERE EXTRACT(year FROM C.CHECKIN\_DATE) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Room\_no) R1

WHERE R.Room\_no = R1.Room\_no AND R1.N\_checkin >= 30;

**3.3 Creation of SQL Queries**

**3.3.1 Retrieve the number of employees who work at the lounge/bar**

SELECT COUNT(\*) AS NO\_OF\_EMPLOYEES FROM LOUNGE\_BAR;

**3.3.2 Retrieve the average salary of the receptionists**

SELECT AVG(E.Salary\_rate) AS AVERAGE\_SALARY

FROM EMPLOYEE E, RECEPTION R

WHERE E.Emp\_id = R.Emp\_id;

**3.3.3 Retrieve the information of individual customers who have been billed more than $1,000 in total this year**

SELECT IND.\*

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, SUM(B.Bill\_amount) AS Total\_bill\_amount

FROM CHECKIN C, BILL B

WHERE C.Bill\_id = B.Bill\_id

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.Total\_bill\_amount > 1000;

**3.3.4 For each individual, retrieve his/her bill amount in ascending order of each check-in date**

SELECT IND.\*, I.Bill\_amount, I.Checkin\_date

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, C.Checkin\_date AS Checkin\_date, B.Bill\_amount AS Bill\_amount

FROM CHECKIN C, BILL B) I

WHERE IND.Client\_id = I.Individual\_id

ORDER BY I.Checkin\_date;

**3.3.5 Retrieve the information of the frequent customers who have stayed for at least 15 nights this year**

SELECT IND.\*

FROM INDIVIDUAL IND,

(SELECT C.Individual\_id AS Individual\_id, SUM(C.Checkout\_date - C.Checkin\_date) AS N\_days\_of\_stay

FROM CHECKIN C

WHERE EXTRACT(year FROM C.Checkin\_date) = EXTRACT(year FROM SYSDATE)

GROUP BY C.Individual\_id) I

WHERE IND.Client\_id = I.Individual\_id AND I.N\_days\_of\_stay >= 15;

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

SELECT AVG((SYSDATE - I.d\_o\_b)/365.242199) AS AVERAGE\_AGE

FROM INDIVIDUAL I

WHERE I.Client\_id IN

(SELECT C.Individual\_id

FROM CHECKIN C,

(SELECT L1.Emp\_id AS Emp\_id

FROM LANGUAGES L1

MINUS

SELECT L2.Emp\_id AS Emp\_id

FROM LANGUAGES L2

WHERE L2.Language != 'Spanish') R

WHERE C.Receptionist\_id = R.Emp\_id);

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

SELECT ORG1.Client\_id

FROM (SELECT ORG3.Organization\_id AS Client\_id

FROM (SELECT H.Organization\_id, COUNT(\*) AS Count\_events

FROM HOLDS H

GROUP BY H.Organization\_id) ORG3

WHERE ORG3.Count\_events >= 2) ORG1,

(SELECT ORG4.Client\_id AS Client\_id

FROM (SELECT C.Client\_id, SUM(B.Bill\_amount) AS Total\_bill\_amount

FROM CONTAINS C, BILL B

WHERE B.Bill\_id = C.Bill\_id

GROUP BY C.Client\_id) ORG4

WHERE ORG4.Total\_bill\_amount > 2000) ORG2

WHERE ORG1.Client\_id = ORG2.Client\_id;

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

SELECT MAX(B.Bill\_amount) AS MAXIMUM\_BILL\_AMOUNT

FROM EVENT E, BILL B, PREPARES P,

(SELECT PM.\*

FROM (SELECT \*

FROM POPULAR\_EVENT\_MANAGER

ORDER BY N\_events DESC) PM

WHERE rownum = 1) MPM

WHERE MPM.Emp\_id = E.Manager\_id AND P.Event\_id = E.Event\_id

AND P.Bill\_id = B.Bill\_id;

**3.3.9 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)**

SELECT E.\*

FROM EVENT E, PREPARES P,

(SELECT O.Bill\_id AS BIll\_id

FROM (SELECT Bill\_id, Client\_id

FROM PAYMENT

GROUP BY (PAYMENT.Bill\_id, PAYMENT.Client\_id)

ORDER BY SUM(PAYMENT.Amount) DESC) O

WHERE rownum = 1) PAY

WHERE E.Event\_id = P.Event\_id AND P.Bill\_id = PAY.Bill\_id;

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

SELECT MAX(C.TIME\_) AS Last\_checkin

FROM CHECKIN C,

(SELECT PR.\*

FROM (SELECT \*

FROM POPULAR\_ROOMS

ORDER BY N\_CHECKIN DESC) PR

WHERE rownum = 1) R

WHERE C.Room\_no = R.Room\_no;

**4. Conclusion**

In this report we modified the EER diagram and relational schemas for ABC Hotel Database according to the requirement of Phase III. We also give dependency diagram for each relational schema in database. Then we created tables for each relational schema and write the SQL statements for the views and queries listed in Question d and Question e.