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 ER 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.

    - DonorCircle1 functional dependency: circleType its is unique and identifies itself
    - Donor1 functional dependency: is in BCNF
        - donorId{name, DOB, address, affiliation , preferredPhone, circleType} - primary key
        - circleType {circleType} - referenced table/foreign key
    - Pledge1 functional dependency: is in BCNF
        - PledgeId {donorId, pledgeAmount, pledgeData}
        - donorId{name, DOB, address, affiliation , preferredPhone, circleType} - foreign key
    - Payment1 functional dependency: is in BCNF
        - paymentId {pledgeId, amountPaid, paymentType, cardNumber, paymentTiming}
        - PledgeId {pledgeAmount}
    - Event1 functional dependency : is in BCNF
        - Event1 {eventName, eventDate, eventCost}
        - eventName {eventId}
    - eventParticipation functional dependency : is in BCNF
        - donorId and eventId {donationAtEvent, isDonor}- forms a composite prime and both attributes are required
    - MatchingGift1 functional dependency : is in BCNF
        - matchingGiftId {donorId, employerName, employerPhone, amount}-all depend on matching giftId

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

- Don’t need any update everything is on the dictionary
- Assumption
    - Id are assigned manually
    - Dates are stored as DATE
    - The constraint name include 1 suffix because i had some workspace collision

- 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.
    - donorCircle : circletype varchar, donor category
    - Donor: donorId number PK, name VARCHAR NOT NULL, DOB DATE, address VARCHAR NOT NULL, affiliation VARCHAR, preferredPhone VARCHAR, circleType VARCHAR
    - Pledge1: pledgeId NUMBER(8,2), donorId NUMBER(8,2) NOT NULL, pledgeAmount Number(10,2) CHECK (pledgeAmount > 0), pledgeDate DATE NOT NULL,
    - Payment: Payment1, paymentId VARCHAR2(8), pledgeId NUMBER(8,2) NOT NULL, amountPaid NUMBER(10,2), paymentType VARCHAR2(30), cardNumber VARCHAR2(16), paymentTiming VARCHAR2(30), CONSTRAINT Payment_paymentId_pk1 PRIMARY KEY (paymentID), CONSTRAINT Payment_pledgeId_fk1 Foreign KEY (pledgeId) REFERENCES Pledge1(pledgeId) ON DELETE SET NULL
    - Event1: eventId VARCHAR2(4) NOT NULL,   eventdonorId, name VARCHAR2(200) NOT NULL,   eventDate DATE NOT NULL, eventCost NUMBER(10,2) NOT NULL, CONSTRAINT Event_eventId_pk1 PRIMARY KEY (eventId), CONSTRAINT Event_eventdonorId, name_uk1 UNIQUE (eventdonorId, name)
    - EventParticipation1:   eventId VARCHAR2(4) NOT NULL,   donorId NUMBER(8,2) NOT NULL, donationAtEvent NUMBER, isDonor CHAR(1) DEFAULT 'Y' CHECK (isDonor IN ('Y', 'N'))
    - MatchingGift1:   matchingGiftId VARCHAR2(6) NOT NULL, donorId NUMBER(8,2) NOT NULL, employerdonorId, name VARCHAR2(50) NOT NULL,employerPhone VARCHAR2(13),amount Number(10,2) CHECK (amount >0),

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

```
CREATE TABLE DonorCircle1(
   circleType VARCHAR2(50),
   minAmount NUMBER(10,2),
   maxAmount NUMBER(10,2),
   CONSTRAINT DonorCircle1_pk PRIMARY KEY (circleType)
);

CREATE TABLE Donor1(
   donorId NUMBER GENERATED ALWAYS AS IDENTITY,
   name VARCHAR2(50) NOT NULL,
   DOB DATE,
   address VARCHAR2(100) NOT NULL,
   affiliation VARCHAR2(100),
   preferredPhone VARCHAR2(16),
   circleType VARCHAR2(50),
   CONSTRAINT Donor1_pk PRIMARY KEY (donorId),
   CONSTRAINT Donor1_circleType_fk FOREIGN KEY (circleType)
      REFERENCES DonorCircle1(circleType) ON DELETE SET NULL
);

CREATE TABLE Pledge1(
   pledgeId NUMBER GENERATED ALWAYS AS IDENTITY,
   donorId NUMBER(8) NOT NULL,
   pledgeAmount Number(10,2) CHECK (pledgeAmount > 0),
   pledgeDate DATE NOT NULL,
   CONSTRAINT Pledge1_pk PRIMARY KEY (pledgeID),
   CONSTRAINT Pledge1_donorId_fk FOREIGN KEY (donorId)
      REFERENCES Donor1(donorId) ON DELETE CASCADE
```

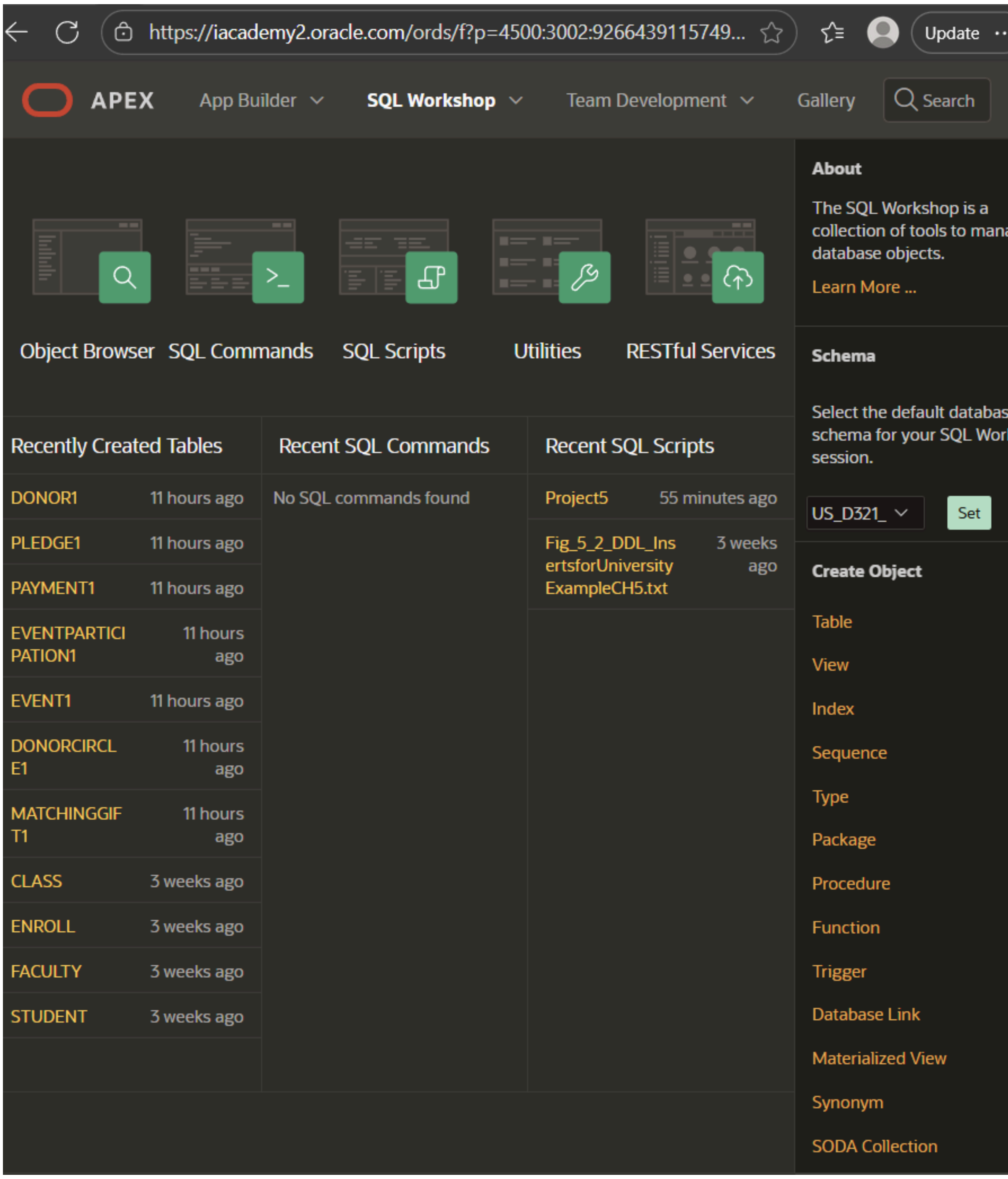
```
);

CREATE TABLE Payment1(
   paymentId NUMBER GENERATED ALWAYS AS IDENTITY,
   pledgeId NUMBER NOT NULL,
   amountPaid NUMBER(10,2) CHECK (amountPaid >= 0),
   paymentType VARCHAR2(30),
   cardNumber VARCHAR2(16),
   paymentTiming VARCHAR2(30),
   CONSTRAINT Payment1_pk PRIMARY KEY (paymentID),
   CONSTRAINT Payment1_pledgeId_fk FOREIGN KEY (pledgeId)
      REFERENCES Pledge1(pledgeId) ON DELETE CASCADE
);

CREATE TABLE Event1(
   eventId NUMBER GENERATED ALWAYS AS IDENTITY,
   eventName VARCHAR2(200) NOT NULL,
   eventDate DATE NOT NULL,
   eventCost NUMBER(10,2) CHECK (eventCost >= 0),
   CONSTRAINT Event1_pk PRIMARY KEY (eventId)
);

CREATE TABLE EventParticipation1(
   eventId NUMBER NOT NULL,
   donorId NUMBER NOT NULL,
   donationAtEvent NUMBER(10,2) CHECK(donationAtEvent > 0),
   isDonor CHAR(1) DEFAULT 'Y' CHECK (isDonor IN ('Y', 'N')),
   CONSTRAINT EventParticipation1_pk PRIMARY KEY (donorId, eventId),
   CONSTRAINT EventParticipation1_donorId_fk FOREIGN KEY (donorId)
      REFERENCES Donor1(donorId) ON DELETE CASCADE,
   CONSTRAINT EventParticipation1_eventId_fk FOREIGN KEY (eventId)
      REFERENCES Event1(eventId) ON DELETE CASCADE
);

CREATE TABLE MatchingGift1(
   matchingGiftId NUMBER GENERATED ALWAYS AS IDENTITY,
   donorId NUMBER NOT NULL,
   employerName VARCHAR2(50) NOT NULL,
   employerPhone VARCHAR2(13),
   amount Number(10,2) CHECK (amount > 0),
   CONSTRAINT MatchingGift1_pk PRIMARY KEY (matchingGiftId),
   CONSTRAINT MatchingGift1_donorId_fk FOREIGN KEY (donorId)
      REFERENCES Donor1(donorId) ON DELETE CASCADE
);
```


https://iacademy2.oracle.com/ords/f?p=4500:3002:9266439115749...

Recently Created Tables	
DONOR1	11 hours ago
PLEDGE1	11 hours ago
PAYMENT1	11 hours ago
EVENTPARTICIPATION1	11 hours ago
EVENT1	11 hours ago
DONORCIRCLE1	11 hours ago
MATCHINGGIFT1	11 hours ago
CLASS	3 weeks ago
ENROLL	3 weeks ago
FACULTY	3 weeks ago
STUDENT	3 weeks ago

Recent SQL Commands

No SQL commands found

Recent SQL Scripts	
Project5	55 minutes ago
Fig_5_2_DDL_InsertsforUniversityExampleCH5.txt	3 weeks ago

About

The SQL Workshop is a collection of tools to mana
database objects.

Learn More ...

Schema

Select the default databas
schema for your SQL Wor
session.

US_D321_ Set

Create Object

Table
View
Index
Sequence
Type
Package
Procedure
Function
Trigger
Database Link
Materialized View
Synonym
SODA Collection

- Step 6.5 Create indexes for foreign keys and any other columns that will be used most often for queries.
    - CREATE INDEX idx_donor1_circleType ON Donor1(circleType);
    - CREATE INDEX idx_pledge1_donorId ON pledge1(donorId);
    - CREATE INDEX idx_payment1_pledgeId ON payment1(pledgeId);
    - CREATE INDEX idx_event1_donorId ON event1(donorId);

- CREATE INDEX idx_eventParticipation_eventId ON eventParticipation(eventId);
- CREATE INDEX idx_matchingGift1_donorId ON matchingGift1(donorId);

- 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 DonorCircle1 VALUES ('Student', 0, NULL);

  INSERT INTO DonorCircle1 VALUES ('Alumni', 200, 487.99);

  INSERT INTO DonorCircle1 VALUES ('Friend', 199, 395.99);

  INSERT INTO Donor1(name, DOB, address, affiliation, preferredPhone, circleType)

  VALUES('Liya', TO_DATE('2025-07-12','YYYY-MM-DD'),'oak st','Alumni','099-111','Alumni');

  INSERT INTO Donor1(name, DOB, address, affiliation, preferredPhone, circleType)

  VALUES('Lisa', TO_DATE('2005-09-27','YYYY-MM-DD'), 'brook st','Friend','009-234','Friend');

  INSERT INTO Donor1(name, DOB, address, affiliation, preferredPhone, circleType)

  VALUES('Mark', TO_DATE('2003-03-13','YYYY-MM-DD'),'pine ave','Student','785-675','Student');

  INSERT INTO Donor1(name, DOB, address, affiliation, preferredPhone, circleType)

  VALUES('Patricia', TO_DATE('2007/07/17','YYYY-MM-DD'),'river st','Alumni','256-756','Alumni');

  INSERT INTO Donor1(name, DOB, address, affiliation, preferredPhone, circleType)

  VALUES('Maria', TO_DATE('2007/08/19','YYYY-MM-DD'),'hills rd','Friend','023-465','Friend');

  INSERT INTO Pledge1(donorId, pledgeAmount, pledgeDate)

  VALUES(1, 200, TO_DATE('2025-07-12','YYYY-MM-DD'));

  INSERT INTO Pledge1(donorId, pledgeAmount, pledgeDate)

  VALUES(2, 400, TO_DATE('2025-06-12','YYYY-MM-DD'));

  INSERT INTO Pledge1(donorId, pledgeAmount, pledgeDate)

  VALUES(3, 600, TO_DATE('2025-05-12','YYYY-MM-DD'));

  INSERT INTO Pledge1(donorId, pledgeAmount, pledgeDate)

  VALUES(4, 800, TO_DATE('2025-03-12','YYYY-MM-DD'));

  INSERT INTO Pledge1(donorId, pledgeAmount, pledgeDate)

```
VALUES(5, 900, TO_DATE('2025-01-12','YYYY-MM-DD'));


INSERT INTO Payment1 (pledgeId, amountPaid, paymentType, cardNumber, paymentTiming)
VALUES (1, 100.00, 'Credit', '123456789012', 'Monthly');
INSERT INTO Payment1 (pledgeId, amountPaid, paymentType, cardNumber, paymentTiming)
VALUES (2, 250.00, 'Cash', '434868338357', 'One-time');
INSERT INTO Payment1 (pledgeId, amountPaid, paymentType, cardNumber, paymentTiming)
VALUES (3, 150.00, 'Debit', '987654321012', 'One-time');
INSERT INTO Payment1 (pledgeId, amountPaid, paymentType, cardNumber, paymentTiming)
VALUES (4, 100.00, 'Credit', '555555555555', 'Monthly');
INSERT INTO Payment1 (pledgeId, amountPaid, paymentType, cardNumber, paymentTiming)
VALUES (5, 300.00, 'Cash', '376493902556', 'One-time');


INSERT INTO Event1 (eventName, eventDate, eventCost)
VALUES ('Charity Gala', TO_DATE('2025-06-20','YYYY-MM-DD'), 50.00);
INSERT INTO Event1 (eventName, eventDate, eventCost)
VALUES ('Fundraiser Dinner', TO_DATE('2025-07-10','YYYY-MM-DD'), 75.00);
INSERT INTO Event1 (eventName, eventDate, eventCost)
VALUES ('Volunteer Fair', TO_DATE('2025-08-01','YYYY-MM-DD'), 1.00);
INSERT INTO Event1 (eventName, eventDate, eventCost)
VALUES ('Alumni Meetup', TO_DATE('2025-09-12','YYYY-MM-DD'), 25.00);
INSERT INTO Event1 (eventName, eventDate, eventCost)
VALUES ('Donor Appreciation Night', TO_DATE('2025-10-05','YYYY-MM-DD'), 100.00);


INSERT INTO EventParticipation1 VALUES (1, 1, 50.00, 'Y');
INSERT INTO EventParticipation1 VALUES (2, 2, 100.00, 'Y');
INSERT INTO EventParticipation1 VALUES (3, 3, 1.00, 'N');
INSERT INTO EventParticipation1 VALUES (4, 4, 20.00, 'Y');
INSERT INTO EventParticipation1 VALUES (5, 5, 150.00, 'Y');
```

INSERT INTO MatchingGift1 (donorId, employerName, employerPhone, amount)

VALUES (2, 'FoodService', '555-7777', 500.00);

INSERT INTO MatchingGift1 (donorId, employerName, employerPhone, amount)

VALUES (3, 'MedicalCenter', '555-8888', 150.00);

INSERT INTO MatchingGift1 (donorId, employerName, employerPhone, amount)

VALUES (5, 'EductaionPurpose', '555-9999', 300.00);

INSERT INTO MatchingGift1 (donorId, employerName, employerPhone, amount)

VALUES (1, 'StudentAid', '555-1212', 200.00);

INSERT INTO MatchingGift1 (donorId, employerName, employerPhone, amount)

VALUES (4, 'FoodDrive', '555-3434', 100.00);

- 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.
    - Total amount pais by every donor
        - SELECT d.name, SUM(p.amountPaid) AS totalPaid

          FROM Donor1 d

          JOIN Pledge1 p1 ON d.donorId = p1.donorID

          JOIN Payment1 p ON p1.pledgeId =p.pledgeId

          GROUP BY d.name;
    - Pledged more than 500
        - SELECT d.donorId, d.name , p.pledgeAmount

          FROM Donor1 d

          JOIN Pledge1 p ON d.donorId = p.donorId

          WHERE p.pledgeAmount >500;
    - Event and total amount donated during that event
        - SELECT e.eventName,

                    SUM(ep.donationAtEvent) AS totalDonations

          FROM Event1 e

          JOIN EventParticipation1 ep ON e.eventId= ep.eventId

GROUP BY e.eventName;

- Attend event but did not donate
    - SELECT d.donorId, d.name , e.eventName

      FROM EventParticipation1 ep

      JOIN Donor1 d ON ep.donorId =d.donorId

      JOIN Event1 e ON ep.eventId = e.eventId

      WHERE ep.isDonor = ‘N’
- Donors who made a matching gift contribution
    - SELECT d.name AS donorName, m.employerName

      FROM Donor1 d

      JOIN MatchingGift1 m ON d.donorId = m.donorId;