**Student Projects Chapter 6 - Normalizing the Relational Model for the Student Project and Creating a Normalized Oracle Database**

Read the sample project steps for this chapter and apply the same techniques to the student project that you are developing.

* Step 6.1 - Begin with the list of the tables that the entities and relationships from the E-R diagram mapped to naturally, from the sample project section at the end of chapter 4. For each table on the list, identify functional dependencies and normalize the relation to BCNF. Then decide whether the resulting tables should be implemented in that form. If not, explain why.

**The following tables resulted from the mapping in chapter 4:**

**Member**(memberID, address, areaCode, city, email, firstName, lastName, phoneNumber, state, zip)

**Play**(playID, author, numberOfActs, title, type, numberOfSets)

**Sponsor** (businessName, income)

**Subscriber** (subscriberID, address, areaCode, city, email, firstName, lastName, pastPlays, phoneNumber, state, zip)

**Production** (playID*,* cost, productionPostions, yearOfTheProduction, seasonStartDate, seasonEndDate)

**Performance** (memberID, date, time, year)

**TicketSale** (ticketIncome)

**DuesPayment (**memberID*,* duesPaid)

**Donation** ( businessName*,* *donationNumber*)

**Ticket﻿﻿﻿﻿﻿﻿﻿** (*qrCode*, playDate, *playNames*, playTime, price, seat, seatNumbers)

**Member**(memberID, address, areaCode, email, firstName, lastName, phoneNumber, *zip*)

**Play**(playID, author, numberOfActs, title, type, numberOfSets)

**Sponsor** (businessName, income)

**Subscriber** (subscriberID, address, areaCode, email, firstName, lastName, pastPlays, phoneNumber, *zip*)

**Production** (playID*,* cost, yearOfTheProduction, seasonStartDate, seasonEndDate)

**Performance** (playID, date, time, year)

**TicketSale** (*subscriberID*, *playID,* ticketIncome)

**DuesPayment (**memberID*,* duesPaid, dateofDuesSubmission, duesAmount)

**Donation** ( businessName*,* donationNumber)

**Ticket﻿﻿﻿﻿﻿﻿﻿** (*qrCode*, subscriberID, playDate, *playNames*, playTime, price, seat, seatNumbers)

**ZipCode**(zip,city,state)

For the Member table, let us identify FDs:

memberID → all attributes

firstName + lastName → all attributes (firstName and lastName are not unique identifiers)

zip → city + state (creating a new table called ZipCode(zip, city, state))

This table is 1NF, 2NF (not 3NF or BCNF) because of these dependencies.

For the Play table, let us identify FDs:

playID → all attributes

author + title → all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the Sponsor table, let us identify FDs:

{businessName} → {income}

This table is 1NF, 2NF, 3NF, BCNF because of these dependencies.

For the Subscriber table, let us identify FDs:

subscriberID → all attributes

firstName + lastName → all attributes

email + phoneNumber → all attributes

zip → city + state (creating a new table called ZipCode(zip, city, state))

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the Production table, let us identify FDs:

playID → all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the Performance table, let us identify FDs:

playID → all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the TicketSale table, let us identify FDs:

subscriberID + playID→ all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the DuesPaymenttable, let us identify FDs:

memberID→ all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the Donationtable, let us identify FDs:

businessName → all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the Tickettable, let us identify FDs:

qrCode → all attributes

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

For the ZipCodetable, let us identify FDs:

zip → city + state (creating a new table called ZipCode(zip, city, state))

This table is 1NF, 2NF, 3NF, BCNF because of the dependencies.

* Step 6.2 - Update the data dictionary and list of assumptions as needed.

no updates necessary.

* Step 6.3 - For each table, write the table name and write out the names, data types, and sizes of all the data items, Identify any constraints, using the conventions of the DBMS you will use for implementation.

**Table Member**

memberID VARCHAR 15 UNIQUE

address VARCHAR 50

areaCode CHAR 3

email VARCHAR 30

firstName VARCHAR 16 (lastName,firstName) PRIMARY KEY

lastName VARCHAR 20 (lastName,firstName) PRIMARY KEY

phoneNumber VARCHAR 7

zip CHAR 5 FOREIGN KEY REF ZipCode

**Table Play**

playID VARCHAR 15 UNIQUE

author VARCHAR 20

numberOfActs NUMBER 2

title VARCHAR 15 PRIMARY KEY

type VARCHAR 15

numberOfSets VARCHAR 2

**Tableb Sponsor**

businessName VARCHAR 15 PRIMARY KEY

income 8,2

**Table Subscriber**

subscriberID VARCHAR 15 PRIMARY KEY

address VARCHAR 50

areaCode CHAR 3

email VARCHAR 30

firstName VARCHAR 15

lastName VARCHAR 20

pastPlays CHAR 30

phoneNumber CHAR 7 UNIQUE

zip CHAR 5 FOREIGN KEY REF ZipCode

**Table Production**

playID VARCHAR 15 PRIMARY KEY, FOREIGN KEY

cost NUMBER 8,2

yearOfTheProduction CHAR 4

seasonStartDate DATE

seasonEndDate DATE

**Table Performance**

playID VARCHAR 15 UNIQUE, PRIMARY KEY, FOREIGN KEY

“date” DATE PRIMARY KEY

time VARCHAR 10

year CHAR 4 PRIMARY KEY

**Table TicketSale**

subscriberID VARCHAR 15 PRIMARY KEY, FOREIGN KEY

title VARCHAR 30 PRIMARY KEY, FOREIGN KEY

ticketIncome NUMBER 8,2

**Table DuesPayment**

memberID VARCHAR 15 PRIMARY KEY, FOREIGN KEY

duesPaid CHAR 1

dateofDuesSubmission DATE

duesAmount NUMBER

**Table Donation**

businessName VARCHAR 25 PRIMARY KEY

donationNumber VARCHAR 10 PRIMARY KEY

**Table Ticket**

qrCode VARCHAR 10 UNIQUE

subscriberID VARCHAR 15 PRIMARY KEY,FOREIGN KEY

playDate DATE

playNames VARCHAR 25

playTime VARCHAR 10

price VARCHAR 10

seat CHAR 3

seatNumbers VARCHAR 3

**Table ZipCode**

zip CHAR 5 PRIMARY KEY

city VARCHAR 20

state CHAR 2

* Step 6.4 - Write and execute SQL statements to create all the tables needed to implement the design.

CREATE TABLE ZipCode(

zip CHAR(5),

city VARCHAR(20),

state CHAR(2),

CONSTRAINT ZipCode\_zip\_pk PRIMARY KEY(zip)

);

CREATE TABLE Member(

memberID VARCHAR(15),

address VARCHAR(50),

areaCode CHAR(3),

email VARCHAR(30),

firstName VARCHAR(16),

lastName VARCHAR(20),

phoneNumber VARCHAR(7),

zip CHAR(5),

CONSTRAINT Member\_mID\_uk UNIQUE (memberID),

CONSTRAINT Member\_Lname\_Fname\_pk PRIMARY KEY (lastName, firstName),

CONSTRAINT Member\_zip\_fk FOREIGN KEY(zip) REFERENCES ZipCode(zip)

);

CREATE TABLE Play(

playID VARCHAR(15),

author VARCHAR(20),

numberOfActs NUMBER(2,0),

title VARCHAR(30),

type VARCHAR(15),

numberOfSets NUMBER(2),

CONSTRAINT Play\_playID\_uk UNIQUE (playID),

CONSTRAINT Play\_title\_pk PRIMARY KEY (title)

);

CREATE TABLE Sponsor(

businessName VARCHAR(15),

income NUMBER(8,2),

CONSTRAINT Sponsor\_busName\_pk PRIMARY KEY (businessName)

);

CREATE TABLE Subscriber(

subscriberID VARCHAR(15),

address VARCHAR(50),

areaCode CHAR (3),

email VARCHAR(30),

firstName VARCHAR(15),

lastName VARCHAR(20),

pastPlays VARCHAR(30),

phoneNumber CHAR(7),

zip CHAR(5),

CONSTRAINT Subscriber\_subID\_pk PRIMARY KEY (subscriberID),

CONSTRAINT Subscriber\_zip\_fk FOREIGN KEY(zip) REFERENCES ZipCode(zip)

);

CREATE TABLE Production(

playID VARCHAR(15),

cost NUMBER(8,2),

yearOfTheProduction CHAR(4),

seasonStartDate DATE,

seasonEndDate DATE,

CONSTRAINT Production\_plID\_pk PRIMARY KEY (playID),

CONSTRAINT Production\_plID\_fk FOREIGN KEY (playID) REFERENCES Play(playID)

);

CREATE TABLE Performance(

playID VARCHAR(15),

“date” DATE,

time VARCHAR(10),

year CHAR (4),

CONSTRAINT Performance\_playID\_dte\_year\_pk PRIMARY KEY (playID, “date”, year),

CONSTRAINT Performance\_playID\_fk FOREIGN KEY(playID)REFERENCES Play(playID)

);

CREATE TABLE TicketSale(  
 subscriberID VARCHAR(15),

title VARCHAR(30),   
 ticketIncome NUMBER(8,2),

CONSTRAINT TicketSale\_subID\_pk PRIMARY KEY (subscriberID),

CONSTRAINT TicketSale\_subscriberID\_fk FOREIGN KEY (subscriberID) REFERENCES Subscriber(subscriberID),

CONSTRAINT TicketSale\_plTitle\_fk FOREIGN KEY (title) REFERENCES Play(title)

);

CREATE TABLE DuesPayment(

memberID VARCHAR(15),

duesPaid CHAR(1),

dateofDuesSubmission DATE,

duesAmount NUMBER,

CONSTRAINT DuesPayment\_mID\_duesPaid\_pk PRIMARY KEY(memberID, duesPaid),

CONSTRAINT DuesPayment\_mID\_fk FOREIGN KEY(memberID)REFERENCES Member(memberID)

);

CREATE TABLE Donation(  
 businessName VARCHAR(25),  
 donationNumber VARCHAR(10),

CONSTRAINT Donation\_donNum\_pk PRIMARY KEY(donationNumber)

);

CREATE TABLE Ticket(

qrCode VARCHAR(10),

subscriberID VARCHAR(15),

playDate DATE,

playNames VARCHAR(25),

playTime VARCHAR(10),

price VARCHAR(10),

seat CHAR(3),

seatNumbers VARCHAR(3),

CONSTRAINT Ticket\_qrCode\_uk UNIQUE (qrCode),

CONSTRAINT Ticket\_subID\_pk PRIMARY KEY (subscriberID),

CONSTRAINT Ticket\_subID\_fk FOREIGN KEY(subscriberID)REFERENCES Subscriber(subscriberID)

);

* Step 6.5 - Create indexes for foreign keys and any other columns that will be used most often for queries.

CREATE INDEX Member\_zip ON Member(zip);

CREATE INDEX Subscriber\_zip ON Subscriber(zip);

CREATE INDEX Member\_info ON Member (firstName, lastName, email)

CREATE INDEX Subscriber\_info ON Subscriber (firstName, lastName, email)

CREATE INDEX Dues\_info ON DuesPayment(memberID, duesPaid)

CREATE INDEX Tickets\_info ON Tickets (subscriberID, seat, seatNumber)

CREATE INDEX Member\_memberId ON Member(memberId);

CREATE INDEX Subscriber\_subscrinerId ON Subscriber(subscriberId);

* Step 6.6 - Insert about five records in each table, preserving all constraints. Put in enough data to demonstrate how the database will function.

INSERT INTO ZipCode VALUES ('33601', 'Tampa','FL');

INSERT INTO ZipCode VALUES ('10001', 'Manhattan', 'NY');

INSERT INTO ZipCode VALUES ('07101', 'Newark','NJ');

INSERT INTO ZipCode VALUES('33010', 'Miami', 'FL');

INSERT INTO ZipCode VALUES('60601', 'Chicago', 'IL');

INSERT INTO Member VALUES ('000001','345 Stein Rd', '546', 'judge.aaron@gmail.com', 'Aaron', 'Judge', '6437865', '33601');

INSERT INTO Member VALUES ('000002', '23 Garden Way', '718', 'lebronnyk@gmail.com', 'Lebron', 'James', '2346432', '10001');

INSERT INTO Member VALUES ('000003','4 Superbowl St', '914', 'objdances@gmail.com', 'Odell', 'Beckham Jr', '7957865', '07101');

INSERT INTO Member VALUES ( '000004', '5403 Ranger Way','718', 'king30@yahoo.com', 'Henrik', 'Lundqvist', '9761002','10001');

INSERT INTO Member VALUES ('000005','30 Warrior Ave', '415', 'wardell@gmail.com', 'Stephen', 'Curry', '1112235', '94016');

INSERT INTO Play VALUES ('143', 'Hao Dong', '3', 'Lion King', 'Drama', '20');

INSERT INTO Play VALUES ('213', 'Jimmy Buffett', '3', 'Margaritaville', 'Musical', '32');

INSERT INTO Play VALUES ('658', 'Robin Williams', '3', 'Aladdin', 'Drama', '22');

INSERT INTO Play VALUES ('496', 'Lin Manuel', '3', 'Hamilton', 'Musical', '18');

INSERT INTO Play VALUES ('104', 'Jack Black', '3', 'Rock of Ages', 'Action', '13');

INSERT INTO Sponsor VALUES('McDonalds', '2000.20');

INSERT INTO Sponsor VALUES('Adidas', '14000.00');

INSERT INTO Sponsor VALUES('Under Armour', '5000.60');

INSERT INTO Sponsor VALUES('Hershey', '1000.00');

INSERT INTO Sponsor VALUES('Peanuts', '4543.56');

INSERT INTO Subscriber VALUES ('100001', '1 Marble Ave', '914', 'jadler@gmail.com', 'Jake', 'Adler', 'Aladdin', '9873566', '33601');

INSERT INTO Subscriber VALUES ('100002', '4456 E. 53rd Street', '718', 'erob234@gmail.com', 'Emily', 'Robinson','Aladdin', '9773126', '10001');

INSERT INTO Subscriber VALUES ('100003', '312 Bush Rd','914', 'farriswheel@gmail.com', 'Michaela', 'Farris ','Aladdin','9870000','07101');

INSERT INTO Subscriber VALUES ('100004', '5 West 100th St','718', 'sanderstim@yahoo.com', 'Timmy', 'Sanders','Margaritaville','4453566', '10001');

INSERT INTO Subscriber VALUES ('100005', '2018 Setters Ave', '914', 'rjax1205@gmail.com', 'Rick', 'Jackson','Aladdin','1233212', '94016');

INSERT INTO Production VALUES ('213', '40650.00', '2017', '01-Jan-2017', '31-Dec-2017');

INSERT INTO Production VALUES ('658', '31000.00', '2018', '01-Jan-2018', '31-Dec-2018');

INSERT INTO Production VALUES ('104', '25570.00', '2019', '01-Jan-2019', '31-Dec-2019');

INSERT INTO Performance VALUES ('213', '05-May-2017', '6:00 PM', '2017');

INSERT INTO Performance VALUES ('213', '05-Dec-2017', '6:00 PM', '2017');

INSERT INTO Performance VALUES ('658', '09-May-2018', '6:00 PM', '2018');

INSERT INTO Performance VALUES ('658', '12-Dec-2018', '6:00 PM', '2018');

INSERT INTO Performance VALUES ('104', '23-May-2019', '6:00 PM', '2019');

INSERT INTO TicketSale VALUES ('100001', 'Rock of Ages', '45.00');

INSERT INTO TicketSale VALUES ('100002', 'Rock of Ages', '45.00');

INSERT INTO TicketSale VALUES ('100003', 'Rock of Ages', '45.00');

INSERT INTO TicketSale VALUES ('100004', 'Rock of Ages', '45.00');

INSERT INTO TicketSale VALUES ('100005', 'Hamilton', '45.00');

INSERT INTO Donation VALUES ('Lucios Pizza', '100');

INSERT INTO Donation VALUES ('Sals Pizza', '101');

INSERT INTO Donation VALUES ('Ons', '102');

INSERT INTO Donation VALUES ('Pleasantville Diner', '103');

INSERT INTO Donation VALUES ('Thornwood Diner', '104');

INSERT INTO Ticket VALUES ('123456', '100001', '23-May-2019', 'Rock of Ages', '6:00 PM', '45.00', 'B', '12');

INSERT INTO Ticket VALUES ('123457', '100002', '23-May-2019', 'Rock of Ages', '6:00 PM', '45.00', 'B', '13');

INSERT INTO Ticket VALUES ('123458', '100003', '23-May-2019', 'Rock of Ages', '6:00 PM', '45.00', 'B', '14');

INSERT INTO Ticket VALUES ('123459', '100004', '23-May-2019', 'Rock of Ages', '6:00 PM', '45.00', 'B', '15');

INSERT INTO Ticket VALUES ('123480', '100005', '23-May-2019', 'Rock of Ages', '6:00 PM', '45.00', 'B', '16');

INSERT INTO DuesPayment VALUES('000001', 'Y', '03-Jan-2019', '20.00');

INSERT INTO DuesPayment VALUES('000002', 'Y', '03-Jan-2019', '20.00');

INSERT INTO DuesPayment VALUES('000003', 'N', '', '20.00');

INSERT INTO DuesPayment VALUES('000004', 'Y', '03-Jan-2019', '20.00');

INSERT INTO DuesPayment VALUES('000005', 'Y', '03-Jan-2019', '20.00');

* Step 6.7 - Write SQL statements that will process five non-routine requests for information from the database just created. For each, write the request in English, followed by the corresponding SQL command.

1. Find the average cost of any production the theater has produced so far.

SELECT AVG(cost)

FROM Production;

2. Find subscribers who saw the production of Rock Of Ages.

SELECT firstName, lastName

FROM Subscriber

WHERE pastPlays = 'Rock Of Ages';

3. Find the members who are not from zip code 10001 and display their names and the zip code from which they are from.

SELECT firstName, lastName, zip

FROM Member

WHERE zip != '10001';

4. Find all the musical plays where the number of sets is greater than the average number of sets for all plays.

SELECT title

FROM Play

WHERE numberOfSets >= (SELECT AVG(numberOfSets) FROM Play) AND type = 'Musical';

5. Find all sponsors where the donation value is greater than $2,000 and display their name.

SELECT businessName

FROM Sponsor

WHERE income > 2000.00;

* Step 6.8 - Create at least one trigger and write the code for it.

CREATE OR REPLACE TRIGGER update\_pastPlays

after insert on TicketSale

for each row

begin

update Subscriber

set pastPlays = :NEW.title

where Subscriber.subscriberID = :NEW.subscriberID;

end;

/

**Creating a new subscriber “100011” where pastPlay = “Cats”**

INSERT INTO Subscriber VALUES ('100011', '201 Setters Ave', '914', 'bcip1205@gmail.com', 'Cipola', 'Brian','Cats','1233212', '94016');

**Query results in “Cats”**

SELECT pastPlays

FROM Subscriber

WHERE subscriberID = 100011;

**Creating a new TicketSale for subscriber “100011” where title = “Aladdin”**

INSERT INTO TicketSale VALUES ('100011', 'Aladdin', '45.00');

**Query results in “Aladdin”**

SELECT pastPlays

FROM Subscriber

WHERE subscriberID = 100011;

