**Database Project – Final Report**

# First Name: Abhinav Last Name: Verma

Database Management Systems - Professor Fernandez

George Washington University

Month 04, 2016

# **Part 1**

## Overview

This project focuses on the creation of a database from the conceptual model design to a fully functional database including primary and foreign key constraints, indices to improve query performance, and triggers for history tracking and auditing. The database will be developed using the Oracle SQL Modeler and SQL Developer Applications. This database will store and manage projects and tasks for a catering company. I will refer to this organization as the Classic Catering People (CCP) although this is not the organization’s real name. A database will be created to store and manage CCP’s events, contractors and clients. Information describing events must be stored along with the contractors for the particular event. Details of different contractors like florist, musicians, entertainers and photographers are added to database. Client details are also added and the location details that the client chose for a particular event along with the order details. The proposed database will assist management on future event handling.

## Purpose and Objectives

Classic Catering People (CCP) is a catering company and requires a database for its operation. There are clients who can be individuals or business organizations. They can book events with CCP on their hall or any other location. In the CCP hall one event can be booked at a time. Any number of events can take place at offsite locations. Minimum 150 guests need to be there for hall to be booked which can handle 200 maximum guests. Offsite locations can host any number of guests. The hall can be booked two years in advance and offsite events can be booked months in advance. The client can choose menu for the event from menus provided by CCP. Client can also choose florist, musicians, entertainers and photographers for the event. CCP charges 10% extra of the cost to the client, that the contractors charge the CCP. Total price depends on the time of year, location, number of guests, contractors hired etc. Client can choose the color of linen, number of waiters bartenders etc too.

The solution provided in appendix is quite effective. But I feel the addition of client Id would be better as the name could be same for two people. Also the addition of event ID would help in creating an effective database.

**Data Modeling Tools**

Oracle SQL Developer Data Modeler will be used to design the Data Flow Diagram, Entity Relationship Diagram, and Relational Model.

**Database Software**

The database will be based on Oracle 11g software.

Oracle SQL Developer will be used to access the database although other Developer and Query Tools will also be compatible including SQL Plus, SQL Navigator and other SQL and PL/SQL developer tools.

**Additional Hardware and Software Details**

The CCP project database will be hosted on a server running the Linux operating system located at the George Washington University. Oracle SQL Developer will be used to access the database with support for other Developer and Query Tools including Oracle SQL Plus and third-party tools including SQL Navigator.

# **Part 2**

**Brief Project Goal**

A database will be created for the Atlantic Classing Catering People (CCP) to store and manage CCP’s short-term and long-term projects, tasks, and staff assignments.

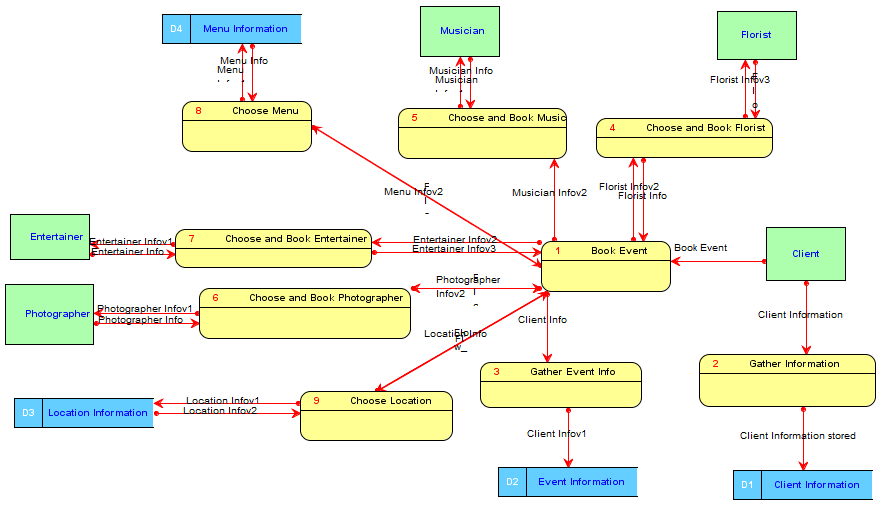
**Business Direction Information**

Assumption: CCP would like the new event management database to be developed using Oracle Database version 11g.

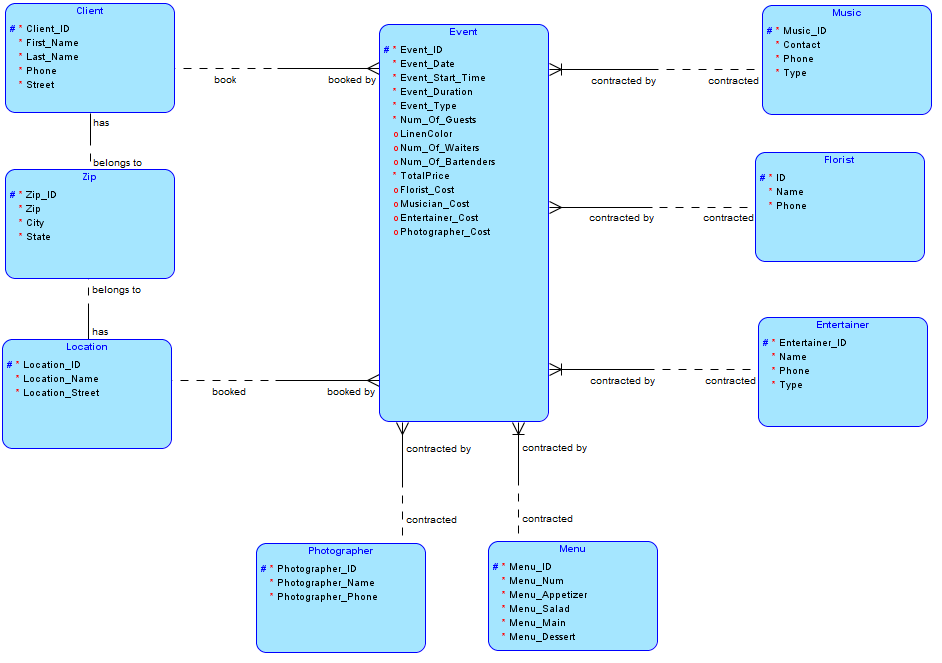
Problem: CCP does not have a database to record all its events and contractors data. As the business is growing it is difficult maintain all the data manually. The company also wants to create a website for the business for which database is required.

Business Objective: To improve the maintenance of records.

## Data Flow Diagram (DFD)

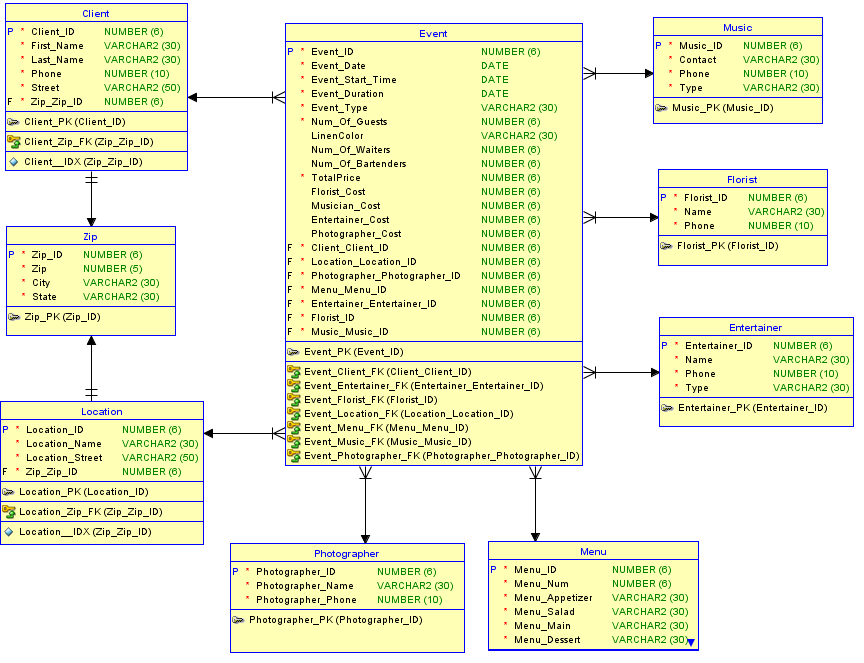


## Entity-Relation Diagram (ERD)



# **Part 3**

## Relational Database Design



## Dependency Diagram

To show high level of relationships among the attributes of the table dependency diagram is created. In the below dependency diagram, primary key is shown with highlighted box. Arrows which are drawn above the attributes shows that there is desirable dependency between attributes.

**Event Table:**

Event\_Date

 Event\_Duration

Client\_client\_id

Event\_start\_time

Entertainer\_Cost

Musician\_Cost

Florist\_Cost

TotalPrice

Num\_Of\_Bartenders

LinenColor

 Num\_Of\_Guests

 Event\_Type

Num\_Of\_Waiters

Photographer\_Cost

Music\_Music\_ID

Florist\_ID

Entertainer\_Entertainer\_ID

Menu\_Menu\_ID

Photographer\_Photographer\_ID

Location\_Location\_ID

**Client Table:**

First\_Name

Last\_Name

Zip\_Zip\_ID

Street

Phone

**Entertainer Table:**

Name

Phone

Type

**Florist Table:**

Name

phone

**Location Table:**

Location\_Name

Location\_street

Zip\_Zip\_ID

**Menu Table:**

Meny\_Num

    Menu\_Dessert

Menu\_Main

Menu\_Salad

Menu\_Appetizer

**Music Table:**

Contact

Type

Phone

**Photographer Table:**

Photographer\_name

Photographer\_Phone

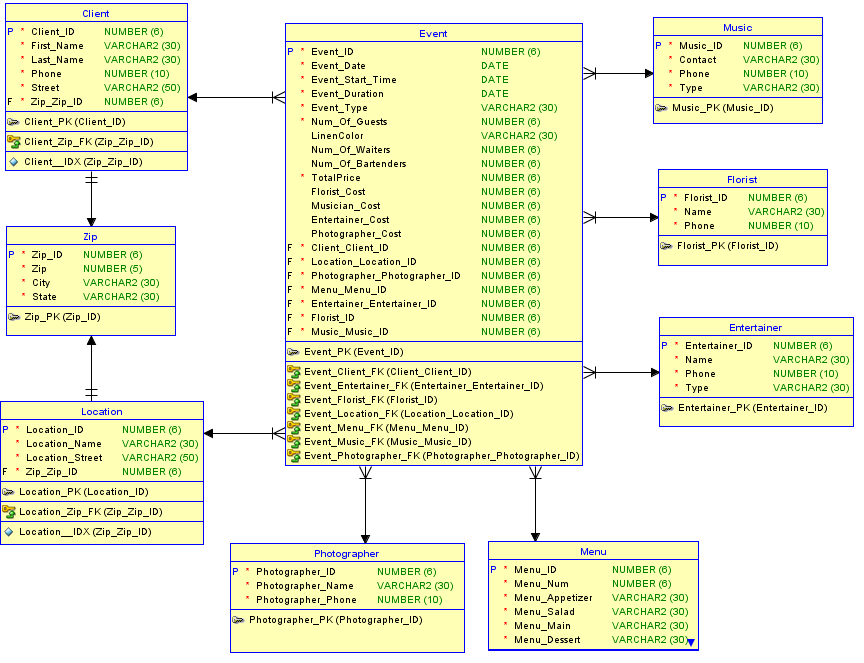
**Zip Table:**

Zip

State

City

## Comparison of Relational Diagrams



## Physical Database Design

|  |
| --- |
| CREATE TABLE Client  (  Client\_ID NUMBER (6) NOT NULL ,  First\_Name VARCHAR2 (30) NOT NULL ,  Last\_Name VARCHAR2 (30) NOT NULL ,  Phone NUMBER (10) NOT NULL ,  Street VARCHAR2 (50) NOT NULL ,  Zip\_Zip\_ID NUMBER (6) NOT NULL  ) ;  CREATE UNIQUE INDEX Client\_\_IDX ON Client  (  Zip\_Zip\_ID ASC  )  ;  ALTER TABLE Client ADD CONSTRAINT Client\_PK PRIMARY KEY ( Client\_ID ) ;  CREATE TABLE Entertainer  (  Entertainer\_ID NUMBER (6) NOT NULL ,  Name VARCHAR2 (30) NOT NULL ,  Phone NUMBER (10) NOT NULL ,  Type VARCHAR2 (30) NOT NULL  ) ;  ALTER TABLE Entertainer ADD CONSTRAINT Entertainer\_PK PRIMARY KEY ( Entertainer\_ID ) ;  CREATE TABLE Event  (  Event\_ID NUMBER (6) NOT NULL ,  Event\_Date DATE NOT NULL ,  Event\_Start\_Time DATE NOT NULL ,  Event\_Duration DATE NOT NULL ,  Event\_Type VARCHAR2 (30) NOT NULL ,  Num\_Of\_Guests NUMBER (6) NOT NULL ,  LinenColor VARCHAR2 (30) ,  Num\_Of\_Waiters NUMBER (6) ,  Num\_Of\_Bartenders NUMBER (6) ,  TotalPrice NUMBER (6) NOT NULL ,  Florist\_Cost NUMBER (6) ,  Musician\_Cost NUMBER (6) ,  Entertainer\_Cost NUMBER (6) ,  Photographer\_Cost NUMBER (6) ,  Client\_Client\_ID NUMBER (6) NOT NULL ,  Location\_Location\_ID NUMBER (6) NOT NULL ,  Photographer\_Photographer\_ID NUMBER (6) NOT NULL ,  Menu\_Menu\_ID NUMBER (6) NOT NULL ,  Entertainer\_Entertainer\_ID NUMBER (6) NOT NULL ,  Florist\_ID NUMBER (6) NOT NULL ,  Music\_Music\_ID NUMBER (6) NOT NULL  ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_PK PRIMARY KEY ( Event\_ID ) ;  CREATE TABLE Florist  (  Florist\_ID NUMBER (6) NOT NULL ,  Name VARCHAR2 (30) NOT NULL ,  Phone NUMBER (10) NOT NULL  ) ;  ALTER TABLE Florist ADD CONSTRAINT Florist\_PK PRIMARY KEY ( Florist\_ID ) ;  CREATE TABLE Location  (  Location\_ID NUMBER (6) NOT NULL ,  Location\_Name VARCHAR2 (30) NOT NULL ,  Location\_Street VARCHAR2 (50) NOT NULL ,  Zip\_Zip\_ID NUMBER (6) NOT NULL  ) ;  CREATE UNIQUE INDEX Location\_\_IDX ON Location  (  Zip\_Zip\_ID ASC  )  ;  ALTER TABLE Location ADD CONSTRAINT Location\_PK PRIMARY KEY ( Location\_ID ) ;  CREATE TABLE Menu  (  Menu\_ID NUMBER (6) NOT NULL ,  Menu\_Num NUMBER (6) NOT NULL ,  Menu\_Appetizer VARCHAR2 (30) NOT NULL ,  Menu\_Salad VARCHAR2 (30) NOT NULL ,  Menu\_Main VARCHAR2 (30) NOT NULL ,  Menu\_Dessert VARCHAR2 (30) NOT NULL  ) ;  ALTER TABLE Menu ADD CONSTRAINT Menu\_PK PRIMARY KEY ( Menu\_ID ) ;  CREATE TABLE Music  (  Music\_ID NUMBER (6) NOT NULL ,  Contact VARCHAR2 (30) NOT NULL ,  Phone NUMBER (10) NOT NULL ,  Type VARCHAR2 (30) NOT NULL  ) ;  ALTER TABLE Music ADD CONSTRAINT Music\_PK PRIMARY KEY ( Music\_ID ) ;  CREATE TABLE Photographer  (  Photographer\_ID NUMBER (6) NOT NULL ,  Photographer\_Name VARCHAR2 (30) NOT NULL ,  Photographer\_Phone NUMBER (10) NOT NULL  ) ;  ALTER TABLE Photographer ADD CONSTRAINT Photographer\_PK PRIMARY KEY ( Photographer\_ID ) ;  CREATE TABLE Zip  (  Zip\_ID NUMBER (6) NOT NULL ,  Zip NUMBER (5) NOT NULL ,  City VARCHAR2 (30) NOT NULL ,  State VARCHAR2 (30) NOT NULL  ) ;  ALTER TABLE Zip ADD CONSTRAINT Zip\_PK PRIMARY KEY ( Zip\_ID ) ;  ALTER TABLE Client ADD CONSTRAINT Client\_Zip\_FK FOREIGN KEY ( Zip\_Zip\_ID ) REFERENCES Zip ( Zip\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Client\_FK FOREIGN KEY ( Client\_Client\_ID ) REFERENCES Client ( Client\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Entertainer\_FK FOREIGN KEY ( Entertainer\_Entertainer\_ID ) REFERENCES Entertainer ( Entertainer\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Florist\_FK FOREIGN KEY ( Florist\_ID ) REFERENCES Florist ( Florist\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Location\_FK FOREIGN KEY ( Location\_Location\_ID ) REFERENCES Location ( Location\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Menu\_FK FOREIGN KEY ( Menu\_Menu\_ID ) REFERENCES Menu ( Menu\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Music\_FK FOREIGN KEY ( Music\_Music\_ID ) REFERENCES Music ( Music\_ID ) ;  ALTER TABLE Event ADD CONSTRAINT Event\_Photographer\_FK FOREIGN KEY ( Photographer\_Photographer\_ID ) REFERENCES Photographer ( Photographer\_ID ) ;  ALTER TABLE Location ADD CONSTRAINT Location\_Zip\_FK FOREIGN KEY ( Zip\_Zip\_ID ) REFERENCES Zip ( Zip\_ID ) ; |

## Non – Unique Indexes for foreign Keys

|  |
| --- |
| CREATE INDEX ZipID\_Loc\_IND ON Location (Zip\_Zip\_ID);  CREATE INDEX ZipID\_Cli\_IND ON Client (Zip\_Zip\_ID);  CREATE INDEX ZipID\_IND ON Client (Client\_Client\_ID);  CREATE INDEX LocID\_IND ON Client (Location\_Location\_ID);  CREATE INDEX PhotoID\_IND ON Client (Photographer\_photographer\_ID);  CREATE INDEX Men\_IND ON Client (Menu\_Menu\_ID);  CREATE INDEX Entr\_IND ON Client (Entertainer\_Entertainer\_ID);  CREATE INDEX Flor\_IND ON Client (Florist\_ID);  CREATE INDEX Musi\_IND ON Client (Music\_Music\_ID); |

## Functional Dependencies of the tables created

**Client Table**

{Last\_Name, First\_Name} 🡪 {Phone, Street, Zip}

We created a surrogate key: Client\_ID

**Event Table**

{client\_ID, event\_Date, event\_Start\_Time} 🡪 {event\_Duration, event\_Type, num\_Of\_Guests, location\_ID, linenColor, num\_Of\_Waiters, num\_Of\_Bartenders, totalPrice, florist\_ID, florist\_Cost, music\_ID, musician\_Cost, entertainer\_ID, entertainer\_Cost, photographer\_ID, photographer\_Cost, menu\_ID}

We created a surrogate key: Event\_ID

**Menu Table**

{menuNumber} 🡪 {menuAppetizer, menuSalad, menuMain, menuDessert}

We created a surrogate key: Menu\_ID

**Location Table**

{locationName} 🡪 {locationStreet, locationZip}

We created a surrogate key: Location\_ID

**Zip Table**

{Zip} 🡪 {City, State}

We created a surrogate key: Zip\_ID

**Florist Table**

{floristName} 🡪 {floristPhone}

We created a surrogate key: Florist\_ID

**Music Table**

{musicContact} 🡪 {musicContactPhone, musicType}

We created a surrogate key: Music\_ID

**Entertainer Table**

{entertainerName} 🡪 {entertainerPhone, entertainerType}

We created a surrogate key: Entertainer\_ID

**Photographer Table**

{photographerName} 🡪 {photographerPhone}

We created a surrogate key: Photographer\_ID

## Lessons Learnt

# **Part 4**

## Unique constraints for Natural and alternate keys

|  |
| --- |
| ALTER TABLE Event ADD CONSTRAINT event\_key UNIQUE (Client\_Client\_ID, Event\_Date, Event\_Start\_Time);  ALTER TABLE Client ADD CONSTRAINT client\_name UNIQUE (First\_Name, Last\_Name);  ALTER TABLE Zip ADD CONSTRAINT unique\_zip UNIQUE (Zip);  ALTER TABLE Location ADD CONSTRAINT unique\_location UNIQUE (Location\_Name);  ALTER TABLE Photographer ADD CONSTRAINT unique\_photographer\_name UNIQUE (Photographer\_Name);  ALTER TABLE Menu ADD CONSTRAINT unique\_menu\_num UNIQUE (Menu\_Num);  ALTER TABLE Entertainer ADD CONSTRAINT unique\_entertainer\_name UNIQUE (Name);  ALTER TABLE Florist ADD CONSTRAINT unique\_florist\_name UNIQUE (Name);  ALTER TABLE Music ADD CONSTRAINT unique\_music\_contact UNIQUE (Contact); |

## Sequences

|  |
| --- |
| CREATE SEQUENCE ClientSeq  START WITH 1  INCREMENT BY 1 ;    CREATE SEQUENCE ZIPSequence  START WITH 1  INCREMENT BY 1 ;  CREATE SEQUENCE LocationSeq  START WITH 1  INCREMENT BY 1 ;      CREATE SEQUENCE FloristSeq  START WITH 1  INCREMENT BY 1 ;    CREATE SEQUENCE PhotoGrapherSeq  START WITH 1  INCREMENT BY 1 ;  CREATE SEQUENCE EntSeq  START WITH 1  INCREMENT BY 1 ;    CREATE SEQUENCE MusicSeq  START WITH 1  INCREMENT BY 1 ;  CREATE SEQUENCE MenuSeq  START WITH 1  INCREMENT BY 1 ;  Commit;  CREATE SEQUENCE EventSeq  START WITH 1  INCREMENT BY 1 ;  Commit; |

## Insert Database Queries

|  |
| --- |
| Insert into ZIP values (ZIPSequence.NextVal,'20004', 'Washington','District of Columbia');  Insert into ZIP values (ZIPSequence.NextVal,'20005', 'Washington','District of Columbia');  Insert into ZIP values (ZIPSequence.NextVal,'10018', 'New York','New York');  Insert into CLIENT values (ClientSeq.NextVaL,'Client1','CF1', [2027423499](tel:2027423499),'1331 Pennsylvania Ave NW', 1);  Insert into CLIENT values (ClientSeq.NextVaL,'Bank of America','CF2',[2027423499](tel:2027423499),'700 13th St NW',2);  Insert into CLIENT values (ClientSeq.NextVaL,'The New York Times','CF3',[9176984637](tel:9176984637),'620 8th Ave',3);  Insert into LOCATION values (LocationSeq.NextVaL,'BOFA','700 13 NW', 2);  Insert into LOCATION values (LocationSeq.NextVaL,'New York Times Building','620 8th Ave', 3);  Insert into LOCATION values (LocationSeq.NextVaL,'JW Marriott Washington, DC','1331 Pennsylvania Ave NW', 1);  Insert into FLORIST VALUES (FloristSeq.NextVal,'Caruso Florist',[2022233816](tel:2022233816));  Insert into FLORIST VALUES (FloristSeq.NextVal,'Nosegay Flowers',[2023381146](tel:2023381146));  Insert into FLORIST VALUES (FloristSeq.NextVal,'Flowers On Fourteenth',[2026677888](tel:2026677888));  Insert into PHOTOGRAPHER VALUES (PhotoGrapherSeq.NextVal,'DuHon Photography',[2026414889](tel:2026414889));  Insert into PHOTOGRAPHER VALUES (PhotoGrapherSeq.NextVal,'Sam Hurd Photography',[4344265018](tel:4344265018));  Insert into PHOTOGRAPHER VALUES (PhotoGrapherSeq.NextVal,'Moshe Zusman Photography',[2025067218](tel:2025067218));  Insert into ENTERTAINER VALUES (EntSeq.NextVal,'Capitol Steps',[2023121555](tel:2023121555),'Agency');  Insert into ENTERTAINER VALUES (EntSeq.NextVal, 'The Walt Disney Company',[2022224810](tel:2022224810),'Agency');  Insert into ENTERTAINER VALUES (EntSeq.NextVal, 'Aloha Luau Entertainment',[2022438484](tel:2022438484),'Agency');  Insert into MUSIC VALUES (MusicSeq.NextVal, 'Blue House Productions',[3015891001](tel:3015891001),'Company');  Insert into MUSIC VALUES (MusicSeq.NextVal, 'Alexandria Music Company',[7036606025](tel:7036606025),'Company');  Insert into MENU VALUES (MenuSeq.NextVal,1,'Parmesian Garlic Cheese Bread','Chicken Salad','Lasagna','Lemon Meringue Cheesecake');  Insert into MENU VALUES (MenuSeq.NextVal,2,'Southern Fried Chicken Sliders','Chicken Salad','Lasagna','Lemon Meringue Cheesecake');  Insert into MENU VALUES (MenuSeq.NextVal,3,'Factory Nachos','Chicken Salad','Lasagna','Lemon Meringue Cheesecake');  Insert into MENU VALUES (MenuSeq.NextVal,4,'Factory Nachos','Chicken Salad','Lasagna','Salted Caramel Cheesecake');  Insert into MENU VALUES (MenuSeq.NextVal,5,'Southern Fried Chicken Sliders','Chicken Salad','Lasagna','Lemon Meringue Cheesecake');  Commit;  Insert into EVENT VALUES (EventSeq.NextVal,  TO\_DATE('2016/05/03 21:02:44', 'yyyy/mm/dd hh24:mi:ss'),  TO\_DATE('2016/05/03 21:02:44', 'yyyy/mm/dd hh24:mi:ss'),  TO\_DATE('2016/05/03 21:02:44', 'yyyy/mm/dd hh24:mi:ss'),  'Conference',  50,  'Off White',  20,  5,  30000,  500,  2000,  5000,  500,  11,1,1,2,1,1,1);  Commit; |

## Select Queries

|  |
| --- |
| /\* to get the photographer name for known event id 2\*/  select p.Photographer\_name from photographer p, event e where e.event\_id = 1 and p.photographer\_id = e.photographer\_photographer\_id;  /\* to find out the menu main dish for an event given event id \*/  select m.menu\_main from event e, menu m where e.menu\_menu\_id = m.menu\_id and e.event\_id = 1;  /\* to find out client phone number for the event id 1 \*/  select c.phone from event e, client c where e.client\_client\_id = c.client\_id and e.event\_id = 1;  /\* to check the total sum of different type of events \*/  select e.event\_type, sum (totalprice) from event e group by e.event\_type;  /\* to calculate the count of number of events by event type \*/  select e.EVENT\_TYPE ,count(e.EVENT\_ID) from event e group by e.event\_type having count(e.EVENT\_TYPE) > 1;  /\* to check total number of guests attended by event type \*/  select event\_type, sum(NUM\_OF\_GUESTS) from event group by event\_type;  /\* to check the location (city) of the client \*/  select c.FIRST\_NAME, c.LAST\_NAME, z.CITY from client c, zip z where c.zip\_zip\_id = z.zip\_id;  /\* to find all the clients in the state of New York \*/  select c.FIRST\_NAME, c.LAST\_NAME from client c where c.zip\_zip\_id in (select zip\_id from zip z where state = 'New York');  /\* to find out the most favorite dessert \*/  select e.event\_type from event e where e.menu\_menu\_id in (select menu\_id from menu where menu\_dessert = 'Salted Caramel Cheesecake');  /\* searching all the events in state of DC \*/  select e.EVENT\_TYPE from event e where e.LOCATION\_LOCATION\_ID in (select l.LOCATION\_ID from location l join zip z on l.ZIP\_ZIP\_ID = z.ZIP\_ID where z.STATE = 'New York'); |

1. **Normalization:**

Functional Dependencies are as follows:

|  |
| --- |
| **Client Table**  {Last\_Name, First\_Name} 🡪 {Phone, Street, Zip}  We created a surrogate key: Client\_ID  **Event Table**  {client\_ID, event\_Date, event\_Start\_Time} 🡪 {event\_Duration, event\_Type, num\_Of\_Guests, location\_ID, linenColor, num\_Of\_Waiters, num\_Of\_Bartenders, totalPrice, florist\_ID, florist\_Cost, music\_ID, musician\_Cost, entertainer\_ID, entertainer\_Cost, photographer\_ID, photographer\_Cost, menu\_ID}  We created a surrogate key: Event\_ID  **Menu Table**  {menu\_Num} 🡪 {menu\_Appetizer, menu\_Salad, menu\_Main, menu\_Dessert}  We created a surrogate key: Menu\_ID  **Location Table**  {location\_Name} 🡪 {location\_Street, Zip}  We created a surrogate key: Location\_ID  **Zip Table**  {Zip} 🡪 {City, State}  We created a surrogate key: Zip\_ID  **Florist Table**  {Name} 🡪 {Phone}  We created a surrogate key: Florist\_ID  **Music Table**  {Contact} 🡪 {Phone, Type}  We created a surrogate key: Music\_ID  **Entertainer Table**  {Name} 🡪 {Phone, Type}  We created a surrogate key: Entertainer\_ID  **Photographer Table**  {Name} 🡪 {Phone}  We created a surrogate key: Photographer\_ID |

|  |
| --- |
| Tables Created:  **Event** (EventID, client\_ID, event\_Date, event\_Start\_Time, event\_Duration, event\_Type, num\_Of\_Guests, location\_ID, linenColor, num\_Of\_Waiters, num\_Of\_Bartenders, totalPrice, florist\_ID, florist\_Cost, music\_ID, musician\_Cost, entertainer\_ID, entertainer\_Cost, photographer\_ID, photographer\_Cost, menu\_ID)  **Client** (Client \_ID, Last\_Name, First\_Name ,Phone, Street, Zip)  **Menu** (Menu\_ID, menu\_Num, menu\_Appetizer, menu\_Salad, menu\_Main, menu\_Dessert)  **Location** (Location\_ID, location\_Name, location\_Street, Zip)  **Zip** (Zip\_ID, Zip, City, State)  **Florist** (Florist\_ID, Phone, Name)  **Music** (Music\_ID, Phone, Name, Contact)  **Entertainer** (Entertainer\_ID, Phone, Type, Name)  **Photographer** (Photographer\_ID, Phone, Name) |

1. 1NF:

As there are no multi valued columns present the above

1. 2NF:

In all the tables, every attribute is dependent on the primary key, and so they are in 2NF form.

1. 3NF:

In all tables, no attribute is dependent on any non-key attribute, and so there are no transitive dependencies. Hence all tables are in 3NF form.

1. BCNF:

All table dependencies include superkeys, hence all tables are in BCNF form.

## Lessons Learnt

# **Part 5**

## 

create or replace

TRIGGER t\_before\_update\_event before update on EVENT

for each row

begin

insert into Event\_hist1

(Event\_ID ,

Event\_Date ,

Event\_Start\_Time ,

Event\_Duration ,

Event\_Type ,

Num\_Of\_Guests ,

LinenColor ,

Num\_Of\_Waiters ,

Num\_Of\_Bartenders ,

TotalPrice ,

Florist\_Cost ,

Musician\_Cost ,

Entertainer\_Cost ,

Photographer\_Cost ,

Client\_Client\_ID ,

Location\_Location\_ID ,

Photographer\_Photographer\_ID ,

Menu\_Menu\_ID ,

Entertainer\_Entertainer\_ID ,

Florist\_ID ,

Music\_Music\_ID )

values

(:old.Event\_ID ,

:old.Event\_Date ,

:old.Event\_Start\_Time ,

:old.Event\_Duration ,

:old.Event\_Type ,

:old.Num\_Of\_Guests ,

:old.LinenColor ,

:old.Num\_Of\_Waiters ,

:old.Num\_Of\_Bartenders ,

:old.TotalPrice ,

:old.Florist\_Cost ,

:old.Musician\_Cost ,

:old.Entertainer\_Cost ,

:old.Photographer\_Cost ,

:old.Client\_Client\_ID ,

:old.Location\_Location\_ID ,

:old.Photographer\_Photographer\_ID ,

:old.Menu\_Menu\_ID ,

:old.Entertainer\_Entertainer\_ID ,

:old.Florist\_ID ,

:old.Music\_Music\_ID);

end t\_before\_update\_event;

1. ALTER TABLE Music

ADD LastInsert\_By varchar(30)

ADD LastInsert\_Date date;

commit;

create or replace TRIGGER MUSIC\_AUDIT

BEFORE INSERT ON Music

FOR EACH ROW

BEGIN

If INSERTING then

:new.LastInsert\_By := user;

:new.LastInsert\_Date := sysdate;

end if;

END;

ALTER TABLE Zip

ADD LastInsert\_By varchar(30)

ADD LastInsert\_Date date;

commit;

create or replace TRIGGER ZIP\_AUDIT

BEFORE INSERT ON Zip

FOR EACH ROW

BEGIN

If INSERTING then

:new.LastInsert\_By := user;

:new.LastInsert\_Date := sysdate;

end if;

END;

ALTER TABLE Client

ADD LastInsert\_By varchar(30)

ADD LastInsert\_Date date;

commit;

create or replace TRIGGER CLIENT\_AUDIT

BEFORE UPDATE ON Client

FOR EACH ROW

BEGIN

If UPDATING then

:new.LastInsert\_By := user;

:new.LastInsert\_Date := sysdate;

end if;

END;

(c) -- Update Queries  
UPDATE client SET FIRST\_NAME = 'Ola' WHERE CLIENT\_ID =1;  
UPDATE entertainer SET NAME = 'OlaHola' WHERE ENTERTAINER\_ID =1;  
UPDATE florist SET NAME = 'Lily' WHERE FLORIST\_ID =1;  
UPDATE photographer SET PHOTOGRAPHER\_NAME = 'Click' WHERE PHOTOGRAPHER\_ID =1;  
UPDATE location SET LOCATION\_NAME = 'Kashmir' WHERE LOCATION\_ID =1; 

**‘What we learned in the project’ and Summary**

The project was very helpful in learning triggers, index; both unique and non-unique, foreign keys, primary keys, triggers, sequences, natural keys, alternate keys, joins, subqueries etc.

We also learned the concepts of normalization. We converted a database from a raw data to a fully functional and efficient system with BCNF normalization. All the stages of normalization was reached in a proper step by step manner.

We learned about sequences in which the primary key of the table is automatically entered in the insert statements to a table using sequences. Sequences help the key to remain unique.

We learned the concept of foreign keys in an in- dept. fashion. We also realized that foreign keys play a very important role in maintaining the normalization of the database.

Primary keys are equally important because they help in uniquely identifying each and every row in a table.

Joins are of many types and they provide a lot of flexibility in extracting data from a database. In a normalized database the significant data is spread across many tables. So joins are required to bring meaningful data out of the database in front of the user from various tables at once.

Subqueries are also very helpful. Their name is apt. Subqueries are queries inside a query. They are usually used with ‘IN’ operators and present a range of values which satisfy a condition. These range of values can be compared with the main set of values in the outer query.

Triggers also form a very important part of the database. It comes under PL/SQL concept of the database. They are similar to stored procedures in a way that they can be invoked repeatedly but on certain conditions only. They cannot be explicitly called. They have many functions like to log events, prevent invalid transactions etc.

We also wrote select queries which included grouping queries. We also wrote DDL and DML statements.

We performed almost all the steps in successfully creating a data base in a step by step professional manner. In addition to this we also learned how to work in a team and maintain a healthy constructive environment.

This project also helped us to gain hands on experience on the aforementioned concepts.

**References**

1. <http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/triggers.htm#LNPLS723>
2. <http://docs.oracle.com/cd/B19306_01/server.102/b14200/clauses002.htm>
3. <https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_6015.htm>
4. <https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm>
5. <https://social.msdn.microsoft.com/Forums/sqlserver/en-US/27bd9c77-ec31-44f1-ab7f-bd2cb13129be/surrogate-keys-natural-keys-candidate-keys-composite-keys-and-super-keys?forum=sqlgetstarted>
6. <https://msdn.microsoft.com/en-us/library/aa291817(v=vs.71).aspx>
7. <http://docs.oracle.com/cd/B28359_01/gateways.111/b31053/c_array_handling.htm>