**Database Foundations**

**PRACTICE SESSION 2**

1. Transform the following entity, which includes a multivalued composite attribute, into a relational schema that shows the referential integrity constraints. Use the graphical notation. (Assume that Skill Code has a unique value for each skill).



**Answer**

EMPLOYEE

EmployeeID

EmployeeName

EMPLOYEESKILL

EmployeeID

SkillCode

SKILL

SkillCode

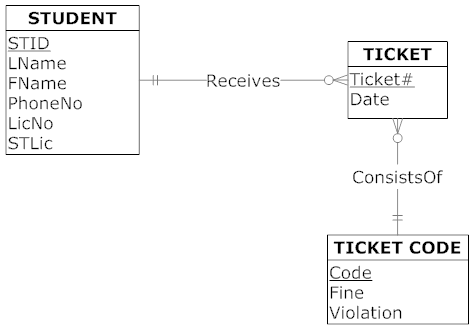
SkillTitle

SkillType

**Note:** First, “EMPLOYEE” and “SKILL” are mapped to relations. Since the association between “EMPLOYEE” and “SKILL” is many-to-many, we create a third relation named “EMPLOYEESKILL” to map this many-to-many relationship into two one-many relations.

There is another way to think about this solution. If you remember the steps in conceptual modeling (chapter 2), I mentioned in the class that “SKILL” could be modeled as an entity because it has its own attributes (see Figure 2.15b). Using the same logic, we could have modeled the relationship between “EMPLOYEE” and “SKILL” using two different entities in conceptual modeling. Now, we are converting this many-to-many relationship into two one-to-many relationships.

1. The public safety office at Millennium College maintains a list of parking tickets issued to vehicles parked illegally on the campus. The diagram below shows the ER model developed for data needs of this office. Transform this diagram into a relational schema that shows the referential integrity constraints. Use the graphical notation.



**Answer**

STID

LName

FName

PhoneNo

LicNo

STLic

Ticket#

Date

Code

STID

Code

Fine

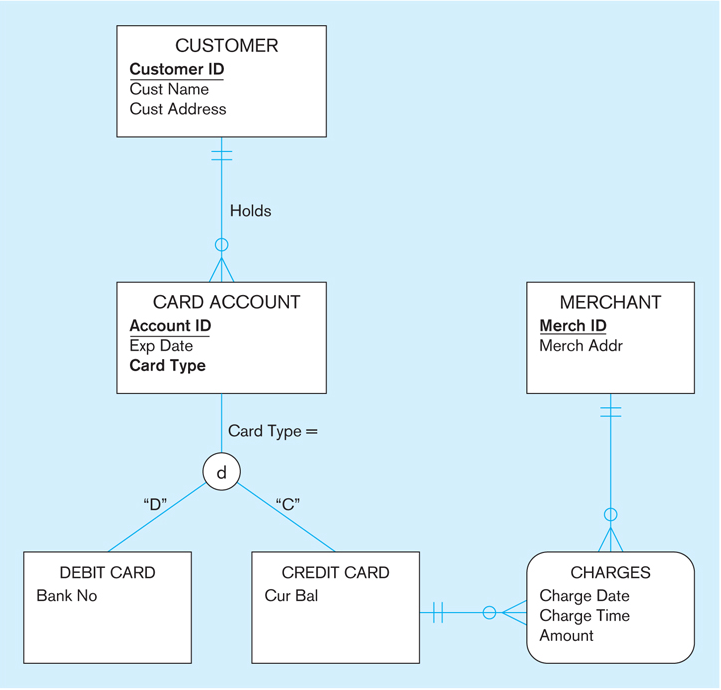
Violation

STUDENT

TICKET

TICKETCODE

1. The diagram below shows a simplified card environment. There are two types of card accounts: debit cards and credit cards. Credit card accounts accumulate charges with merchants. Each charge is identified by the date and time of the charge as well as the primary keys of merchant and credit card. Transform the diagram into a relational schema that shows referential integrity constraints. Use the graphical notation.



**Answer**

CustomerID

CustName

CustAddress

Csd

AccountID

ExpDate

CardType

CustomerID

DAccountID

BankCode

CAccountID

CurBal

MerchantID

CAccountID

ChargeDate

ChargeTime

Amount

MerchantID

MerchAddr

CARDACCOUNT

DEBITCARD

CREDITCARD

CHARGES

MERCHANT

CUSTOMER

1. Develop a relational schema for the following ER model. Show the referential integrity constraints. Use the graphical notation.



**Answer**

INVENTORY

STORE

QuantityOnHand

StoreID

PetID

StoreID

StoreName

PET

Cost

Price

PetDescription

PetID

PetName

SHIPMENT

DELIVERY

SupplierID

PetID

ShippingTime

DeliveryDate

PetID

StoreID

SUPPLIER

SupplierName

SupplierID

PURCHASE

CustomerID

StoreID

PetID

PurchaseDate

CUSTOMER

CustomerID

CustomerName

Exercises 5 through 13 are based on the class scheduling relations along with some sample data shown in the figure below. (Note: Data for the ASSIGNMENT relation, which represents a many-to-many association between FACULY and SECTION, are not shown in this figure.)



1. Create the relational schema for the class scheduling relations using the textual representation. Specify the referential integrity constraints.

**Answer:**

STUDENT(StudentID, StudentName)

FACULTY(FacultyID, FacultyName)

COURSE(CourseID, CourseName)

QUALIFIED(FacultyID, CourseID, DateQualified)

SECTION(SectionNo, Semester, CourseID)

REGISTRATION(StudentID, SectionNo)

ASSIGNMENT(FacultyID, SectionNo)

FacultyID in QUALIFIED references FacultyID in FACULTY

CourseID in QUALIFIED references CourseID in COURSE

CourseID in SECTION references CourseID in COURSE

StudentID in REGISTRATION references StudentID in STUDENT

SectionNo in REGISTRATION references SectionNo in SECTION

FacultyID in ASSIGNMENT references FacultyID in FACULTY

SectionNo in ASSIGNMENT references SectionNo in SECTION

1. Write a database description for each of the relations shown, using SQL DDL. Assume the following attribute data types:

StudentID (integer)

StudentName (25 characters (variable-length character string))

FacultyID (integer)

FacultyName (25 characters (variable-length character string))

CourseID (8 characters (fixed-length character string))

CourseName (30 characters (variable-length character string))

DateQualified (date)

SectionNo (integer)

Semester (7 characters (fixed length character string))

**Answer:**

Solutions are presented with the Oracle SQL 11g syntax that should work in most Oracle installations. Solutions may need to be modified for a particular Oracle installation or if you are using a different relational database management system (e.g., MS SQL) to illustrate the use of SQL commands. The solutions here use the naming conventions for tables, relations, attributes/columns, and views as shown in lectures. By default, Oracle database object names (e.g., table names, column names, etc.) are case-**insensitive.** On the other hand, string comparisons are case-**sensitive** (e.g., WHERE field='STRING' will only match columns where it's 'STRING').

**Note:** A particular SQL system may restrict the length of column names or may not permit embedded spaces in names. The following answer may not be acceptable to the SQL system you use, but you may modify the grammar accordingly.

Optionally, for numeric fields you can also specify a **precision** (total number of digits) and **scale** (number of digits to the right of the decimal point): NUMBER(precision, scale).

You can also use the data type INTEGER for numeric fields that do not contain floating points.

CREATE TABLE Student\_T

(StudentID NUMBER NOT NULL,

StudentName VARCHAR2(25),

CONSTRAINT Student\_PK PRIMARY KEY (StudentID));

CREATE TABLE Faculty\_T

(FacultyID NUMBER NOT NULL,

FacultyName VARCHAR2(25),

CONSTRAINT Faculty\_PK PRIMARY KEY (FacultyID));

CREATE TABLE Course\_T

(CourseID CHAR(8) NOT NULL,

CourseName VARCHAR2(30),

CONSTRAINT Course\_PK PRIMARY KEY (CourseID));

CREATE TABLE Section\_T

(SectionNo NUMBER NOT NULL,

Semester CHAR(7) NOT NULL,

CourseID CHAR(8) NOT NULL,

CONSTRAINT Section\_PK

PRIMARY KEY(SectionNo),

CONSTRAINT Section\_FK FOREIGN KEY (CourseID)

REFERENCES Course\_T (CourseID));

CREATE TABLE Qualified\_T

(FacultyID NUMBER NOT NULL,

CourseID CHAR(8) NOT NULL,

DateQualified DATE,

CONSTRAINT Qualified\_PK PRIMARY KEY (FacultyID,

CourseID),

CONSTRAINT QualifiedFaculty\_FK FOREIGN KEY (FacultyID) REFERENCES Faculty\_T (FacultyID),

CONSTRAINT QualifiedCourse\_FK FOREIGN KEY (CourseID) REFERENCES Course\_T (CourseID));

CREATE TABLE Registration\_T

(StudentID NUMBER NOT NULL,

SectionNo NUMBER NOT NULL,

CONSTRAINT Registration\_PK PRIMARY KEY (StudentID,

SectionNo),

CONSTRAINT RegistrationStudent\_FK

FOREIGN KEY(StudentID)

REFERENCES Student\_T(StudentID),

CONSTRAINT RegistrationSection\_FK

FOREIGN KEY (SectionNo)

REFERENCES Section\_T(SectionNo));

CREATE TABLE Assignment\_T

(FacultyID NUMBER NOT NULL,   
 SectionNo NUMBER NOT NULL,

CONSTRAINT Assignment\_PK PRIMARY KEY (FacultyID,

SectionNo),

CONSTRAINT AssignmentFaculty\_FK

FOREIGN KEY(FacultyID)

REFERENCES Faculty\_T(FacultyID),

CONSTRAINT AssignmentSection\_FK

FOREIGN KEY (SectionNo)

REFERENCES Section\_T(SectionNo));

1. Use SQL to create a view for the Student table

**Answer**

CREATE VIEW Student\_V AS

SELECT StudentID, StudentName FROM Student\_T;

or

CREATE VIEW Student\_V AS

SELECT \* FROM Student\_T;

1. Write SQL data definition commands for each of the following queries:
   1. How would you add an attribute, Class, (a variable length field up to 5 characters) to the Student table?
   2. How would you remove the Registration table?
   3. How would you change the FacultyName field from 25 characters to 40 characters?

**Answer**

1. ALTER TABLE Student\_T

ADD Class VARCHAR2(5);

1. DROP TABLE Registration\_T;
2. ALTER TABLE Faculty\_T

MODIFY FacultyName VARCHAR2(40);

1. Write SQL commands for the following:
2. Create two different forms of the INSERT command to add a student with a student ID of “65798” and name “Lopez” to the student table.
3. Now write a command that will remove this new student named Lopez from the student table
4. Create an SQL command that will modify the name of course ISM 4212 from “Database” to “Relational Databases”

**Answer**

1. INSERT INTO Student\_T (StudentID, StudentName)

VALUES (65798, 'Lopez');

INSERT INTO Student\_T VALUES (65798, 'Lopez');

1. DELETE FROM Student\_T WHERE StudentID = 65798;
2. UPDATE Course\_T

SET CourseName = 'Relational Databases'

WHERE CourseID = 'ISM 4212';

1. Write SQL queries to answer the following questions:
2. Which students have an ID number that is less than 50000?
3. What is the name of the faculty member whose ID is 4756?
4. What is the smallest section number used in the first semester of 2015?

**Answer**

* 1. SELECT StudentID, StudentName

FROM Student\_T

WHERE StudentID < 50000;

* 1. SELECT FacultyName

FROM Faculty\_T

WHERE FacultyID = 4756;

* 1. SELECT MIN(SectionNo)

FROM Section\_T

WHERE Semester = 'I-2015';

1. Write SQL queries to answer the following questions:
2. How many students are enrolled in Section 2714?
3. Which faculty members have qualified to teach a course since 2013 List the faculty ID, course ID, and date of qualification.

**Answer**

* 1. SELECT COUNT(\*)

FROM Registration\_T

WHERE SectionNo = 2714;

* 1. SELECT FacultyID, CourseID, DateQualified

FROM Qualified\_T

WHERE DateQualified >= '01-01-2013';

Note: You can also use '01/01/2013' format

1. Write SQL queries to answer the following questions:
2. Which students are enrolled in both Database and Networking courses in the second semester of 2015? (Hint: Use SectionNo for each course so that you can determine the answer from the registration table.)
3. Which instructors cannot teach both Syst Analysis and Syst Design?
4. Which courses were taught in the first semester of 2015 but not in the second semester of 2015?

**Answer**

1. We can only display Student IDs in the result table. The Database course is ISM 4212, which is Section 2714 in the Registration\_T table, and the Networking course is ISM 4930, which is Section 2715 in the Registration\_T table:

SELECT StudentID

FROM Registration\_T

WHERE SectionNo IN (2714, 2715)

GROUP BY StudentID

HAVING COUNT(\*) > 1;

1. In answering this question, we assume we are not interested in seeing those instructors who can teach neither course, but rather only those who can teach one but not the other course. To find those instructors who cannot teach either course requires SQL capabilities introduced in Chapter 7:

SELECT FacultyID

FROM Qualified\_T

WHERE CourseID IN ('ISM 3113', 'ISM 3112')

GROUP BY FacultyID

HAVING COUNT(\*) = 1;

1. Using Chapter 6 SQL tools, a user could use two single table queries over the SECTION table and then manually inspect the results to find the courses that are taught in first semester but not in the second semester. This manual inspection solution works moderately well when there are few rows in the tables. Another possible manual solution is to list all the courses sorted by CourseID and Semester in the SECTION table, and then manually inspect the entries to find those with only first semester offering. Again, this approach works moderately well if there are few rows in the tables.

Using a subquery approach, the following query could be used to find the requested information from the database:

SELECT CourseID, Semester

FROM Section\_T

WHERE Semester = 'I-2015'

AND CourseID NOT IN

(SELECT CourseID FROM Section\_T WHERE Semester = 'II-2015');

1. Write SQL queries to answer the following questions:
2. What are the courses included in the Section table? List each course only once.
3. List all students in an alphabetical order by StudentName.
4. List students who are enrolled in each course by student ID in an ascending order. Group the students by sections in which they are enrolled.
5. List the courses available. Group them by course prefix. (ISM is the only prefix shown but there are many others throughout the university.)

**Answer**

1. SELECT DISTINCT CourseID

FROM Section\_T;

1. SELECT StudentName

FROM Student\_T

ORDER BY StudentName;

1. SELECT SectionNo, StudentID

FROM Registration\_T

ORDER BY SectionNo, StudentID;

1. SELECT CourseID, CourseName

FROM Course\_T

ORDER BY CourseID;