**Use the contents of Figure 2.1 to work problems 1-3.**

1. **Write the business rule(s) that governs the relationship between AGENT and CUSTOMER.**

Given the data in the two tables, you can see that an AGENT – through AGENT\_CODE -- can occur many times in the CUSTOMER table. But each customer has only one agent. Therefore, the business rules may be written as follows:

One agent can have many customers.

Each customer has only one agent.

Given these business rules, you can conclude that there is a 1:M relationship between AGENT and CUSTOMER.

1. **Given the business rule(s) you wrote in Problem 1, create the basic Crow’s Foot ERD.**

The Crow’s Foot ERD is shown in Figure P2.2a.

**Figure P2.2a The Crow’s Foot ERD for Problem 3**



For discussion purposes, you might use the Chen model shown in Figure P2.2b. Compare the two representations of the business rules by noting the different ways in which connectivities (1,M) are represented. The Chen ERD is shown in Figure P2.2b.

**Figure P2.2b The Chen ERD for Problem 2**



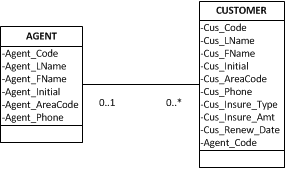
1. **Using the ERD you drew in Problem 2, create the equivalent Object representation and UML class diagram. (Use Figure 2.4 as your guide.)**

The OO model is shown in Figure P2.3.

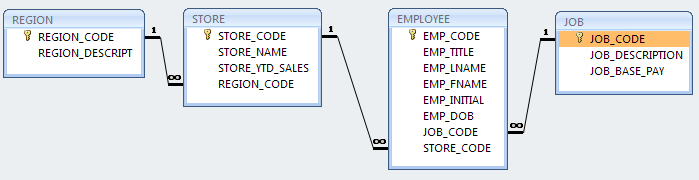
**Figure P2.3a The OO Model for Problem 3**



**Figure P.3b The UML Model for Problem 3**



**Using Figure P2.4 as your guide, work Problems 4–5. The DealCo relational diagram shows the initial entities and attributes for the DealCo stores, located in two regions of the country.**



**Figure P2.4 The DealCo relational diagram**

1. **Identify each relationship type and write all of the business rules.**

One region can be the location for many stores. Each store is located in only one region. Therefore, the relationship between REGION and STORE is 1:M.

Each store employs one or more employees. Each employee is employed by one store. (In this case, we are assuming that the business rule specifies that an employee cannot work in more than one store at a time.) Therefore, the relationship between STORE and EMPLOYEE is 1:M.

A job – such as accountant or sales representative -- can be assigned to many employees. (For example, one would reasonably assume that a store can have more than one sales representative. Therefore, the job title “Sales Representative” can be assigned to more than one employee at a time.) Each employee can have only one job assignment. (In this case, we are assuming that the business rule specifies that an employee cannot have more than one job assignment at a time.) Therefore, the relationship between JOB and EMPLOYEE is 1:M.

1. **Create the basic Crow’s Foot ERD for DealCo.**

The Crow’s Foot ERD is shown in Figure P2.5a.

**Figure P2.5a The Crow’s Foot ERD for DealCo**

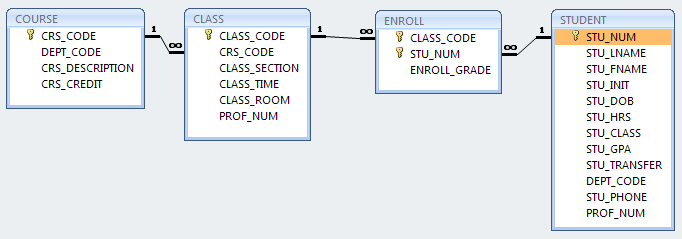


The Chen model is shown in Figure P2.5b. (Note that you always read the relationship from the “1” to the “M” side.)

**Figure P2.5b The Chen ERD for DealCo**



**Using Figure P2.6 as your guide, work Problems 6−8 The Tiny College relational diagram shows the initial entities and attributes for Tiny College.**



**Figure P2.6 The Tiny College relational diagram**

**Use the database shown in Figure P3.1 to answer Problems 1-9.**

**FIGURE P3.1 The Ch03\_StoreCo Database Tables**



1. **For each table, identify the primary key and the foreign key(s). If a table does not have a foreign key, write *None* in the space provided.**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **PRIMARY KEY** | **FOREIGN KEY(S)** |
| EMPLOYEE | EMP\_CODE | STORE\_CODE |
| STORE | STORE\_CODE | REGION\_CODE, EMP\_CODE |
| REGION | REGION\_CODE | NONE |

1. **Do the tables exhibit entity integrity? Answer yes or no and then explain your answer.**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **ENTITY INTEGRITY** | **EXPLANATION** |
| EMPLOYEE | Yes | Each EMP\_CODE value is unique and there are no nulls. |
| STORE | Yes | Each STORE\_CODE value is unique and there are no nulls. |
| REGION | Yes | Each REGION\_CODE value is unique and there are no nulls. |

1. **Do the tables exhibit referential integrity? Answer yes or no and then explain your answer. Write *NA* (Not Applicable) if the table does not have a foreign key.**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **REFERENTIAL INTEGRITY** | **EXPLANATION** |
| EMPLOYEE | Yes | Each STORE\_CODE value in EMPLOYEE points to an ***existing*** STORE\_CODE value in STORE. |
| STORE | Yes | Each REGION\_CODE value in STORE points to an ***existing*** REGION\_CODE value in REGION and each EMP\_CODE value in STORE points to an ***existing*** EMP\_CODE value in EMPLOYEE. |
| REGION | NA |  |

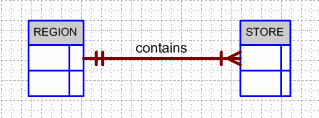
1. **Describe the type(s) of relationship(s) between STORE and REGION.**

Because REGION\_CODE values occur more than once in STORE, we may conclude that each REGION can contain many stores. But since each STORE is located in only one REGION, the relationship between STORE and REGION is M:1. (It is, of course, equally true that the relationship between REGION and STORE is 1:M.)

1. **Create the ERD to show the relationship between STORE and REGION.**

The Crow’s Foot ERD is shown in Figure P3.5. Note that each store is located in a single region, but that each region can have many stores located in it. (It’s always a good time to focus a discussion on the role of business rules in the creation of a database design.)

**Figure P3.5 ERD for the STORE and REGION Relationship**



**Problem #6 – Retrieving a subset of rows with multiple complex Boolean expressions**

Retrieve the student last name, student first name, and GPA for all students with a GPA (more than 2.2 and less than 2.7) OR (more than 3.2 and less than 3.8)

SELECT s.StdLastName, s.StdFirstName, s.StdGPA

FROM Student s

WHERE (s.StdGPA > 2.2 AND s.StdGPA < 2.7) OR (s.StdGPA > 3.2 AND s.StdGPA < 3.8)

ORDER BY 3 DESC, 1, 2

**Problem #7 – Retrieving a subset of rows with the BETWEEN operator**

Retrieve the student last name, student first name, and GPA for all students with a GPA that is between 2.7 and 3.2 inclusive

SELECT s.StdLastName, s.StdFirstName, s.StdGPA

FROM Student s

WHERE s.StdGPA BETWEEN 2.7 AND 3.2

ORDER BY 3 DESC, 1, 2

**Problem #8 – Retrieving a subset of rows with testing for NULLs**

Retrieve the offer number, course number, year, and faculty number from all course offerings that has not yet been assigned a Faculty

Repeat query for course offerings that have been assigned a Faculty

SELECT o.OfferNo, o.CourseNo, o.OffYear, o.FacNo

FROM Offering o

WHERE o.FacNo IS NULL

ORDER BY 1,2,3

SELECT o.OfferNo, o.CourseNo, o.OffYear, o.FacNo

FROM Offering o

WHERE o.FacNo IS NOT NULL

ORDER BY 1,2,3

**Problem #9 – Retrieving a subset of rows with testing for an exact string and inexact string**

Retrieve the offer number, course number, location, year, and faculty number from all course offerings in location BLM302

Retrieve the offer number, course number, location, year, and faculty number from all course offerings in location BLM 3rd floor

SELECT o.OfferNo, o.CourseNo, o.OffLocation, o.OffYear, o.FacNo

FROM Offering o

WHERE o.OffLocation= 'BLM302'

ORDER BY 1,2,3,4

SELECT o.OfferNo, o.CourseNo, o.OffLocation, o.OffYear, o.FacNo

FROM Offering o

WHERE o.OffLocation like 'BLM%'

ORDER BY 1,2,3,4

**Problem #10 – Using a derived column in both the column list and the WHERE clause**

Retrieve the student last name, student first name, and GPA plus 10% for all students with GPA plus 10% greater than 3

SELECT s.StdLastName, s.StdFirstName, (s.StdGPA\* 1.1) AS GPABoost

FROM Student s

WHERE (s.StdGPA\* 1.1) > 3

ORDER BY 3 DESC, 1, 2

**Problem #11 – Retrieving the number of rows from all of our tables**

For each of our tables, retrieve the number of rows

Tables are Student, Faculty, Offering, Course, and Enrollment

(omit sorting, table aliases, and column aliases)

SELECT COUNT(\*)

FROM Student

SELECT COUNT(\*)

FROM Faculty

SELECT COUNT(\*)

FROM Offering

SELECT COUNT(\*)

FROM Course

SELECT COUNT(\*)

FROM Enrollment

**Problem #12 – Examining the effect of NULL values on aggregate functions**

Retrieve the number of rows in the Faculty table using

COUNT(\*)

COUNT(f.FacSupervisor)

How many rows does each one return? Why?

SELECT COUNT(\*) AS NumFaculty

FROM Faculty

•returns 6 rows

SELECT COUNT(f.FacSupervisor) AS NumFaculty

FROM Faculty f

•returns 4 rows due to 2 NULL values

**Problem #13 – Aggregates on all rows of a table**

Retrieve the average GPA for all students

SELECT AVG(s.StdGPA) AS AverageGPA

FROM Student s

**Problem #14 – Aggregates on a subset of rows of a table (using a WHERE clause)**

Retrieve the minimum GPA, maximum GPA, average GPA, and average GPA plus 10% for freshman students

SELECT MIN(s.StdGPA) AS MinGPA, MAX(s.StdGPA) AS MaxGPA,

AVG(s.StdGPA) AS AvgGPA, AVG(s.StdGPA) \* 1.1 AS BoostAverageGPA

FROM Student s

WHERE s.StdClass= 'FR'

**Problem #15 – Aggregates on a group of rows (using a GROUP BY clause)**

Retrieve the class name, minimum GPA, maximum GPA, average GPA, and average GPA plus 10% for each class

SELECT s.StdCLass, MIN(s.StdGPA) AS MinGPA, MAX(s.StdGPA) AS MaxGPA, AVG(s.StdGPA) AS AvgGPA, AVG(s.StdGPA) \* 1.1 AS BoostAverageGPA

FROM Student s

GROUP BY s.StdClass

ORDER BY 1