**Chapter 6 – Team Project: Normalizing the Relational Model for the Team Project and Creating a Normalized Oracle Database**

Read the sample project steps for this chapter and apply the same techniques to the team project that you are developing. For the team project, do the following:

**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.

Solution:



Client (clientId, firstName, lastName, street, city, state, zip, areaCode, phoneNumber)

Meeting (clientId, meetDate, meetTime, repName)

Photographer (empID, firstName, lastName, street, city, state, zip, areaCode, number)

Booking (clientId, bookingDate, bookingTime, duration, type, empID)

PackageMenu (packageNo, numWallet, num5by7, num8by10, num11by14, num16by20, albumType, albumPages, albumCover, price)

Job (contractNo, type, eventName, location, clientId , date, time, duration, cost, empid, packageNoChosen, totalCost, amtPaid, amtDue)

Proof (contractNo, proofNo, quality)

Order (orderNo, dateOrdered, totalAmount, packageNoOrdered, contractNo)

OrderItem (orderNo, proofNo, size, quantity, dateDelivered)

Payment (Paymentid, contractNo, datePaid, payType, amount)

Changes made to the tables as per BCNF rule to remove functional dependencies (fds) are as follows:

**Changes were made to Job table because they had functional dependencies and also a new attribute Paymentid was added to the was added to payment table.**

**Also new foreign keys, Primary keys and unique keys are defined.**

1. Client table: all the attributes on the first table are dependent on ClientID so it is a unique numeric identifier and is also the primary key for this table. It can also be called as surrogate key and there are no functional dependencies seen in client table.
2. Meeting table: we do not have any functional dependencies here, so we keep the table as it is.
3. Photographer: we do not have any functional dependencies here so we keep the table as it is.
4. Booking table:

we do not have any functional dependencies here so we keep the table as it is.

1. PackageMenu table: we do not have any functional dependencies here, so we keep the table as it is.
2. Job table:

Contract number 🡪 all attributes

Totalcost 🡪 all attributes

there are dependencies, so we are creating 2 new different tables ie. JobContract,

JobPayment

1. Proof table: we do not have any functional dependencies here so we keep the table as it is. Proof number Is the primary key here and contract number is the foreign key.
2. Order table: In order table orderno is the primary key whereas contractno is the foreign key
3. Orderitem table proofno and contractno number are connected but they are not functionally dependent to each other so we leave the table in its original form
4. In Payment table paymentid is a new updated attribute (primary) whereas contractno is the foreign key

**Step 6.2 – Review and update the data dictionary and list of assumptions (as needed).**

**Solution:** Data Dictionary:

ClientId: Identification Number of Client

ClientfirstName: First Name of the client

ClientlastName: Last Name of the client

Clientaddress: complete address of the client (street,city,state,zip,areacode)

ClientphoneNumber: contact number of client

meetDate- Date of the meet

meetTime-time of meet

repName- Name of receptionist

PhotographerempId: Photographer id number

PhotographerfirstName: Photographer First Name

PhotographerlastName: Photographer Last Name

Photographeraddress: Complete address of the photographer (street,city,state,zip,areacode)

PhotographerphoneNumber: Phone number of the photographer

BookingDate: Date of event

BookingTime: Time of the event

BookingDuration: duration of event

Bookingtype: type of event(weeding,birthday..etc)

PackagemenuNumber: identification number of package

PackagaeMenunumwallet:

PackageMenunum5by7: size of photo 5 x 7

PackageMenunum8by10: size of photo 8 x 10

PackageMenunum11by14: size of photo 11 x 14

PackageMenunum16by20: size of photo 16 x 20

PackageMenualbumType:

PackageMenualbumPages: number of pages in number

PackageMenualbumCover: cover type

PackageMenuprice: Total PackageMenu Cost

JobcontractNo: Contract Number

JobeventName: Name of the event

Joblocation: Location of the event

JobpackageNoChosen: type of package chosen

JobtotalCost: total cost

JobamtPaid: Amount Paid

JobamtDue: Amount due

proofNo: Proof of Payment

proofquality: quality check

orderNo: order number of job

OderdateOrdered: Date of order

OrderpackageNoOrdered: Package Number of order

OrderItemquantity: total quantity

OrderItemdateDelivered:

PaymentdatePaid: Dates amount was paid on

PaymentpayType: type of Payment Cash/card

Paymentamount: Total payment Completed

List of assumptions:

1. Clients have unique identifiers such as their first and last name.

2. The database contains the client's contact information as well as payment information from when they signed the contract.

3. Unique identification is required for every photographer in the studio.

4. Method of payment for an event should be either credit/Debit or cash.

5. A single photographer may have covered several events.

6. The unique id for the packages available.

7. Each Package is identifiable by an unique identification that is issued to it.

8. At least one lead photographer and one optional helper photographer are required for each event.

9. When signing the contract, the client should pay an initial deposit and then make

further payments.

10. Before signing the contract client should provide proper identification proof.

11. The studio provides a list of photographer’s contact details.

12. The final payment amount may change from the first request.

13. Photographer type values include ordinary photographers and freelance

photographers.

14. The event and seating values are on the list of Event kinds.

15. Event location and time will be informed to the client a week prior to the actual event.

16. The final payment amount may change from the first request.

17. The database does not include file metadata for photos retained by the studio for

six months and then deleted.

**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.

Solution:

For an Oracle database, the tables will have the structures shown below.

TABLE Client

Item Datatype Size Constraints Comments

Clientid Number 03 not null PRIMARY KEY

lastName VARCHAR2 20 not null UNIQUE

firstName VARCHAR2 15 not null UNIQUE

street VARCHAR2 20

city VARCHAR2 15

state CHAR 02

zip NUMBER 05

areaCode NUMBER 03

phoneNumber NUMBER 07

TABLE Meeting

Item Datatype Size Constraints Comments

Clientid NUMBER 03 not null Foreign KEY REF client

meetdate DATE

repName VARCHAR2 20

TABLE Booking

Item Datatype Size Constraints Comments

Clientid NUMBER 03 not null Foreign KEY REF client

bookingDate DATE

empid VARCHAR2 20 not null Foreign KEY. REF photographer

TABLE Packagemenu

Item Datatype Size Constraints Comments

Packageno NUMBER. 6 not null PRIMARY KEY,

numwallet NUMBER 4

num5by7 NUMBER 4

num8by10 NUMBER 4

num11by14 NUMBER 4

num16by20 NUMBER 4

albumType VARCHAR2 20

albumPages NUMBER 4

albumCover VARCHAR2 20

price NUMBER 20

TABLE JOB1

Item Datatype Size Constraints Comments

Contractno NUMBER 6 not null PRIMARY KEY

Type VARCHAR2 10

eventname VARCHAR2 2

Location VARCHAR2 5

Date1 DATE

Time timestamp

duration DECIMAL 4,4

cost DECIMAL 9,2

empid VARCHAR2 20 not null Foreign KEY REF photographer

Packagenochosen VARCHAR2 5

TABLE JOB2

Item Datatype Size Constraints Comments

Clientid NUMBER 3 not null Foreign KEY REF client

totalcost DECIMAL 9,2

amtpaid DECIMAL 9,2

amtdue DECIMAL 9,2

TABLE Proof

Item Datatype Size Constraints Comments

contractNo NUMBER 6 not null Foreign KEY REF Job

proofNo NUMBER 10 NOT NULL PRIMARY KEY

quality VARCHAR2 20

TABLE Orders

Item Datatype Size Constraints Comments

orderNo NUMBER 10 NOT NULL PRIMARY KEY

dateOrdered DATE

totalAmount DECIMAL 9,2

packageNoOrdered VARCHAR. 5

contractNo NUMBER. 10 not null Foreign KEY REF JOB

TABLE Orderitem

Item Datatype Size Constraints Comments

orderNo NUMBer 5 NOT NULL PRIMARY KEY

proofNo NUMBER 5 not null Foreign KEY REF Proof

sizes VARCHAR 5

quantity NUMBER 5

dateDelivered DATE

TABLE Payment

Item Datatype Size Constraints Comments

Payment id NUMBER 2 not null PRIMARY KEY

contractNo NUMBER 10 not null Foreign KEY REF JOB1

datePaid date

payType varchar 5

amount NUMBER 5

TABLE photographer

Item Datatype Size Constraints Comments

empid NUMBER 2 not null PRIMARY KEY

lastName VARCHAR2 10 not null UNIQUE

firstName VARCHAR2 10 not null UNIQUE

street VARCHAR2 10

city VARCHAR2 8

state CHAR 2

zip CHAR 5

areaCode CHAR 3

phoneNumber CHAR 7

**Step 6.4 - Design SQL statements to create all tables needed to implement the design. Then create the tables in the database.** Show your work by providing screenshots of executing the CREATE TABLE SQL statements in the database.

Dropping all the previous tables:

-- drop the existing tables for the Image photography studio,

drop table Client;

drop table Meeting;

drop table Booking;

drop table Orders;

drop table Orderitem;

drop table Job1;

drop table photographer;

drop table payment;

drop table packagemenu;

**SQL Statement to be executed:**

CREATE TABLE Client (

Clientid Number(3) not null,

lastName VARCHAR2(20),

firstName VARCHAR2(15),

street VARCHAR2(20),

city VARCHAR2(15),

state CHAR (2),

zip NUMBER (5),

areaCode NUMBER (3),

phoneNumber NUMBER (7),

CONSTRAINT Client\_clientid\_pk PRIMARY KEY (clientid),

CONSTRAINT Client\_firstName\_lastName\_uk UNIQUE (firstName, lastName)

);

Create TABLE meeting

(

Clientid NUMBER(3) not null,

meetdate DATE,

repName VARCHAR(20),

CONSTRAINT meeting\_clientid\_fk FOREIGN KEY (clientid) REFERENCES Client(Clientid)

);

Create TABLE Booking

(

Clientid NUMBER (3) not null,

bookingDate DATE,

empid VARCHAR2(20),

CONSTRAINT booking\_clientid\_fk FOREIGN KEY (clientid) REFERENCES Client(Clientid)

);

Create TABLE Packagemenu

(

Packageno varchar2(6) not null PRIMARY KEY,

numwallet NUMBER (4),

num5by7 NUMBER (4),

num8by10 NUMBER (4),

num11by14 NUMBER (4),

num16by20 NUMBER (4),

albumType VARCHAR2(20),

albumPages NUMBER (4),

albumCover VARCHAR2(20),

price NUMBER (20)

);

Create TABLE Job1

(

Contractno NUMBER (6) not null PRIMARY KEY,

Type VARCHAR2(10),

eventname VARCHAR2(2),

Location VARCHAR2(5),

Dateofjob DATE,

duration NUMBER(4),

cost NUMBER(3),

empid NUMBER(2) not null,

Packagenochosen VARCHAR2(5)

);

Create TABLE Job2

(

Clientid NUMBER(3),

totalcost NUMBER(3),

amtpaid NUMBER(3),

amtdue NUMBER(3),

CONSTRAINT Job2\_clientid\_fk FOREIGN KEY (clientid) REFERENCES Client(Clientid)

);

CREATE TABLE proof(

contractNo NUMBER(6) not null,

proofNo NUMBER(10) NOT NULL PRIMARY KEY,

quality VARCHAR2(20),

CONSTRAINT Proof\_contractNo\_fk FOREIGN KEY (contractNo) REFERENCES Job1(contractNo)

);

CREATE TABLE ORDERS

(

orderNo NUMBER(10) NOT NULL PRIMARY KEY,

dateOrdered DATE,

totalAmount DECIMAL(9,2),

packageNoOrdered VARCHAR(5),

contractNo NUMBER(6),

CONSTRAINT orders\_contractNo\_fk FOREIGN KEY (contractNo) REFERENCES Job1(contractNo));

);

Create table OrderItem

(

orderNo NUMBER(5) NOT NULL PRIMARY KEY,

proofNo NUMBER(10) not null,

sizes VARCHAR(5),

quantity NUMBER(5),

dateDelivered DATE,

CONSTRAINT orderitem\_proofNo\_fk FOREIGN KEY (proofNo) REFERENCES proof(proofNo)

);

CREATE TABLE Payment

(

Paymentid Number(2) Not null primary key,

contractNo NUMBER(10) NOT NULL,

datePaid date,

payType varchar (5),

amount NUMBER(5),

CONSTRAINT payment\_contractNo\_fk FOREIGN KEY (contractNo) REFERENCES Job1(contractNo)

);

CREATE TABLE photographer

(

empid NUMBER(2) not null PRIMARY KEY,

lastName VARCHAR2(10),

firstName VARCHAR2(10),

street VARCHAR2(10),

city VARCHAR2(8),

state CHAR(2),

zip CHAR(5),

areaCode CHAR(3),

phoneNumber CHAR(7),

CONSTRAINT photographer\_firstName\_lastName\_uk UNIQUE(firstName, lastName)

);

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

**Step 6.5 - Design SQL statements to create indexes for foreign keys and for any other columns that will be used most often for queries. Then execute the SQL statements in the database.** Show your work by providing screenshots of executing the SQL statements in the database.

Solution:

create index ClientIndex on CLient ( CLIENTID, firstname, lastName, city, zip, areaCode, state, street);

create index meetingIndex on meeting (CLIENTID, meetdate);

create index bookingINdex on Booking (CLIENTID, bookingdate,empid);

create index PMIndex on Packagemenu (packageno,price);

create index JOB1Index on JOB1(contractno,eventname);

create index JOB2Index on JOB2(clientid,totalcost,amtpaid,amtdue);

create index paymentIndex on payment (contractno,paytype,amount);

create index orderindex on orders (orderno,totalamount,contractno);

Text

Description automatically generated

**Step 6.6 - Design SQL statements to insert at least five records in each table, preserving all constraints.** **Then insert the records into the tables.** Show your work by providing screenshots of executing the INSERT SQL statements in the database.

Ans.

CLIENT:

INSERT INTO Client VALUES('11','Hrithik','Roshan','South','MUMBAI','MH','10013','412','1662334');

INSERT INTO Client VALUES('12','Mukesh','Ambani','South','chapelhill','OH','10023','312','1122334');

INSERT INTO Client VALUES ('13','Ratan','tata','East','Pune','NJ','20023','412','5544223');

INSERT INTO Client VALUES ('14','Elon' ,'Musk','west','queens','NY','10007','752','9989811');

INSERT INTO Client VALUES ('15','Allu','Arjun','Norteast','newyorkcity','NY','10007','752','4354231');

Meeting

Insert into meeting(‘’,’’,’’);

Insert into meeting(‘’,’’,’’);

Insert into meeting(‘’,’’,’’);

Insert into meeting(‘’,’’,’’);

Insert into meeting(‘’,’’,’’);

Booking:

INSERT INTO Booking VALUES('11','11-FEB-2013','01');

INSERT INTO Booking VALUES('12','11-MAR-2020','02');

INSERT INTO Booking VALUES('13','11-JUN-2021','01');

INSERT INTO Booking VALUES('14','12-JUN-2021','01');

INSERT INTO Booking VALUES('15','21-JUL-2021','02');

PACKAGEMENU:

INSERT INTO Packagemenu VALUES('a','14','00','10','12','00','WED','20','M','120');

INSERT INTO Packagemenu VALUES('b','21','10','00','12','13','BIRTH','20','L','220');

INSERT INTO Packagemenu VALUES('d','20','20','10','00','00','EVNT','20','S','130');

INSERT INTO Packagemenu VALUES('e','13','05','00','12','13','WED','20','L','260');

INSERT INTO Packagemenu VALUES('f','19','22','10','12','00','BIRTH','20','s','145');

Orders values:

INSERT INTO Orders VALUES ('2','12-FEB-2013','120','25','5');

INSERT INTO Orders VALUES ('5','12-MAR-2020','10','24','9');

INSERT INTO Orders VALUES ('4','11-JUN-2021','20','12','8');

INSERT INTO Orders VALUES ('1','12-JUN-2021','78','45','7');

INSERT INTO Orders VALUES ('7','21-JUL-2021','26','45','2');

ORDERITEM:

INSERT INTO Orderitem VALUES('225','124','m','10','12-JAN-2013');

INSERT INTO Orderitem VALUES ('226','126','l','15','12-JAN-2020');

INSERT INTO Orderitem VALUES ('229','129','S','20','12-MAY-2021');

INSERT INTO Orderitem VALUES ('232','131','l','18','12-JUN-2021');

INSERT INTO Orderitem VALUES('235','135','l','17','12-NOV-2021');

PHOTOGRAPHER:

INSERT INTO Photographer VALUES ('01','daniel','thomas','east st',

'NYC','NY','10127','585','125455');

INSERT INTO Photographer VALUES ('02','ronald','lara','hotwell st',

'SEATTLE','CA','10252','225','252565');

JOB 1:

INSERT INTO Job1 VALUES('5','wed','W','NY','11-FEB-2013','2','298','1','a');

INSERT INTO Job1 VALUES('9','BIR','K','JC','12-MAR-2020','3','343','2','b');

INSERT INTO Job1 VALUES('8','EVNT','P','SM','11-JUN-2021','5','455','1','d');

INSERT INTO Job1 VALUES('7','wed','L','RI','12-JUN-2021','5','101','1','a');

INSERT INTO Job1 VALUES('2','wed','E','LA','21-JUL-2021','7','903','2','f');

JOB 2:

INSERT INTO Job2 VALUES ('11','300','200','100');

INSERT INTO Job2 VALUES ('12', '400','200','200');

INSERT INTO Job2 VALUES ('13', '300','150','150');

INSERT INTO Job2 VALUES ('14', '250','200','50');

INSERT INTO Job2 VALUES ('15', '300','200','100');

PAYMENT

INSERT INTO payment VALUES('22','5','12-FEB-2013','cash','200');

INSERT INTO payment VALUES('34','9','11-MAR-2020','cash','200');

INSERT INTO payment VALUES('32','8','11-JUN-2021','card','150');

INSERT INTO payment VALUES('45','7','12-JUN-2021','cash','50');

INSERT INTO payment VALUES('61','2','21-JUL-2021','card','100');

PROOF VALUES:

INSERT INTO Proof VALUES ('5','124','Eco');

INSERT INTO Proof VALUES ('9','126','Med');

INSERT INTO Proof VALUES ('8','129','Premium');

INSERT INTO Proof VALUES ('7','131','Eco');

INSERT INTO Proof VALUES ('2','135','Premium');

Text

Description automatically generated

Text

Description automatically generated with medium confidence

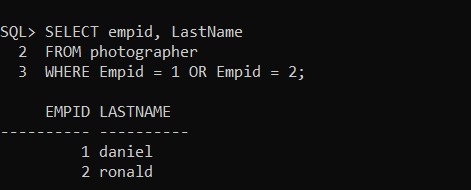
**Text

Description automatically generated with medium confidence**

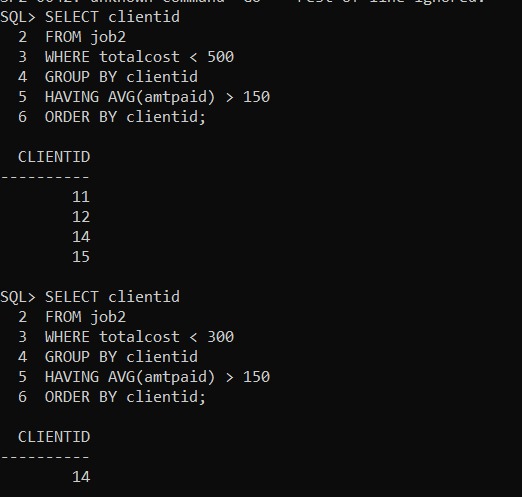
**STEP 6.7 - Design SQL statements that will process five non-routine requests for information from the database. Then execute the SQL statements in the database.** Show your work by providing screenshots of executing the SQL statements in the database along with the results.

Note: These five non-routine requests should be different from the ones you created in Chapter 5.

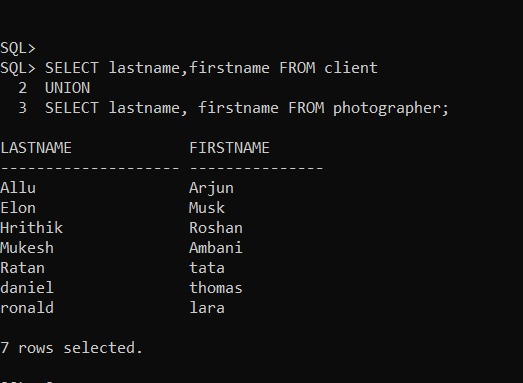
**1 . Generate a list of all employees with their id titles in the photographer table**



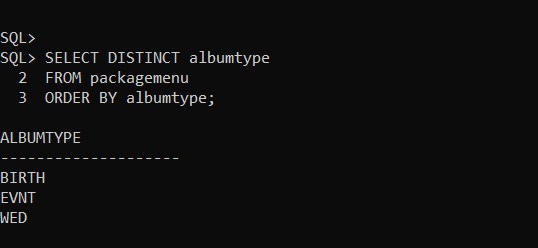
**2. Generate a list of all clientid depending on the amount they paid (total cost and amtpaid)**



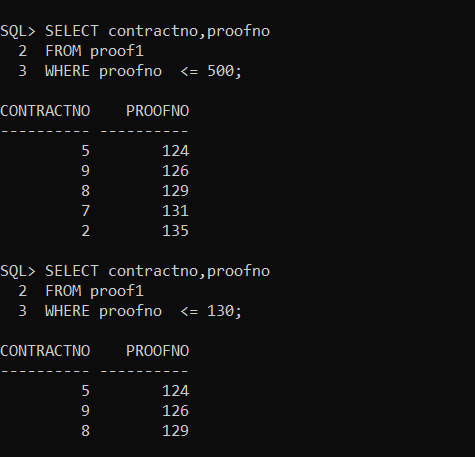
**3. generate a list of all the names in the both client and photographer tables present in the database**



**4 . Generate a list of all unique titles in the packagemenu**

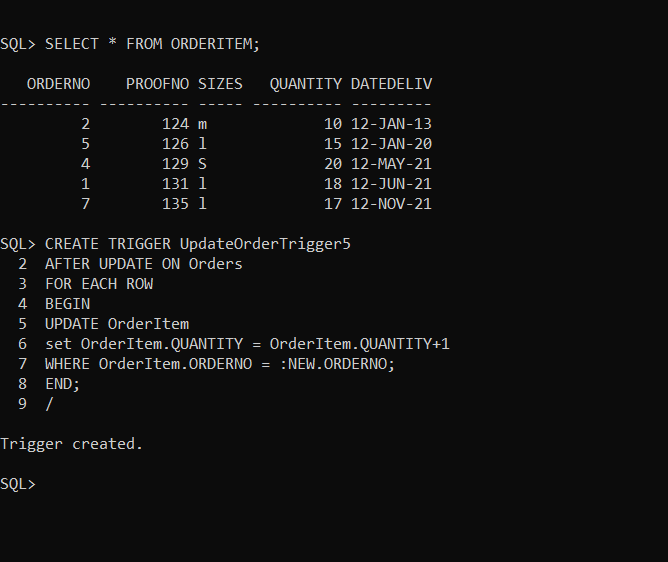


**5. Finding the proofnumber of the events for getting their contractnumber**



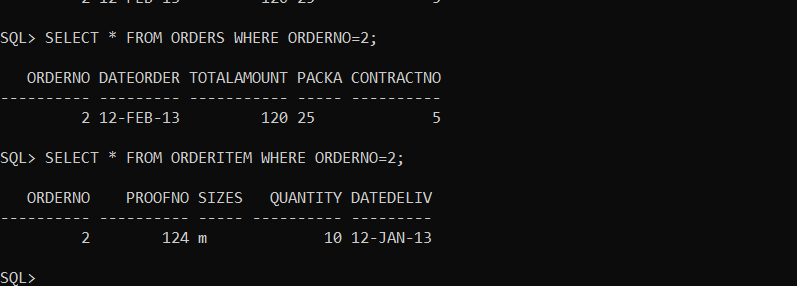
**Step 6.8 - Design one trigger for your project. Then create the trigger in the database.** Show your work by providing screenshots of creating the trigger in the database.

Note: This trigger should be different from the one you created in Chapter 5.



**Step 6.9 - Design and execute SQL statements to demonstrate that the trigger is working as expected.** To demonstrate that the trigger is working as expected, provide a screenshot of the data before and after the trigger is executed.

BEFORE:



AFTER:

