**Database Project – Final Report**

# Group Project

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.

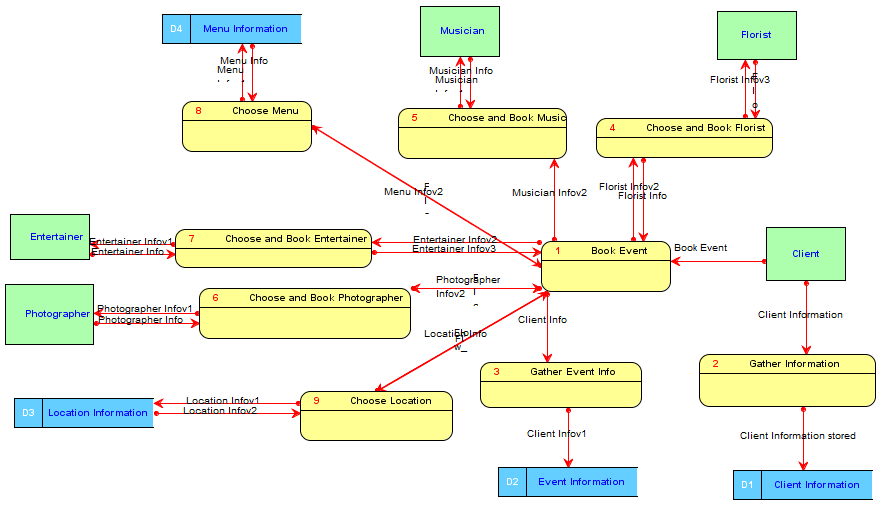
**Business Direction Information**

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

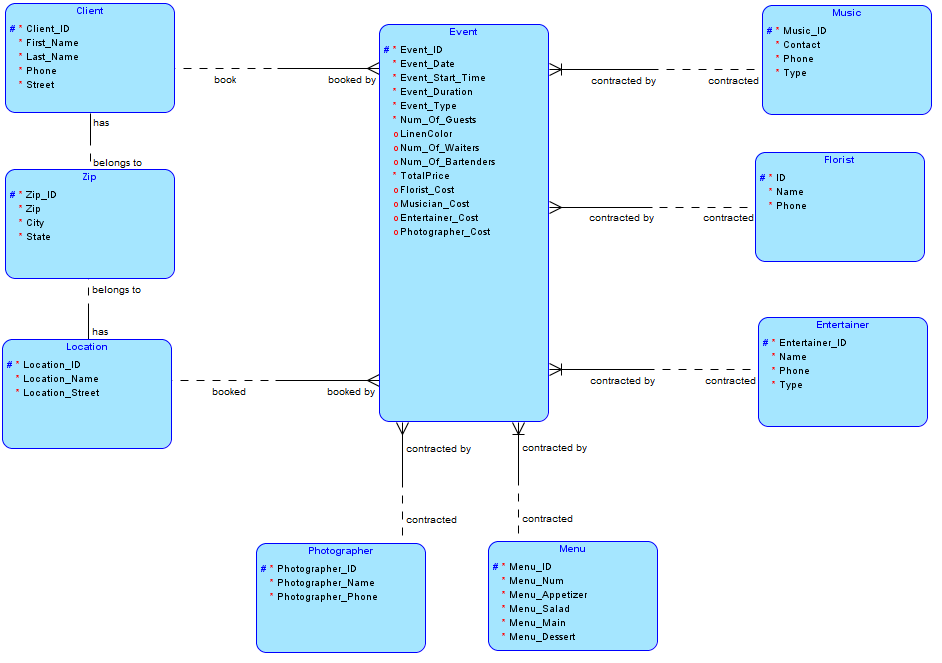
Problem: CCP does not have a database to record all its events and contractors data. As the business is growing it is difficult maintain all the data manually. The company also wants to create a website for the business for which database is required.

Business Objective: To improve the maintenance of records.

## Data Flow Diagram (DFD)

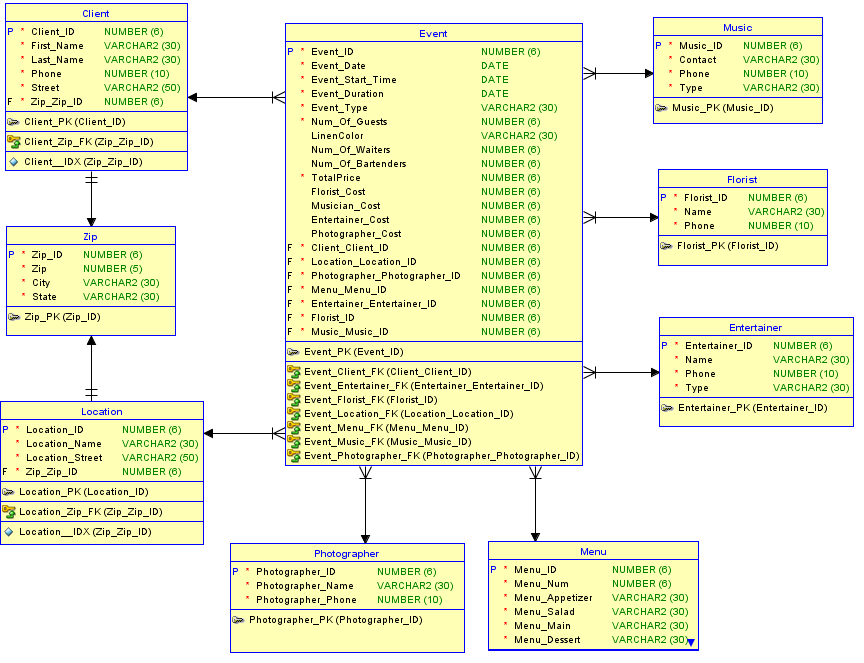


## Entity-Relation Diagram (ERD)



# **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

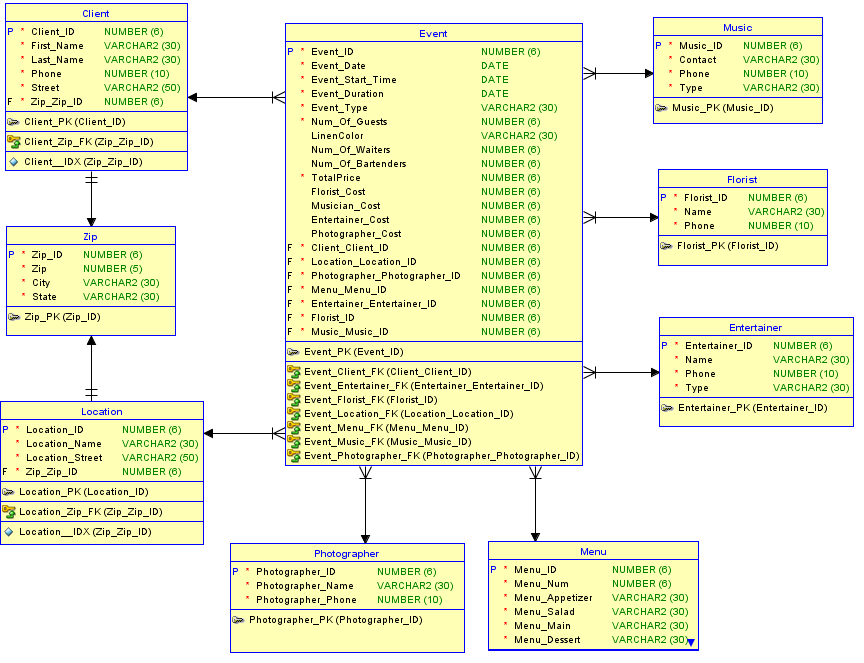
**Zip Table:**

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

|  |
| --- |
|  |

## 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

|  |
| --- |
|  |

## Insert Database Queries

|  |
| --- |
|  |

## Select Queries

|  |
| --- |
|  |

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**

## --

## --

## --

## --

## Lessons Learnt