Answers to Review Questions

1. What two conditions must be met before an entity can be classified as a weak entity? Give an example of a weak entity.

To be classified as a weak entity, **two conditions must be met**:
1. The entity must be existence-dependent on its parent entity.
2. The entity must inherit at least part of its primary key from its parent entity.

For example, the (strong) relationship depicted in the text's Figure 4.10 shows a weak CLASS entity:
1. CLASS is clearly existence-dependent on COURSE. (You can't have a database class unless a database course exists.)
2. The CLASS entity's PK is defined through the combination of CLASS_SECTION and CRS_CODE. The CRS_CODE attribute is also the PK of COURSE.

The conditions that define a weak entity are the same as those for a strong relationship between an entity and its parent. In short, the existence of a weak entity produces a strong relationship. And if the entity is strong, its relationship to the other entity is weak. (Note the solid relationship line in the text's Figure 4.10.)

Keep in mind that whether or not an entity is weak usually depends on the database designer's decisions. For instance, if the database designer had decided to use a single-attribute as shown in the text's Figure 4.8, the CLASS entity would be strong. (The CLASS entity's PK is CLASS_CODE, which is not derived from the COURSE entity.) In this case, the relationship between COURSE and CLASS is weak. (Note the dashed relationship line in the text's Figure 4.8.) **However, regardless of how the designer classifies the relationship – weak or strong – CLASS is always existence-dependent on COURSE**.

2. What is a strong (or identifying) relationship, and how is it depicted in a Crow's Foot ERD?

A strong relationship exists when en entity is existence-dependent on another entity and inherits at least part of its primary key from that entity. The Visio Professional software shows the strong relationship as a solid line. In other words, a strong relationship exists when a weak entity is related to its parent entity. (Note the discussion in question 1.)

3. **Given the business rule "an employee may have many degrees," discuss its effect on attributes, entities, and relationships. (Hint: Remember what a multivalued attribute is and how it might be implemented.)**

Suppose that an employee has the following degrees: BA, BS, and MBA. These degrees could be stored in a single string as a multivalued attribute named EMP_DEGREE in an EMPLOYEE table such as the one shown next:

EMP_NUM	EMP_LNAME	EMP_DEGREE
123	Carter	AA, BBA
124	O'Shanski	BBA, MBA, Ph.D.
125	Jones	AS
126	Ortez	BS, MS

Although the preceding solution has no obvious design flaws, it is likely to yield reporting problems. For example, suppose you want to get a count for all employees who have BBA degrees. You could, of course, do an "in-string" search to find all of the BBA values within the EMP_DEGREE strings. But such a solution is cumbersome from a reporting point of view. Query simplicity is a valuable thing to application developers – and to end users who like maximum query execution speeds. Database designers ought to pay some attention to the competing database interests that exist in the data environment.

One – very poor – solution is to create a field for each expected value. This "solution is shown next:

EMP_NUM	EMP_LNAME	EMP_DEGREE1	EMP_DEGREE2	EMP_DEGREE3
123	Carter	AA	BBA	
124	O'Shanski	BBA	MBA	Ph.D.
125	Jones	AS		
126	Ortez	BS	MS	

This "solution yields nulls for all employees who have fewer than three degrees. And if even one employee earns a fourth degree, the table structure must be altered to accommodate the new data value. (One piece of evidence of poor design is the need to alter table structures in response to the need to add data of an existing type.) In addition, the query simplicity is not enhanced by the fact that any degree can be listed in any column. For example, a BA degree might be listed in the second column, after an "associate of arts (AA) degree has been entered in EMP_DEGREE1. One might simplify the query environment by creating a set of attributes that define the data entry, thus producing the following results:

EMP_NUM	EMP_LNAME	EMP_AA	EMP_AS	EMP_BA	EMP_BS	EMP_BBA	EMP_MS	EMP_MBA	EMP_PhD
123	Carter	X				X			
124	O'Shanski					X		X	X
125	Jones		X						
126	Ortez				X		X		

This "solution" clearly proliferates the nulls at an ever-increasing pace.

The only reasonable solution is to create a new DEGREE entity that stores each degree in a separate record, this producing the following tables. (There is a 1:M relationship between EMPLOYEE and DEGREE. Note that the EMP_NUM can occur more than once in the DEGREE table. The DEGREE table's PK is EMP_NUM + DEGREE_CODE. This solution also makes it possible to record the date on which the degree was earned, the institution from which it was earned, and so on.

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME
123	Carter
124	O'Shanski
125	Jones
126	Ortez

Table name: DEGREE

EMP_NUM	DEGREE_CODE	DEGREE_DATE	DEGREE_PLACE
123	AA	May-1999	Lake Sumter CC
123	BBA	Aug-2004	U. of Georgia
124	BBA	Dec-1990	U. of Toledo
124	MBA	May-2001	U. of Michigan
124	Ph.D.	Dec-2005	U. of Tennessee
125	AS	Aug-2002	Valdosta State
126	BS	Dec-1989	U. of Missouri
126	MS	May-2002	U. of Florida

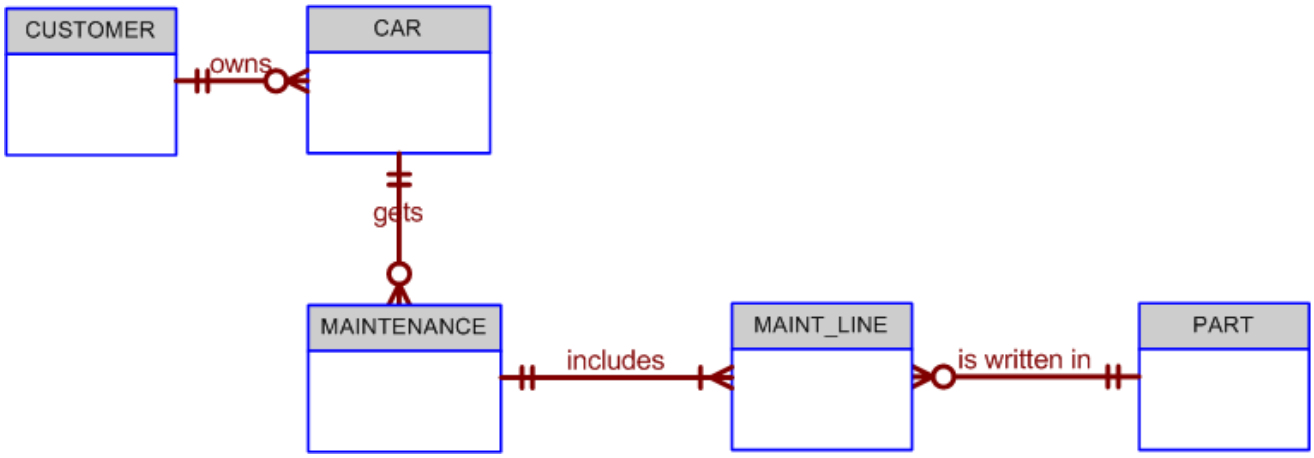
Note that this solution leaves no nulls, produces a simple query environment, and makes it unnecessary to alter the table structure when employees earn additional degrees. (You can make the environment even more flexible by naming the new entity QUALIFICATION, thus making it possible to store degrees, certifications, and other useful data that define an employee's qualifications.)

4. What is a composite entity, and when is it used?

A composite entity is generally used to transform M:N relationships into 1:M relationships. (Review the discussion that accompanied Figures IM4.3 through IM4.5.) A composite entity, also known as a bridge entity, is one that has a primary key composed of multiple attributes. The PK attributes are inherited from the entities that it relates to one another.

5. Suppose you are working within the framework of the conceptual model in Figure Q4.5.

Figure Q4.5 The Conceptual Model for Question 5



Given the conceptual model in Figure Q4.5:

a. Write the business rules that are reflected in it.

Even a simple ERD such as the one shown in Figure Q4.5 is based on many business rules. Make sure that each business rule is written on a separate line and that all of its details are spelled out. In this case, the business rules are derived from the ERD in a "reverse-engineering" procedure designed to document the database design. In a real world database design situation, the ERD is generated on the basis of business rules that are written before the first entity box is drawn. (Remember that the business rules are derived from a carefully and precisely written description of operations.)

Given the ERD shown in Figure Q4.5, you can identify the following business rules:

1. A customer can own many cars.
2. Some customers do not own cars.
3. A car is owned by one and only one customer.
4. A car may generate one or more maintenance records.
5. Each maintenance record is generated by one and only one car.
6. Some cars have not (yet) generated a maintenance procedure.
7. Each maintenance procedure can use many parts.
   (Comment: A maintenance procedure may include multiple maintenance actions, each one of which may or may not use parts. For example, 10,000-mile check may include the installation of a new oil filter and a new air filter. But tightening an alternator belt does not require a part.)
8. A part may be used in many maintenance records.
   (Comment: Each time an oil change is made, an oil filter is used. Therefore, many oil filters may be used during some period of time. Naturally, you are not using the same oil filter each time – but the part classified as "oil filter" shows up in many maintenance records as time passes.)

Note that the apparent M:N relationship between MAINTENANCE and PART has been resolved through the use of the composite entity named MAINT_LINE. The MAINT_LINE entity ensures that the M:N relationship between MAINTENANCE and PART has been broken up to produce the two 1:M relationships shown in business rules 9 and 10.

9. Each maintenance procedure generates one or more maintenance lines.
10. Each part may appear in many maintenance lines. (Review the comment in business rule 8.)

As you review the business rules 9 and 10, use the following two tables to show some sample data entries. For example, take a look at the (simplified) contents of the following MAINTENANCE and LINE tables and note that the MAINT_NUM 10001 occurs three times in the LINE table:

Sample MAINTENANCE Table Data

MAINT_NUM	MAINT_DATE
10001	15-Mar-2014
10002	15-Mar-2014
10003	16-Mar-2014

Sample LINE Table Data

MAINT_NUM	LINE_NUM	LINE_DESCRIPTION	LINE_PART	LINE_UNITS
10001	1	Replace fuel filter	FF-015	1
10001	2	Replace air filter	AF-1187	1
10001	3	Tighten alternator belt	NA	0
10002	1	Replace taillight bulbs	BU-2145	2
10003	1	Replace oil filter	OF-2113	1
10003	2	Replace air filter	AF-1187	1

b. Identify all of the cardinalities.

The Visio-generated Crow's Foot ERD, shown in Figure Q4.5, does not show cardinalities directly. Instead, the cardinalities are implied through the Crow's Foot symbols. You might write the cardinality (0,N) next to the MAINT_LINE entity in its relationship with the PART entity to indicate that a part might occur "N" times in the maintenance line entity or that it might never show up in the maintenance line entity. The latter case would occur if a given part has never been used in maintenance.

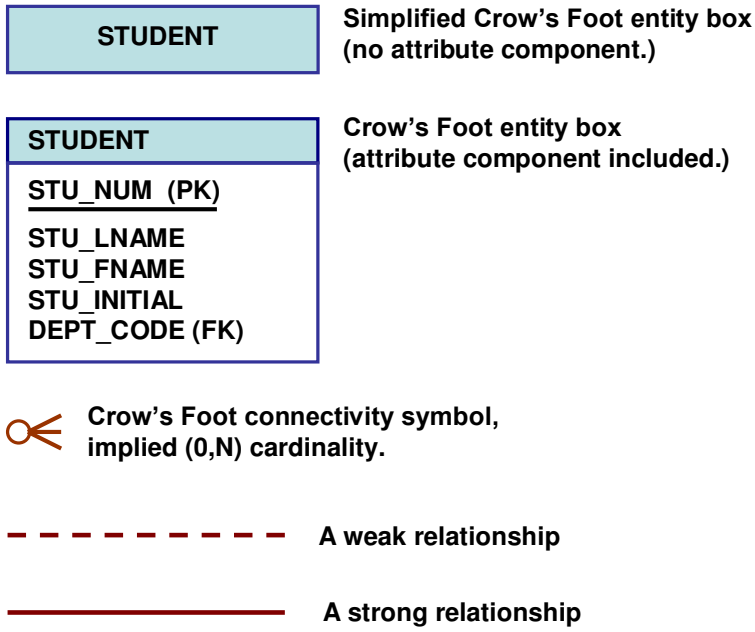
6. What is a recursive relationship? Given an example.

A recursive relationship exists when an entity is related to itself. For example, a COURSE may be a prerequisite to a COURSE. (See Section 4.1.10, "Recursive Relationships," for additional examples.

7. How would you (graphically) identify each of the following ERM components in a Crow's Foot model?

The answers to questions (a) through (d) are illustrated with the help of Figure Q4.7.

FIGURE Q4.7 Crow's Foot ERM Components



a. an entity

An entity is represented by a rectangle containing the entity name. (Remember that, in ER modeling, the word "entity" actually refers to the entity set.)

The Crow's Foot ERD – as represented in Visio Professional – does not distinguish among the various entity types such as weak entities and composite entities. Instead, the Crow's Foot ERD uses relationship types – strong or weak – to indicate the nature of the relationships between entities. For example, a strong relationship indicates the existence of a weak entity.

A composite entity is defined by the fact that at least one of the PK attributes is also a foreign key. Therefore, the Visio Crow's Foot ERD's composite and weak entities are not differentiated – whether or not an entity is weak or composite depends on the definition of the business rule(s) that describe the relationships. In any case, two conditions must be met before an entity can be classified as weak:
1. The entity must be existence-dependent on its parent entity
2. The entity must inherit at least part of its primary key from its parent entity.

b. the cardinality (0,N)

Cardinalities are implied through the use of Crow's Foot symbols. For example, note the implied (0,N) cardinality in Figure Q4.7.

c. a weak relationship

A weak relationship exists when the PK of the related entity does not contain at least one of the PK attributes of the parent entity. For example, if the PK of a COURSE entity is CRS_CODE and the PK of the related CLASS entity is CLASS_CODE, the relationship between COURSE and CLASS is weak. (Note that the CLASS PK does not include the CRS_CODE attribute.) A weak relationship is indicated by a dashed line in the (Visio) ERD.

d. a strong relationship

A strong relationship exists when the PK of the related entity contains at least one of the PK attributes of the parent entity. For example, if the PK of a COURSE entity is CRS_CODE and the PK of the related CLASS entity is CRS_CODE + CLASS_SECTION, the relationship between COURSE and CLASS is strong. (Note that the CLASS PK includes the CRS_CODE attribute.) A strong relationship is indicated by a solid line in the (Visio) ERD.

8. Discuss the difference between a composite key and a composite attribute. How would each be indicated in an ERD?

A composite key is one that consists of more than one attribute. If the ER diagram contains the attribute names for each of its entities, a composite key is indicated in the ER diagram by the fact that more than one attribute name is underlined to indicate its participation in the primary key.

A composite attribute is one that can be subdivided to yield meaningful attributes for each of its components. For example, the composite attribute CUS_NAME can be subdivided to yield the CUS_FNAME, CUS_INITIAL, and CUS_LNAME attributes. There is no ER convention that enables us to indicate that an attribute is a composite attribute.

9. What two courses of action are available to a designer when encountering a multivalued attribute?

The discussion that accompanies the answer to question 3 is valid as an answer to this question.

10. What is a derived attribute? Give an example.

A derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm. For example, an employee's age, EMP_AGE, may be found by computing the integer value of the difference between the current date and the EMP_DOB. If you use MS Access, you would use INT((DATE() – EMP_DOB)/365).
Similarly, a sales clerk's total gross pay may be computed by adding a computed sales commission

to base pay. For instance, if the sales clerk's commission is 1%, the gross pay may be computed by
EMP_GROSSPAY = INV_SALES*1.01 + EMP_BASEPAY
Or the invoice line item amount may be calculated by
LINE_TOTAL = LINE_UNITS*PROD_PRICE

11. How is a relationship between entities indicated in an ERD? Give an example, using the Crow's Foot notation.

Use Figure Q4.7 as the basis for your answer. Note the distinction between the dashed and solid relationship lines, then tie this distinction to the answers to question 7c and 7d.

12. Discuss two ways in which the 1:M relationship between COURSE and CLASS can be implemented. (Hint: Think about relationship strength.)

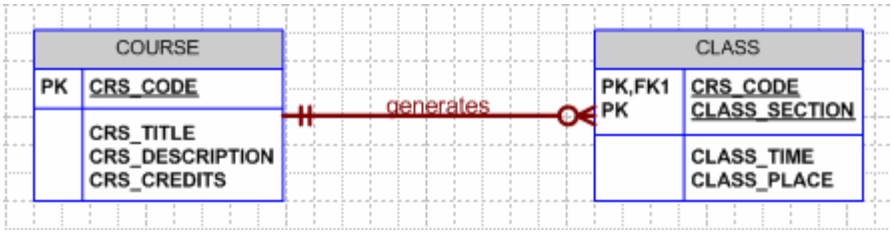
Note the discussion about weak and strong entities in questions 7c and 7d. Then follow up with this discussion:

The relationship is implemented as strong when the CLASS entity's PK contains the COURSE entity's PK. For example,

COURSE(**CRS_CODE**, CRS_TITLE, CRS_DESCRIPTION, CRS_CREDITS)
CLASS(**CRS_CODE**, **CLASS_SECTION**, CLASS_TIME, CLASS_PLACE)

Note that the CLASS entity's PK is CRS_CODE + CLASS_SECTION – and that the CRS_CODE component of this PK has been "borrowed" from the COURSE entity. (Because CLASS is existence-dependent on COURSE and uses a PK component from its parent (COURSE) entity, the CLASS entity is weak in this strong relationship between COURSE and CLASS. The Visio Crow's Foot ERD shows a strong relationship as a solid line. (See Figure Q4.12a.) Visio refers to a strong relationship as an identifying relationship.

Figure Q4.12a Strong COURSE and CLASS Relationship



Sample data are shown next:

Table name: COURSE

CRS_CODE	CRS_TITLE	CRS-DESCRIPTION	CRS_CREDITS
ACCT-211	Basic Accounting	An introduction to accounting. Required of all business majors.	3
CIS-380	Database Techniques I	Database design and implementation issues. Uses CASE tools to generate designs that are then implemented in a major database management system.	3
CIS-490	Database Techniques II	The second half of CIS-380. Basic Web database application development and management issues.	4

Table name: CLASS

CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_PLACE
ACCT-211	1	8:00 a.m. – 9:30 a.m. T-Th.	Business 325
ACCT-211	2	8:00 a.m. – 8:50 a.m. MWF	Business 325
ACCT-211	3	8:00 a.m. – 8:50 a.m. MWF	Business 402
CIS-380	1	11:00 a.m. – 11:50 a.m. MWF	Business 415
CIS-380	2	3:00 p.m. – 3:50 a.m. MWF	Business 398
CIS-490	1	1:00 p.m. – 3:00 p.m. MW	Business 398
CIS-490	2	6:00 p.m. – 10:00 p.m. Th.	Business 398

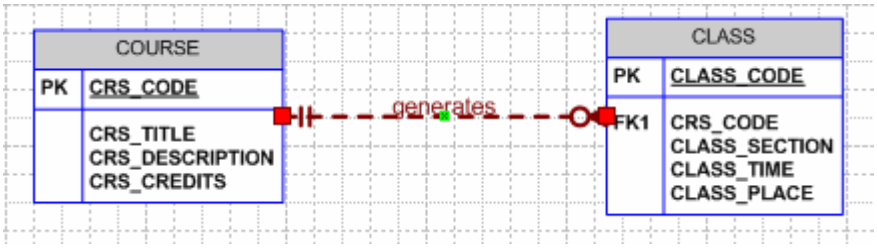
The relationship is implemented as weak when the CLASS entity's PK does not contain the COURSE entity's PK. For example,

COURSE(**CRS_CODE**, CRS_TITLE, CRS_DESCRIPTION, CRS_CREDITS)
CLASS(**CLASS_CODE**, CRS_CODE, CLASS_SECTION, CLASS_TIME, CLASS_PLACE)

(Note that CRS_CODE is no longer part of the CLASS PK, but that it continues to serve as the FK to COURSE.)

The Visio Crow's Foot ERD shows a weak relationship as a dashed line. (See Figure Q4.12b.) Visio refers to a weak relationship as a non-identifying relationship.

Figure Q4.12b Weak COURSE and CLASS Relationship



Given the weak relationship depicted in Figure Q4.13b, the CLASS table contents would look like this:

Table name: CLASS

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_PLACE
21151	ACCT-211	1	8:00 a.m. – 9:30 a.m. T-Th.	Business 325
21152	ACCT-211	2	8:00 a.m. – 8:50 a.m. MWF	Business 325
21153	ACCT-211	3	8:00 a.m. – 8:50 a.m. MWF	Business 402
38041	CIS-380	1	11:00 a.m. – 11:50 a.m. MWF	Business 415
38042	CIS-380	2	3:00 p.m. – 3:50 a.m. MWF	Business 398
49041	CIS-490	1	1:00 p.m. – 3:00 p.m. MW	Business 398
49042	CIS-490	2	6:00 p.m. – 10:00 p.m. Th.	Business 398

The advantage of the second CLASS entity version is that its PK can be referenced easily as a FK in another related entity such as ENROLL. Using a single-attribute PK makes implementation easier. This is especially true when the entity represents the "1" side in one or more relationships. **In general, it is advisable to avoid composite PKs whenever it is practical to do so**.

13. How is a composite entity represented in an ERD, and what is its function? Illustrate the Crow's Foot model.

The label "composite" is based on the fact that the composite entity contains at least the primary key attributes of each of the entities that are connected by it. The composite entity is an important component of the ER model because relational database models should not contain M:N relationships – and the composite entity can be used to break up such relationships into 1:M relationships.

Remind students to heed the advice given in the answer to the previous question. That is, **avoid composite PKs whenever it is practical to do so**. Note that the CLASS entity structure shown in Figure Q4.12b is far better than that of the CLASS entity structure shown in Figure Q4.12a. Suppose, for example, that you want to design a class enrollment entity to serve as the "bridge" between STUDENT and CLASS in the M:N relationship defined by these two business rules:

- A student can take many classes.
- Each class can be taken by many students.

In this case, you could create a (composite) entity named ENROLL to link CLASS and STUDENT, using these structures:

STUDENT(**STU_NUM**, STU_LNAME ……..)
ENROLL(**STU_NUM**, **CLASS_NUM**, ENROLL_GRADE ……)
CLASS(**CLASS_CODE**, CRS_CODE, CLASS_SECTION, CLASS_TIME, CLASS_PLACE)

Your students might argue that a composite PK in ENROLL does no harm, since it is not likely to be related to another entity in the typical academic database setting. Although that is a good observation, you would run into a problem in the event that might trigger a required relationship between ENROLL and another entity. In any case, you may simplify the creation of future

relationships if you create an "artificial" single-attribute PK such as ENROLL_NUM, while maintaining the STU_NUM and CLASS_NUM as FK attributes. In other words:

ENROLL(**ENROLL_NUM**, STU_NUM, CLASS_NUM, ENROLL_GRADE ……)

The ENROLL_NUM attribute values can easily be generated through the proper use of SQL code or application software, thus eliminating the need for data entry by humans.

The use of composite vs. single-attribute PKs is worth discussing. Composite PKs are frequently encountered in composite entities and your students will see that MS Visio will generate composite PKs automatically when you classify a relationship as strong. Composite PKs are not "wrong" is any sense, but minimizing their use does make the implementation of multiple relationships simple … Simple is generally a good thing!

NOTE

Because composite entities are frequently encountered in the real world environment, we continue to use them in the text and in many of our exercises and examples. However, the words of caution about their use should be repeated from time to time and you might ask your students to convert such composite entities.

Let's examine another example of the use of composite entities. Suppose that a trucking company keeps a log of its trucking operations to keep track of its driver/truck assignments. The company may assign any given truck to any given driver many times and, as time passes, each driver may be assigned to drive many of the company's trucks. Since this M:N relationship should not be implemented, we create the composite entity named LOG whose attributes are defined by the end-user information requirements. In this case, it may be useful to include LOG_DATE, TRUCK_NUM, DRIVER_NUM, LOG_TIME_OUT, and LOG_TIME_IN.

Note that the LOG's TRUCK_NUM and DRIVER_NUM attributes are the driver LOG's foreign keys. The TRUCK_NUM and DRIVER_NUM attribute values provide the bridge between the TRUCK and DRIVER, respectively. In other words, to form a proper bridge between TRUCK and DRIVER, the composite LOG entity must contain at least the primary keys of the entities connected by it.

You might think that the combination of the composite entity's foreign keys may be designated to be the composite entity's primary key. However, this combination will not produce unique values over time. For example, the same driver may drive a given truck on different dates. Adding the date to the PK attributes will solve that problem. But we still have a non-unique outcome when the same driver drives a given truck twice on the same date. Adding a time attribute will finally create a unique set of PK attribute values – but the PK is now composed of four attributes: TRUCK_NUM, DRIVER_NUM, LOG_DATE, and LOG_TIME_OUT. (The combination of these attributes yields a unique outcome, because the same driver cannot check out two trucks at the same time on a given date.)

Because multi-attribute PKs may be difficult to manage, it is often advisable to create an "artificial" single-attribute PK, such as LOG_NUM, to uniquely identify each record in the LOG table. (Access users can define such an attribute to be an "autonumber" to ensure that the system will generate unique LOG_NUM values for each record.) Note that this solution produces a LOG table that contains two candidate keys: the designated primary key and the combination of foreign keys that could have served as the primary key.

While the preceding solution simplifies the PK definition, it does not prevent the creation of duplicate records that merely have a different LOG_NUM value. Note, for example, the first two records in the following table:

LOG_NUM	LOG_DATE	TRUCK_NUM	DRIVER_NUM	LOG_TIME_OUT	LOG_TIME_IN
10015	12-Mar-2014	322453	1215	07:18 a.m.	04:23 p.m.
10016	12-Mar-2014	322453	1215	07:18 a.m.	04:23 p.m.
10017	12-Mar-2014	545567	1298	08:12 a.m.	09:15 p.m.

To avoid such duplicate records, **you can create a unique index** on TRUCK_NUM + DRIVER_NUM + LOG_DATE + LOG_TIME_OUT.

Composite entities may be named to reflect their component entities. For example, an employee may have several insurance policies (life, dental, accident, health, etc.) and each insurance policy may be held by many employees. This M:N relationship is converted to a set of two 1:M relationships, by creating a composite entity named EMP_INS. The EMP_INS entity must contain at least the primary key components of each of the two entities connected by it. How many additional attributes are kept in the composite entity depends on the end-user information requirements.

14. What three (often conflicting) database requirements must be addressed in database design?

Database design must reconcile the following requirements:

a. Design elegance requires that the design must adhere to design rules concerning nulls, derived attributes, redundancies, relationship types, and so on.
b. Information requirements are dictated by the end users
c. Operational (transaction) speed requirements are also dictated by the end users.

Clearly, an elegant database design that fails to address end user information requirements or one that forms the basis for an implementation whose use progresses at a snail's pace has little practical use.

15. Briefly, but precisely, explain the difference between single-valued attributes and simple attributes. Give an example of each.

A single -valued attribute is one that can have only one value. For example, a person has only one first name and only one social security number.

A simple attribute is one that cannot be decomposed into its component pieces. For example, a person's sex is classified as either M or F and there is no reasonable way to decompose M or F. Similarly, a person's first name cannot be decomposed into meaningful components. (In contrast, if a phone number includes the area code, it can be decomposed into the area code and the phone number. And a person's name may be decomposed into a first name, an initial, and a last name.)

Single-valued attributes are not necessarily simple. For example, an inventory code HWPRIJ23145 may refer to a classification scheme in which HW indicates Hardware, PR indicates Printer, IJ indicates Inkjet, and 23145 indicates an inventory control number. Therefore,

HWPRIJ23145 may be decomposed into its component parts... even though it is single-valued. To facilitate product tracking, manufacturing serial codes must be single-valued, but they may not be simple. For instance, the product serial number TNP5S2M231109154321 might be decomposed this way:

TN = state = Tennessee
P5 = plant number 5
S2 = shift 2
M23 = machine 23
11 = month, i.e., November
09 = day
154321 = time on a 24-hour clock, i.e., 15:43:21, or 3:43 p.m. plus 21 seconds.

16. What are multivalued attributes, and how can they be handled within the database design?

The answer to question 3 is just as valid as an answer to this question. You can augment that discussion with the following discussion:

As the name implies, multi-valued attributes may have many values. For example, a person's education may include a high school diploma, a 2-year college associate degree, a four-year college degree, a Master's degree, a Doctoral degree, and various professional certifications such as a Certified Public Accounting certificate or a Certified Data Processing Certificate.

There are basically three ways to handle multi-valued attributes -- and two of those three ways are bad:

1. Each of the possible outcomes is kept as a separate attribute within the table. This solution is undesirable for several reasons. First, the table would generate many nulls for those who had minimal educational attainments. Using the preceding example, a person with only a high school diploma would generate nulls for the 2-year college associate degree, the four-year college degree, the Master's degree, the Doctoral degree, and for each of the professional certifications. In addition, how many professional certification attributes should be maintained? If you store two professional certification attributes, you will generate a null for someone with only one professional certification and you'd generate two nulls for all persons without professional certifications. And suppose you have a person with five professional certifications? Would you create additional attributes, thus creating many more nulls in the table, or would you simply ignore the additional professional certifications, thereby losing information?

2. The educational attainments may be kept as a single, variable-length string or character field. This solution is undesirable because it becomes difficult to query the table. For example, even a simple question such as "how many employees have four-year college degrees?" requires string partitioning that is time-consuming at best. Of course, if there is no need to ever group employees by education, the variable-length string might be acceptable from a design point of view. However, as database designers we know that, sooner or later, information requirements are likely to grow, so the string storage is probably a bad idea from that perspective, too.

3. Finally, the most flexible way to deal with multi-valued attributes is to create a composite entity that links employees to education. By using the composite entity, there will never be a situation in which additional attributes must be created within the EMPLOYEE table to accommodate people with multiple certifications. In short, we eliminate the generation of nulls. In addition, we gain information flexibility because we can also store the details (date earned, place earned, etc.) for each of the educational attainments. The (simplified) structures might look like those in Figure Q4.16 A and B.

Figure Q4.16a The Ch04_Questions Database Tables

Database name: Ch04_Questions

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME
215	Smith
216	Romero
217	Randall
218	Aarden

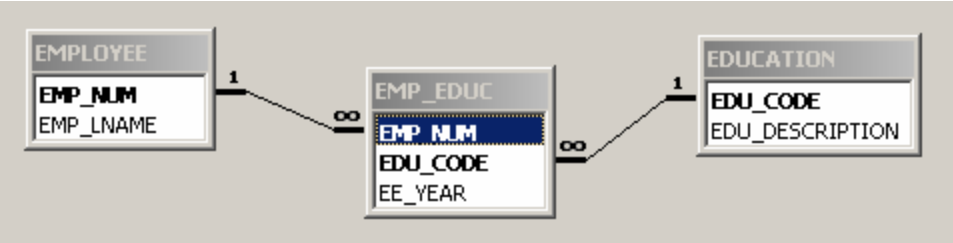
Table name: EDUCATION

EDU_CODE	EDU_DESCRIPTION
B4	Bachelor's degree
CDP	Certified data Processing
CNP	Certified Network Professional
CPA	Certified Public Accountant
DR	Earned Doctorate degree
HS	Highschool diploma
JC	Junior College degree
M2	Master's degree

Table name: EMP_EDUC

EMP_NUM	EDU_CODE	EE_YEAR
215	B4	2002
215	HS	1991
215	JC	1994
216	B4	1989
216	CDP	2004
216	CNP	2002
216	HS	1985
217	HS	1992
217	JC	2000
218	B4	1990
218	CDP	2000
218	CNP	2005
218	HS	1984
218	M2	1995

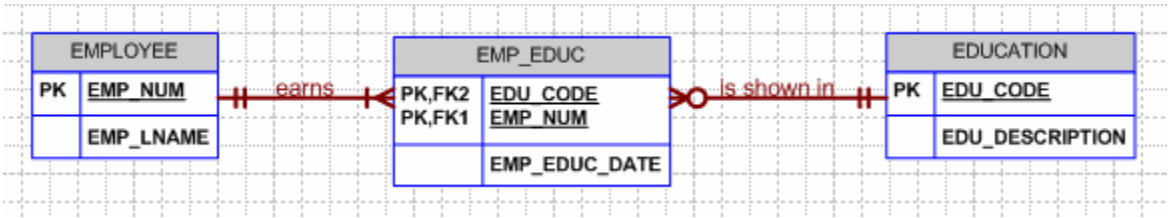
Figure Q4.16b The Ch04_Questions Relational Diagram



By looking at the structures shown in Figures Q4.16a and Q4.16b, we can tell that the employee named Romero earned a Bachelor's degree in 1989, a Certified Network Professional certification in 2002, and a Certified Data Processing certification in 2004. If Randall were to earn a Master's degree and a Certified Public Accountant certification later, we merely add another two records in the EMP_EDUC table. If additional educational attainments beyond those listed in the EDUCATION table are earned by any employee, all we need to do is add the appropriate record(s) to the EDUCATION table, then enter the employee's attainments in the EMP_EDUC table. There are no nulls, we have superb query capability, and we have flexibility. Not a bad set of design goals!

The database design on which Figures Q4.16a and Q4.16b are based is shown in Figure Q4.16c.

Figure Q4.16c The Crow's Foot ERD for the Ch04_Questions Database

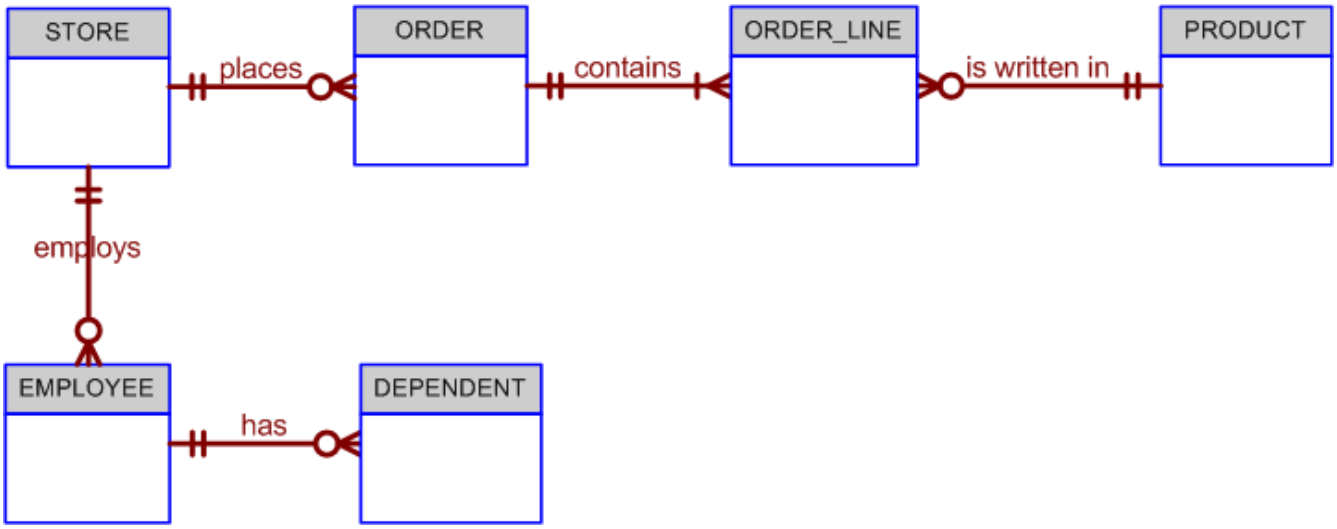


NOTE

Discuss with the students that the design in Figure Q4.16c shows that an employee must meet at least one educational requirement, because EMP_EDUC is not optional to EMPLOYEE. Thus each employee must appear at least once in the EMP_EDUC table. And, given this design, some of the educational attainments may not yet been earned by employees, because the design shows EMP_EDUC to be optional to EDUCATION. In other words, some of the EDUCATION records are not necessarily referenced by any employee. (In the original M:N relationship between EMPLOYEE and EDUCATION, EMPLOYEE must have been optional to EDUCATION.)

The final four questions are based on the ERD in Figure Q4.17.

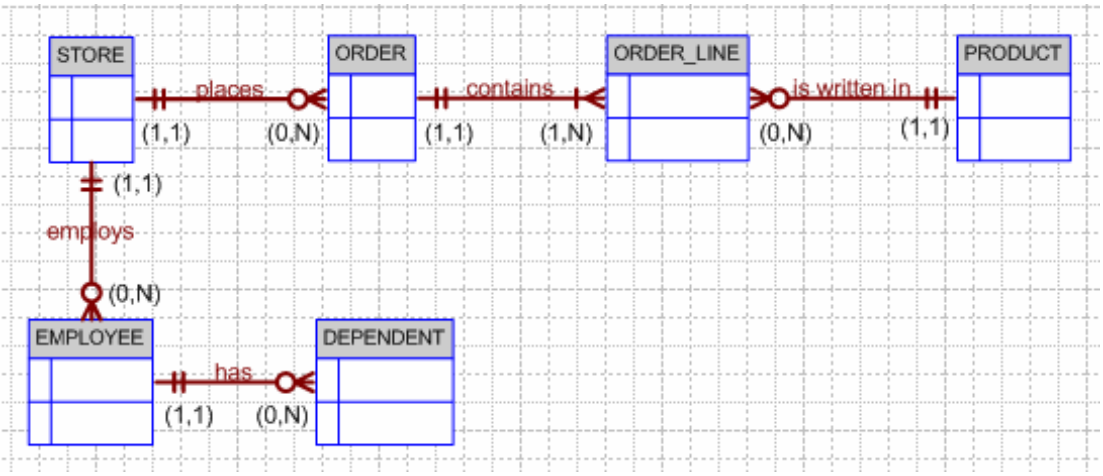
FIGURE Q4.17 The ERD For Questions 17–20



17. Write the ten cardinalities that are appropriate for this ERD.

The cardinalities are indicated in Figure Q4.17sol.

FIGURE Q4.17sol The Cardinalities



18. Write the business rules reflected in this ERD.

The following business rules are reflected in the ERD:

- A store may place many orders. (Note the use of "may" – which is reflected in the ORDER optionality.)
- An order must be placed by a store. (Note that STORE is mandatory to ORDER. In this ERD, the order environment apparently reflects a wholesale environment.)

- An order contains at least one order line. (Note that ORDER_LINE is mandatory to ORDER, and vice-versa.)
- Each order line is contained in one and only one order. (Discussion: Although a given item – such as a hammer – may be found in many orders, a specific hammer sold to a specific store is found in only one order.)
- Each order line has a specific product written in it.
- A product may be written in many orders. (Discussion: Many stores can order one or more specific products, but a product that is not in demand may never be sold to a store and will, therefore, not show up in any order line -- note that ORDER_LINE is optional to PRODUCT. Also, note that each order line may indicate more than one of a specific item. For example, the item may be "hammer" and the number sold may be 1 or 2, or 500. The ORDER_LINE entity would have at least the following attributes: ORDER_NUM, ORDLINE_NUM, PROD_CODE, ORDLINE_PRICE, ORDLINE_QUANTITY. The ORDER_LINE composite PK would be ORDER_NUM + ORDLINE_NUM. You might add the derived attribute ORDLINE_AMOUNT, which would be the result of multiplying ORDLINE_PRICE and ORDLINE_QUANTITY.)
- A store may employ many employees. (Discussion: A new store may not yet have any employees, yet the database may already include the new store information … location, type, and so on. If you made the EMPLOYEE entity mandatory to STORE, you would have to create an employee for that store before you had even hired one.)
- Each employee is employed by one (and only one) store.
- An employee may have one or more dependents. (Discussion: You cannot require an employee to have dependents, so DEPENDENT is optional to EMPLOYEE. Note the use of the word "may" in the relationship.)
- A dependent must be related to an employee. (Discussion: It makes no sense to keep track of dependents of people who are not even employees. Therefore, EMPLOYEE is mandatory to DEPENDENT.)

19. What two attributes must be contained in the composite entity between STORE and PRODUCT? Use proper terminology in your answer.

The composite entity must at least include the primary keys of the entities it references. The combination of these attributes may be designated to be the composite entity's (composite) primary key. Each of the (composite) primary key's attributes is a foreign key that references the entities for which the composite entity serves as a bridge.

As you discuss the model in Figure Q4.17sol, note that an order is represented by two entities, ORDER and ORDER_LINE. Note also that the STORE's 1:M relationship with ORDER and the ORDER's 1:M relationship with ORDER_LINE reflect the conceptual M:N relationship between STORE and PRODUCT. The original business rules probably read:

- A store can order many products
- A product can be ordered by many stores.

20. Describe precisely the composition of the DEPENDENT weak entity's primary key. Use proper terminology in your answer.

The DEPENDENT entity will have a composite PK that includes the EMPLOYEE entity's PK and one of its attributes. For example, if the EMPLOYEE entity's PK is EMP_NUM, the DEPENDENT entity's PK might be EMP_NUM + DEP_NUM.

21. The local city youth league needs a database system to help track children that sign up to play soccer. Data needs to be kept on each team and the children that will be playing on each team and their parents. Also, data needs to be kept on the coaches for each team. Draw the data model described below.

Entities required: Team, Player, Coach, and Parent.

Attributes required:

Team: Team ID number, Team name, and Team colors.

Player: Player ID number, Player first name, Player last name, and Player age.

Coach: Coach ID number, Coach first name, Coach last name, and Coach home phone number.

Parent: Parent ID number, Parent last name, Parent first name, Home phone number, and Home Address (Street, City, State, and ZIP Code).

The following relationships must be defined:

- **Team is related to Player.**
- **Team is related to Coach.**
- **Player is related to Parent.**

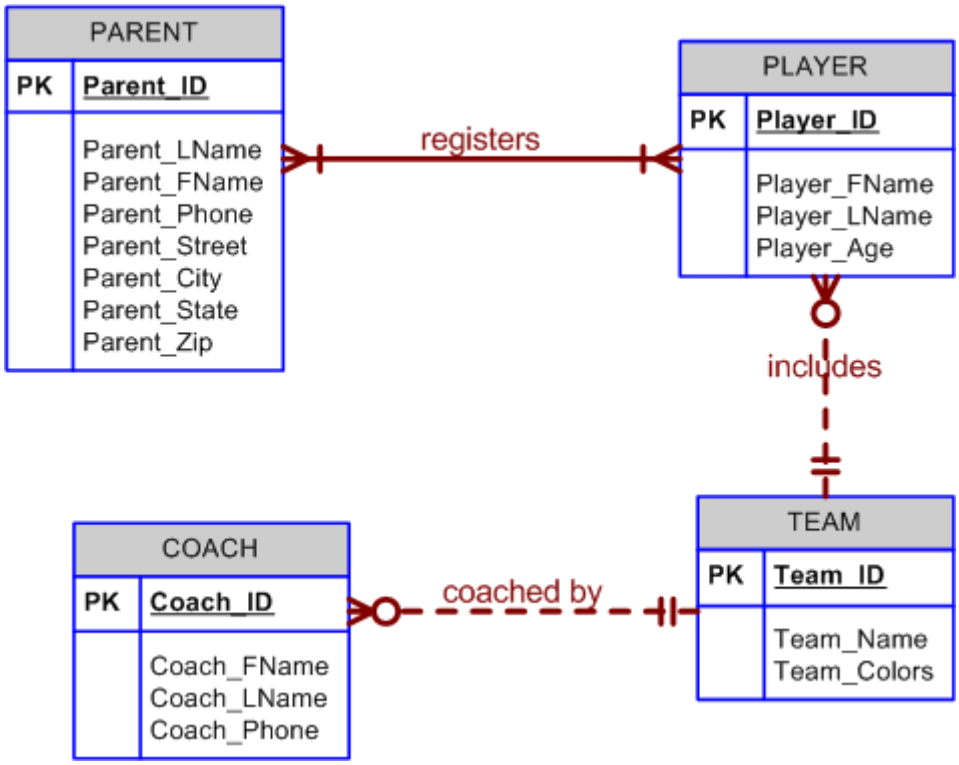
Connectivities and participations are defined as follows:

- **A Team may or may not have a Player.**
- **A Player must have a Team.**
- **A Team may have many Players.**
- **A Player has only one Team.**
- **A Team may or may not have a Coach.**
- **A Coach must have a Team.**
- **A Team may have many Coaches.**
- **A Coach has only one Team.**
- **A Player must have a Parent.**
- **A Parent must have a Player.**
- **A Player may have many Parents.**

- **A Parent may have many Players.**

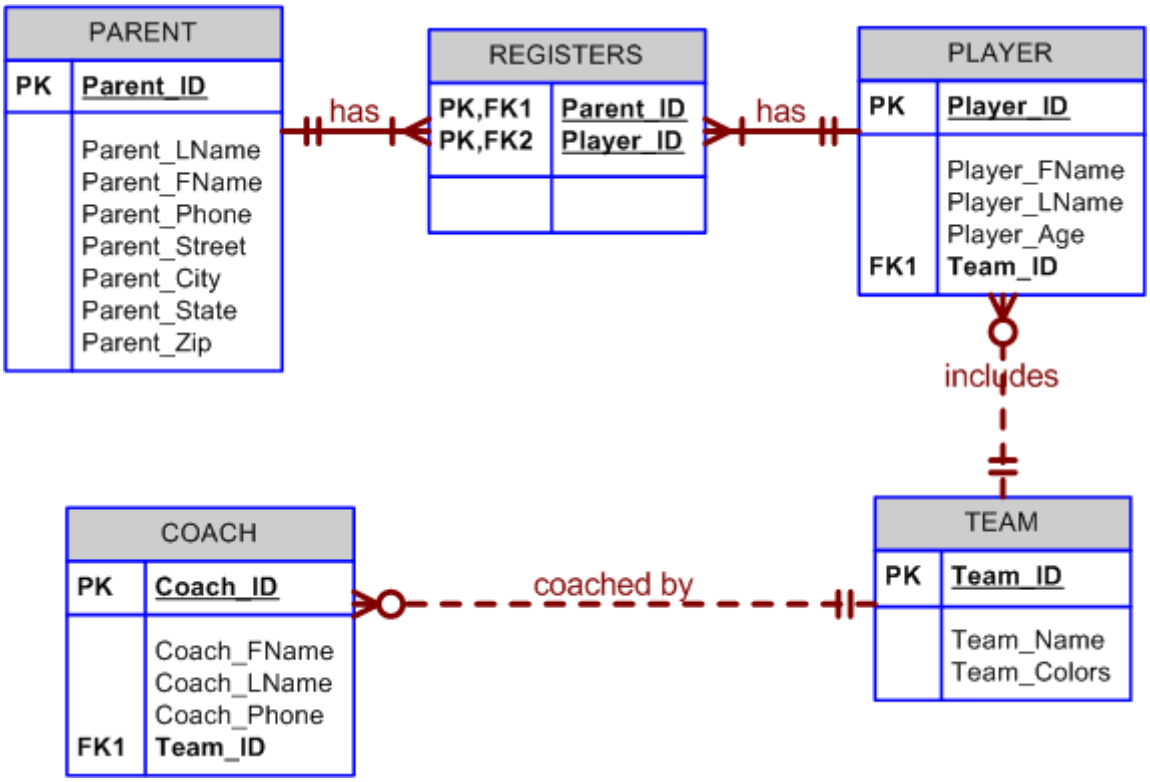
This is a great exercise in that it opens up possibilities for several discussion points. The conceptual ERD prior to placement of foreign keys and the resolution of the M:N relationship is shown in Figure Q4.21a.

FIGURE Q4.21a Conceptual ERD for Question 21



The most apparent issue that must be resolved is the M:N relationship. This is necessary so that foreign keys can be appropriately placed throughout the data model. The revised ERD with properly placed foreign keys is shown in Figure Q4.21b.

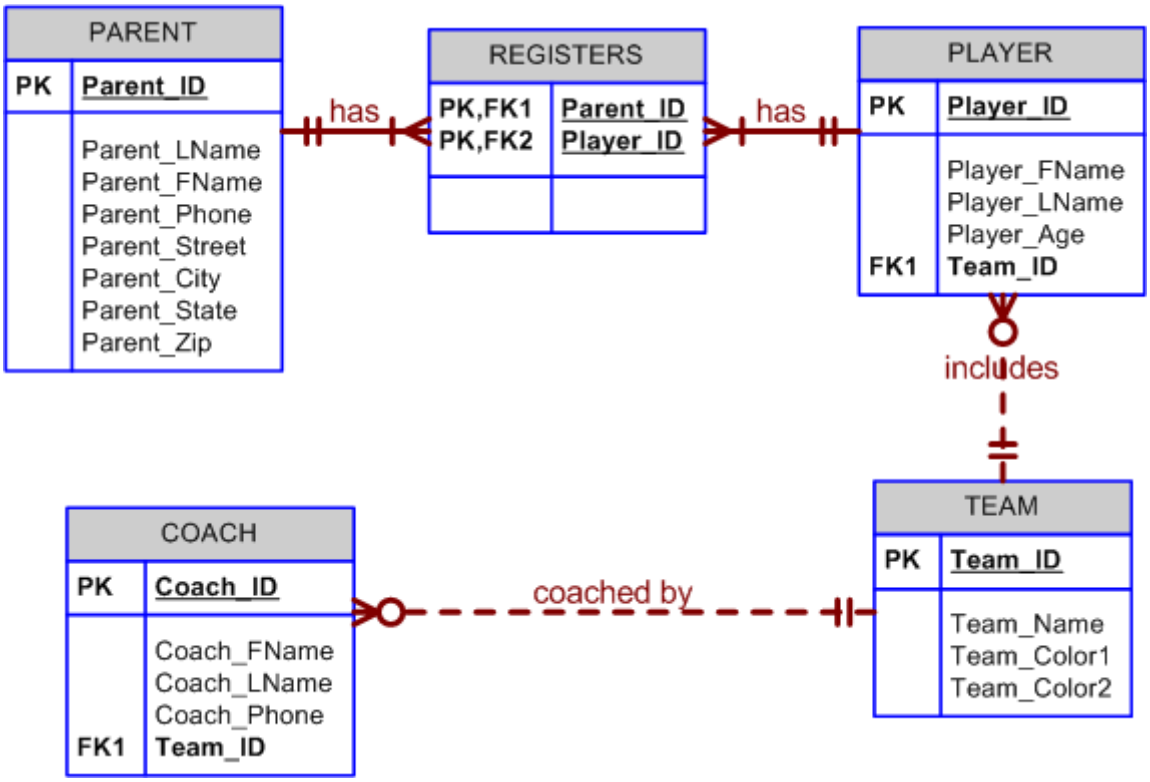
FIGURE Q4.21b ERD with foreign keys for Question 21



This solution, however, still leaves an interesting question about the Team_Colors attribute. What if teams have more than one color as is implied by the plural "colors" being used by the business users? Let's consider three options: 1) leave it as is (as if Team_Colors is a single-valued attribute), 2) create multiple attributes within the TEAM entity, or 3) create a new COLOR table.

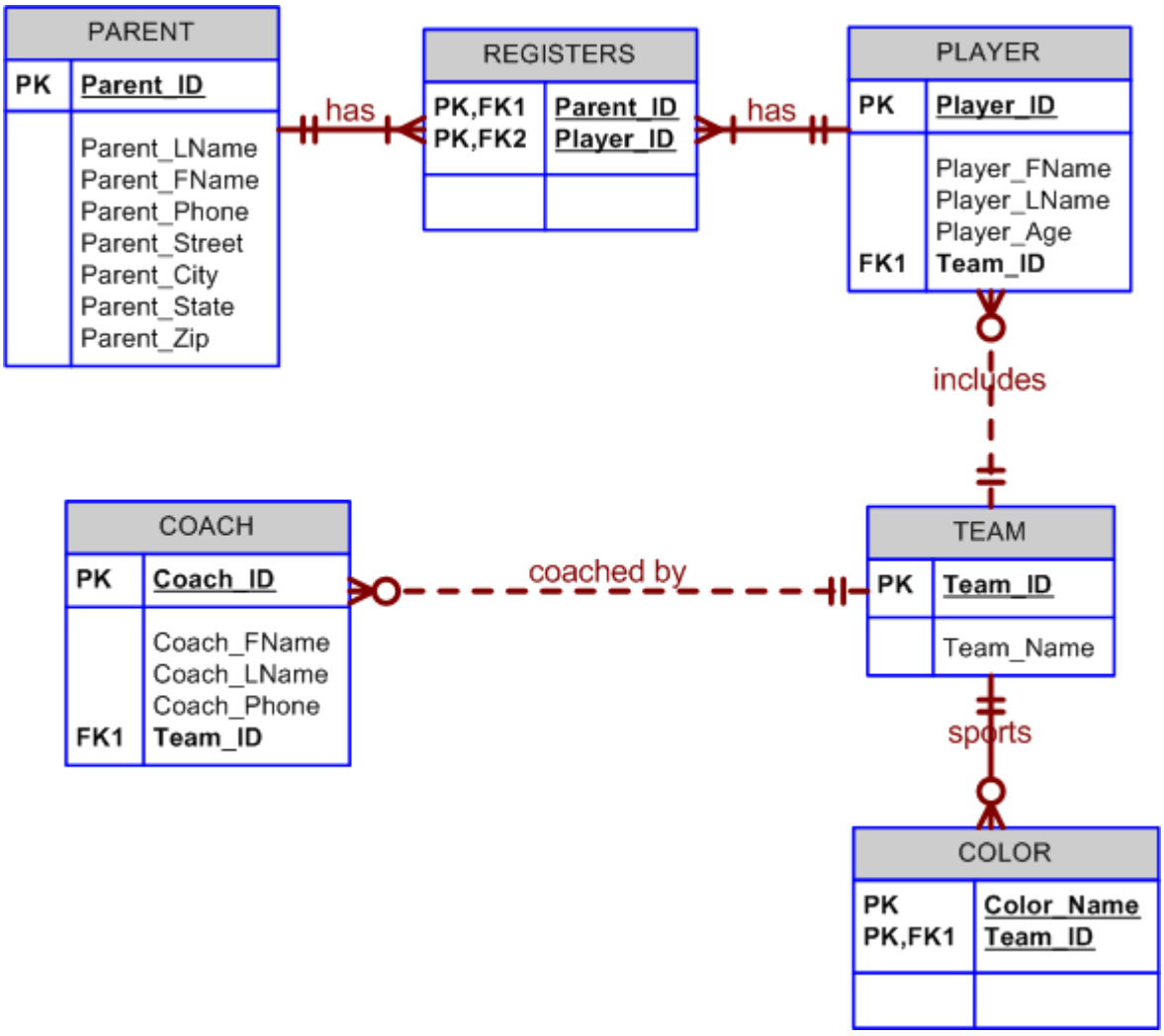
Team_Colors may be left as a single attribute if it is determined through discussion with the business users that they are not concerned with dealing with the different colors individually. For example, they will never be interested to know how many teams have the color Blue as one of their team colors, then we may choose to implement the design as given above. However, if the users are interested, or foresee the possibility that at some time in the future they may become interested, in addressing the different colors for a given team individually, then we must modify the above design to accommodate this need. If we determine that all teams have the same number of colors, and no team now or in the future will ever have more than that number of colors, then we may modify the design by adding additional attributes in the TEAM entity. For example, if all teams, now and forever, will always have exactly two team colors then we may produce the design shown in Figure Q4.21c.

FIGURE Q4.21c ERD with two team colors for Question 21



This is a reasonable solution given the assurance that all teams now and forever will have exactly two team colors. A problem arises, however, if we cannot rely on that assurance. If some teams have fewer colors, then our design will lead to an increased number of nulls. If a team ever has more than two colors, we will have to modify the structure of the database after it has been built to add another team color attribute. This change in structure may require changes in the front-end applications so that they can properly address this new attribute. To avoid these potentially serious modifications in the future, we can re-design the database with a more robust structure that can handle any number of team colors without future modifications to the database or the front-end applications. The design with a separate table to handle the multi-valued Team_Colors attribute is shown in Figure Q4.21d.

FIGURE Q4.21d ERD with Color table for Question 21

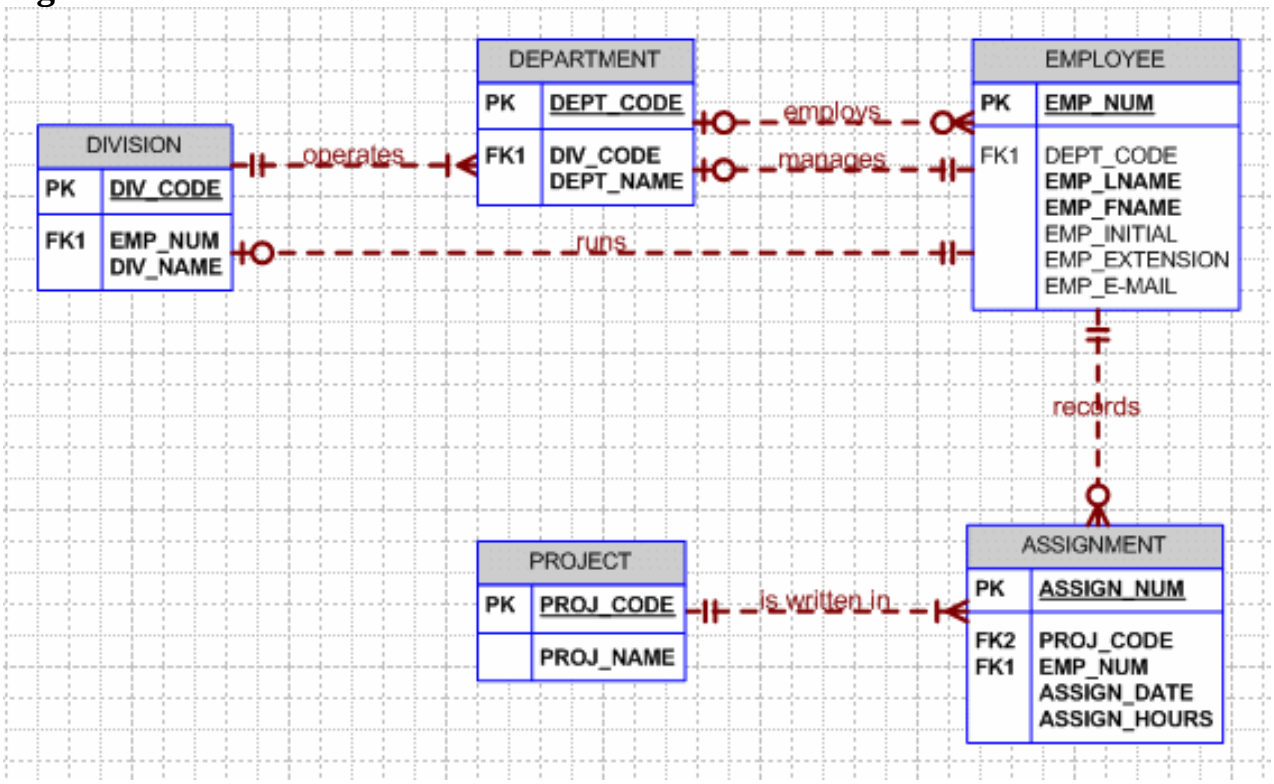


Problem Solutions

1. **Use the following business rules to create a Crow's Foot ERD. Write all appropriate connectivities and cardinalities in the ERD.**
   a. **A department employs many employees, but each employee is employed by one department.**
   b. **Some employees, known as "rovers," are not assigned to any department.**
   c. **A division operates many departments, but each department is operated by one division.**
   d. **An employee may be assigned many projects, and a project may have many employees assigned to it.**
   e. **A project must have at least one employee assigned to it.**
   f. **One of the employees manages each department, and each department is managed by only one employee.**
   g. **One of the employees runs each division, and each division is run by only one employee.**

The answers to problem 1 (all parts) are included in Figure P4.1.

Figure P4.1 Problem 1 ERD Solution



As you discuss the ERD shown in Figure P4.1, note that this design reflects several useful features that become especially important when the design is implemented. For example:

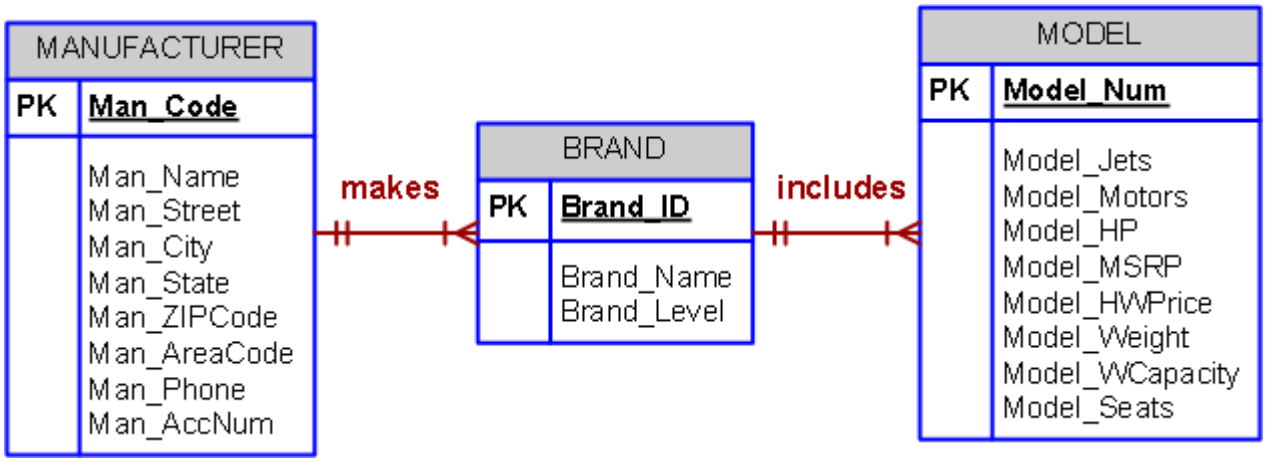
- The ASSIGN entity is shown to be optional to the PROJECT. This decision makes sense from a practical perspective, because it lets you create a new project record without having to create a new assignment record. (If a new project is started, there will not yet be any assignments.)
- The relationship expressed by "DEPARTMENT employs EMPLOYEE" is shown as mandatory on the EMPLOYEE side. This means that a DEPARTMENT must have at least one EMPLOYEE in order to have departmental status. However, DEPARTMENT is optional to EMPLOYEE, so an employee can be entered without entering a departmental FK value. If the existence of nulls is not acceptable, you can create a "No assignment" record in the DEPARTMENT table, to be referenced in the EMPLOYEE table if an employee is not assigned to a department.
- Note also the implications of the 1:1 "EMPLOYEE manages DEPARTMENT" relationship. The flip side of this relationship is that "each DEPARTMENT is managed by one EMPLOYEE". (This latter relationship is shown as mandatory in the ERD. That is, each department must be managed by an employee!) Therefore, one of the EMPLOYEE table's PK values must appear as the FK value in the DEPARTMENT table. (**Because this is a 1:1 relationship, the index property of the EMP_NUM FK in the DEPARTMENT table must be set to "unique."**)

- Although you ought to approach a 1:1 relationship with caution – most 1:1 relationships are the result of a misidentification of attributes as entities – the 1:1 relationships reflected in the "EMPLOYEE manages DEPARTMENT" and "EMPLOYEE runs DISISION" are appropriate. These 1:1 relationships avoid the data redundancies you would encounter if you duplicated employee data – such a names, phones, and e-mail addresses – in the DIVISION and DEPARTMENT entities.

Also, if you have multiple relationships between two entities -- such as the "EMPLOYEE manages DEPARTMENT" and "DEPARTMENT employs EMPLOYEE" relationships – you must make sure that each relationship has a designated primary entity. For example, the 1:1 relationship expressed by "EMPLOYEE manages DEPARTMENT" requires that the EMPOYEE entity be designated as the primary (or "first") entity. If you use Visio to create your Crow's Foot ERDs, Figure P4.3 show how the 1:1 relationship is specified. If you use some other CASE tool, you will discover that it, too, is likely to require similar relationship specifications.

2. **Create a complete ERD in Crow's Foot notation that can be implemented in the relational model using the following description of operations. Hot Water (HW) is a small start-up company that sells spas. HW does not carry any stock. A few spas are set up in a simple warehouse so customers can see some of the models available, but any products sold must be ordered at the time of the sale.**
   - **HW can get spas from several different manufacturers.**
   - **Each manufacturer produces one or more different brands of spas.**
   - **Each and every brand is produced by only one manufacturer.**
   - **Every brand has one or more models.**
   - **Every model is produced as part of a brand. For example, Iguana Bay Spas is a manufacturer that produces Big Blue Iguana spas, a premium-level brand, and Lazy Lizard spas, an entry-level brand. The Big Blue Iguana brand offers several models, including the BBI-6, an 81-jet spa with two 6-hp motors, and the BBI-10, a 102-jet spa with three 6-hp motors.**
   - **Every manufacturer is identified by a manufacturer code. The company name, address, area code, phone number, and account number are kept in the system for every manufacturer.**
   - **For each brand, the brand name and brand level (premium, mid-level, or entry-level) are kept in the system.**
   - **For each model, the model number, number of jets, number of motors, number of horsepower per motor, suggested retail price, HW retail price, dry weight, water capacity, and seating capacity must be kept in the system.**

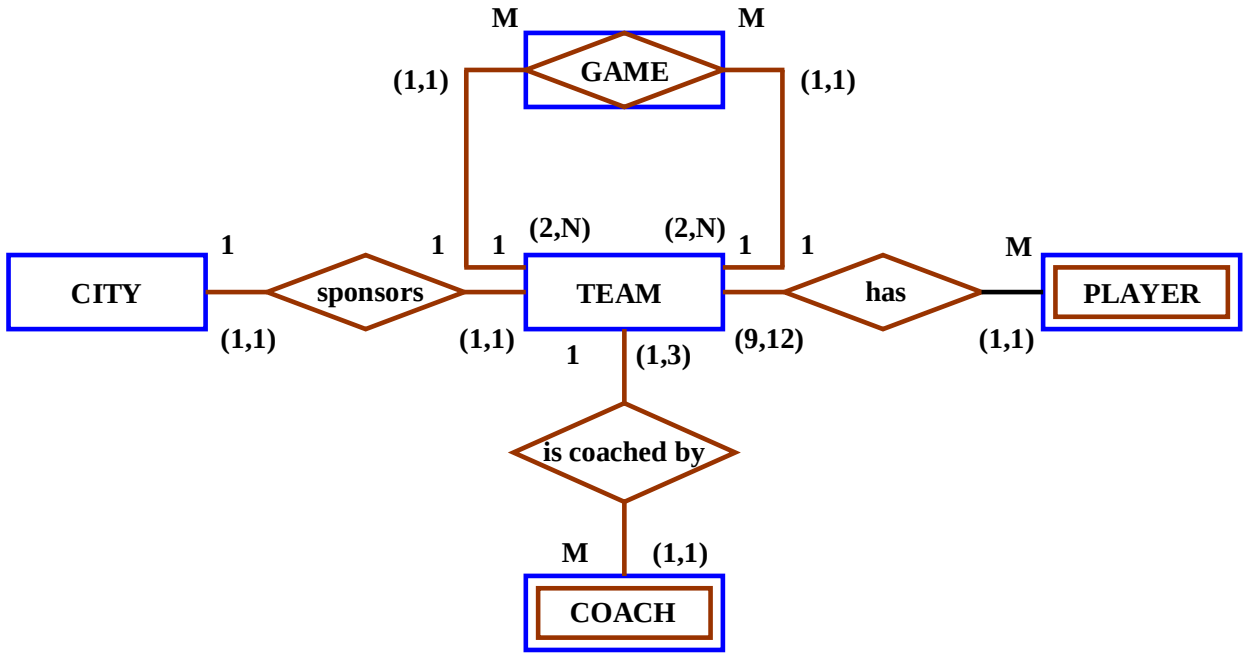
Figure P4.2 Problem 2 ERD Solution



3. **The Jonesburgh County Basketball Conference (JCBC) is an amateur basketball association. Each city in the county has one team as its representative. Each team has a maximum of 12 players and a minimum of 9 players. Each team also has up to three coaches (offensive, defensive, and physical training coaches). During the season, each team plays two games (home and visitor) against each of the other teams. Given those conditions, do the following:**
   a. **Identify the connectivity of each relationship.**
   b. **Identify the type of dependency that exists between CITY and TEAM.**
   c. **Identify the cardinality between teams and players and between teams and city.**
   d. **Identify the dependency between coach and team and between team and player.**
   e. **Draw the Chen and Crow's Foot ERDs to represent the JCBC database.**
   f. **Draw the UML class diagram to depict the JCBC database.**

   The Chen ERD solution is shown in Figure P4.3Chen. (The Crow's Foot solution is shown after the discussion.)

Figure P4.3Chen The JCBC Chen ERD



To help the students understand the ER diagram's components better, note the following relationships:

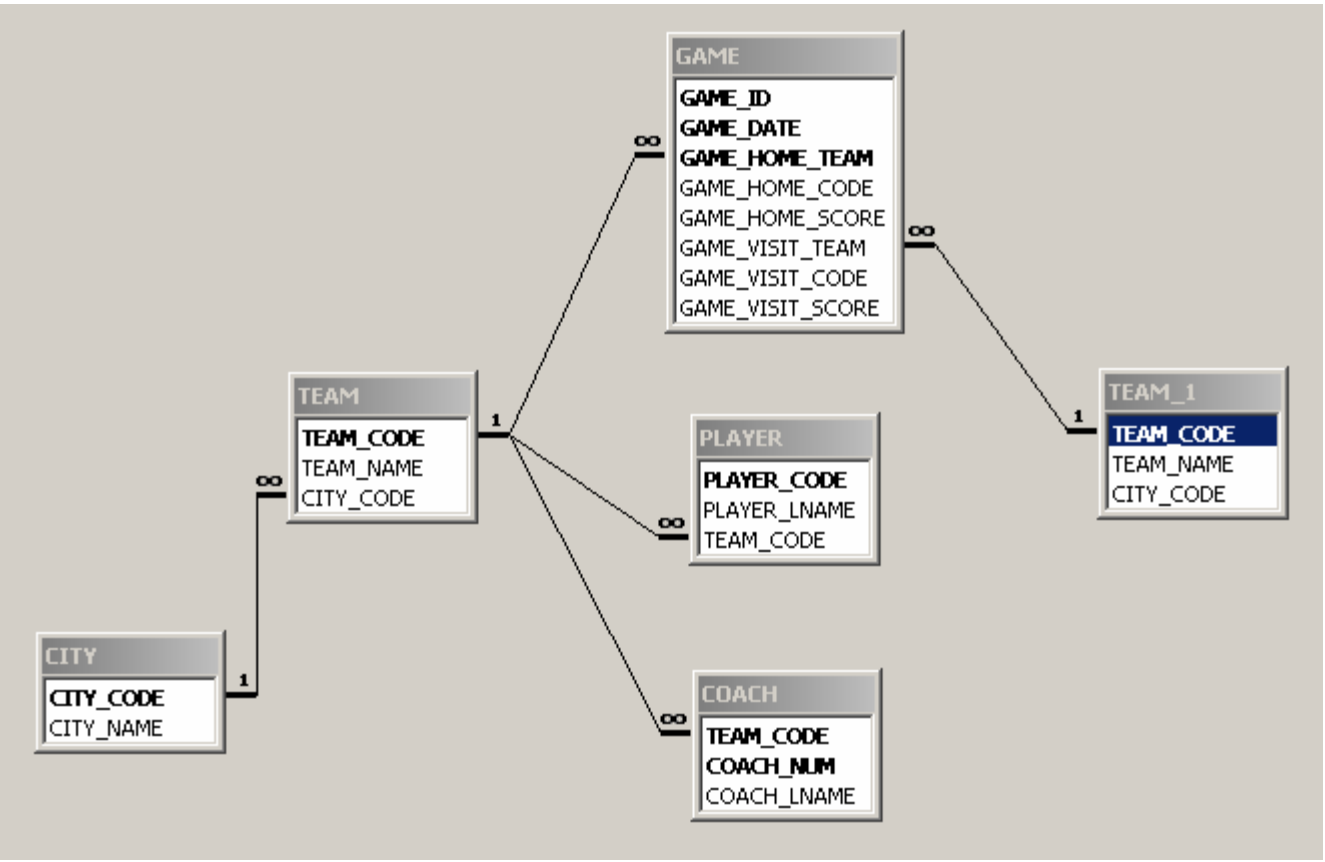
- The main components are TEAM and GAME.
- Each team plays each other team at least twice.
- To play a game, two teams are necessary: the home team and the visiting team.
- Each team plays at least twice: once as the home team and once as the visiting team.

Given these relationships, it becomes clear that TEAM participates in a recursive M:N relationship with GAME. The relationship between TEAM and GAME becomes clearer if we list some attributes for each of these entities. **Note that the TEAM_NUM appears twice in a GAME record: once as a GAME_HOME_TEAM and once as a GAME_VISIT_TEAM.**

GAME entity	**TEAM entity**
GAME_ID	TEAM_CODE
GAME_DATE	TEAM_NAME
GAME_HOME_TEAM	CITY_CODE
GAME_VISIT_TEAM	
GAME_HOME_SCORE	
GAME_VISIT_SCORE	

Implementation of this solution yields the relational diagram shown in Figure P4.3RD. (If you implement this design in Microsoft Access, note that Access will generate a virtual table named TEAM_1 to indicate that two relationships exist between GAME and TEAM. **We created a database named Ch04_JCBC_V1 to illustrate this design implementation**.

Figure P4.3RD The JCBC Relational Diagram, Version 1



The solution shown in Figure P4.3Chen yields a database that enables its users to track all games. For example, a simple query – based on the two relationships between TEAM and GAME yields the output shown in Figure P4.3SO. (We have created only a few records to show the results for games 1 and 2 played by teams named Bears, Rattlers, Sharks, and Tigers, respectively.)

Figure P4.3SO The JCBC Database Game Summary Output, Version 1

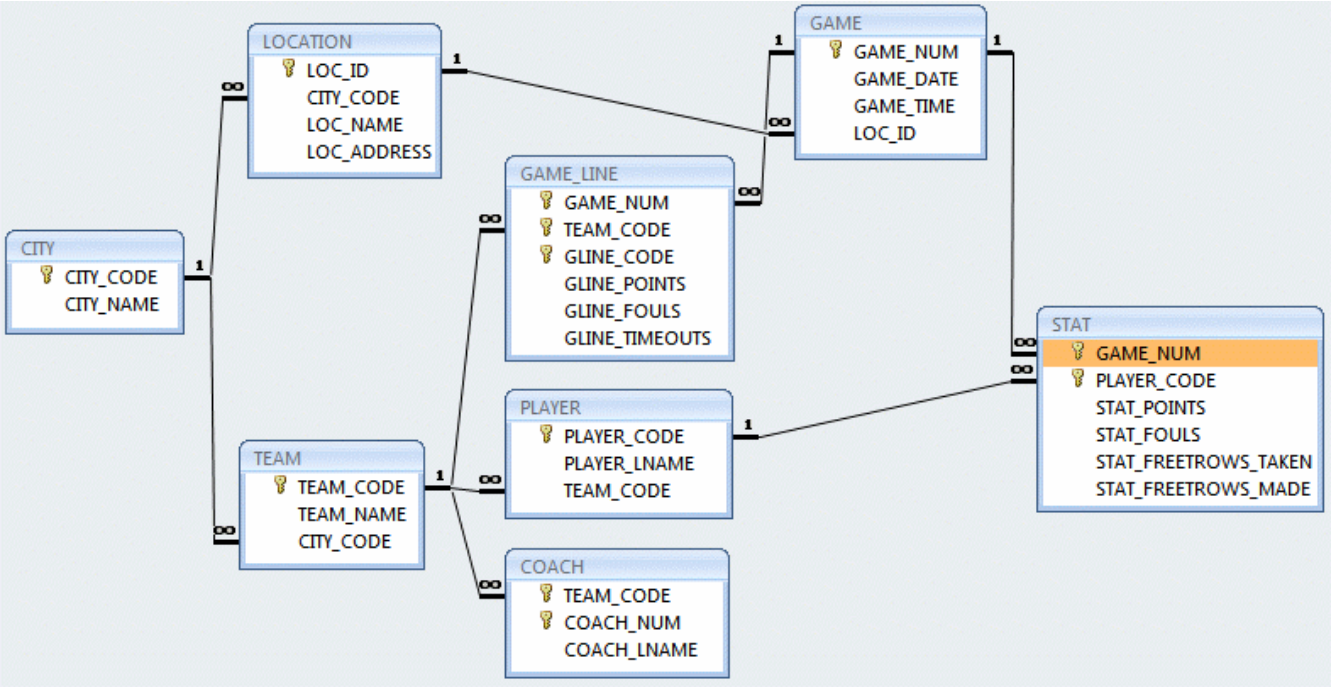
Game	Date	Team	Designation	Score	Team	Designation	Score
1	11-Mar-2008	Bears	Home	65	Rattlers	Visit	63
1	12-Mar-2008	Sharks	Home	64	Tigers	Visit	69
2	15-Mar-2008	Bears	Visit	75	Rattlers	Home	78
2	18-Mar-2008	Sharks	Visit	56	Tigers	Home	52

As you examine the design and its implementation – check the relational diagram in Figure P4.3RD -- note that this solution uses synonyms, because the TEAM_NUM shows up in GAME twice: once as the GAME_HOME_TEAM and once as the GAME_VISIT_TEAM. Given the use of these synonyms, the GAME entity also becomes very cumbersome structurally as you decide to track more game data. For example, if you wanted to keep track of runs, hits, and errors, you would have to have one set of each for each of the two teams – all in the same record. Clearly, such a structure is undesirable: the use of synonyms requires the addition of two new attributes – one for the home

team and one for the visiting team -- for each additional characteristic you want to track.

To eliminate the structural problem discussed in the previous paragraph, you can let each game be represented by two entities: GAME and GAME_LINE. Figure P4.3RD2 shows the structures of these two entities in a segment of the revised relational diagram. We have added a LOCATION entity to specify the actual location of the game – knowing that a game is played in Nashville is not sufficiently specific. Players, coaches, and spectators ought to know where in Nashville the game is played.

Figure P4.3RD2 The Revised JCBC Database Relational Diagram



NOTE

Quite aside from the fact that we ought to know where in each city any given game is played, the LOC_ID attribute in GAME refers to a LOCATION entity that was created to make the database more flexible by permitting the use of multiple locations in each city. Although this capability was not required by the problem description – each city only fields one team at this point – is very likely that additional teams will be organized in the future.

Good design first ensures that current requirements are met. This design does that. But good design also anticipates the reasonably expected changing dynamics of the database environment. This revised design does that, too.

Additional flexibility is gained by the use of the GAME entity. For example, if you want to track the assignment of referees in each of the games, you can easily create a REFEREE entity in a M:N relationship with the GAME entity. (A referee may referee many games and many referees referee each game.) This M:N relationship may then be transformed into two 1:M relationships through the use of a composite entity, perhaps named REF_GAME.

Finally, point out to the students that the relationship between the newly created GAME and GAME_LINE entities is structurally similar to the by now familiar relationship between INVOICE and INV_LINE entities.

The completed database design is implemented as shown in the Crow's Foot ERD in Figure P4.3CF.

Figure P4.3CF The JCBC Crow's Foot ERD

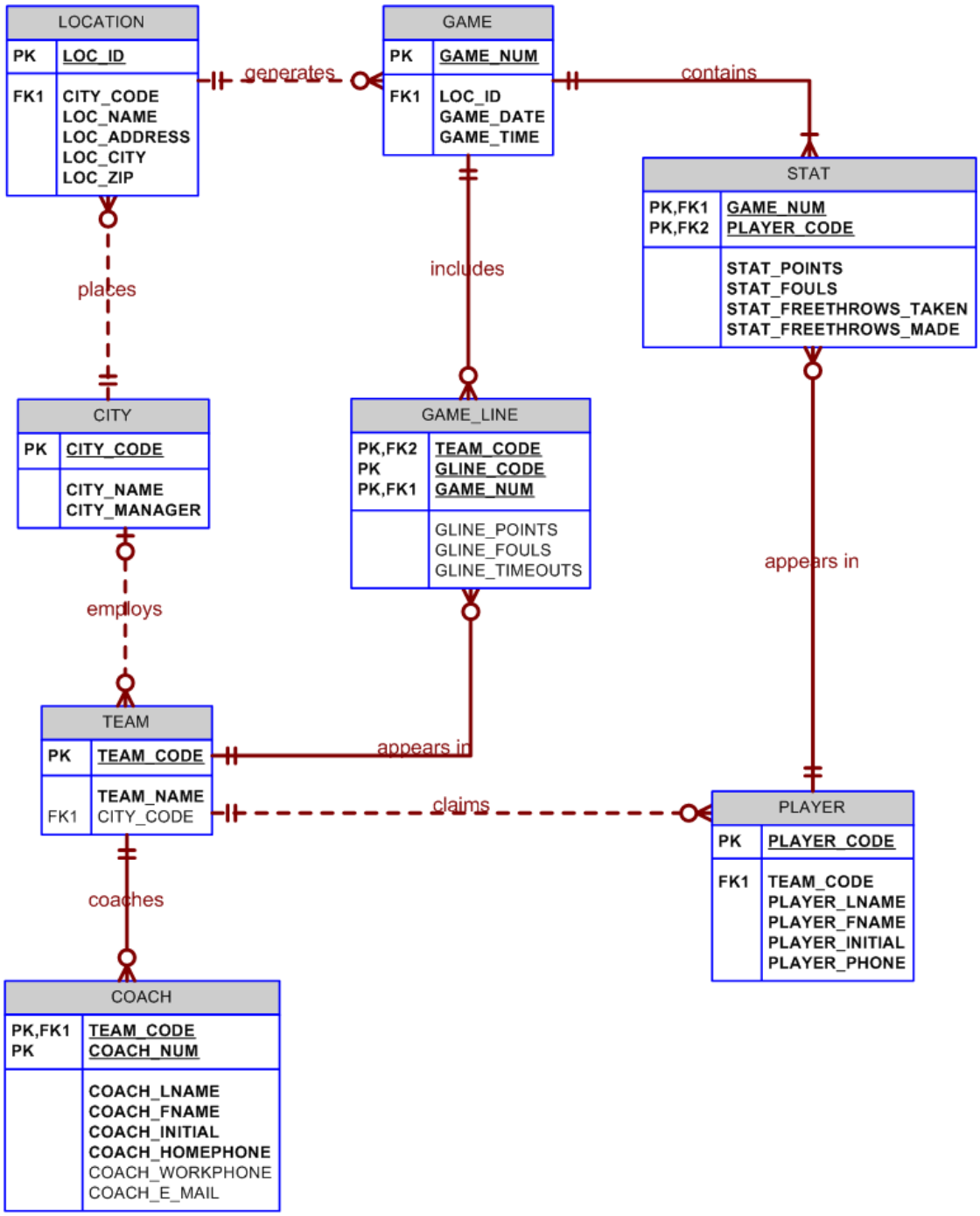
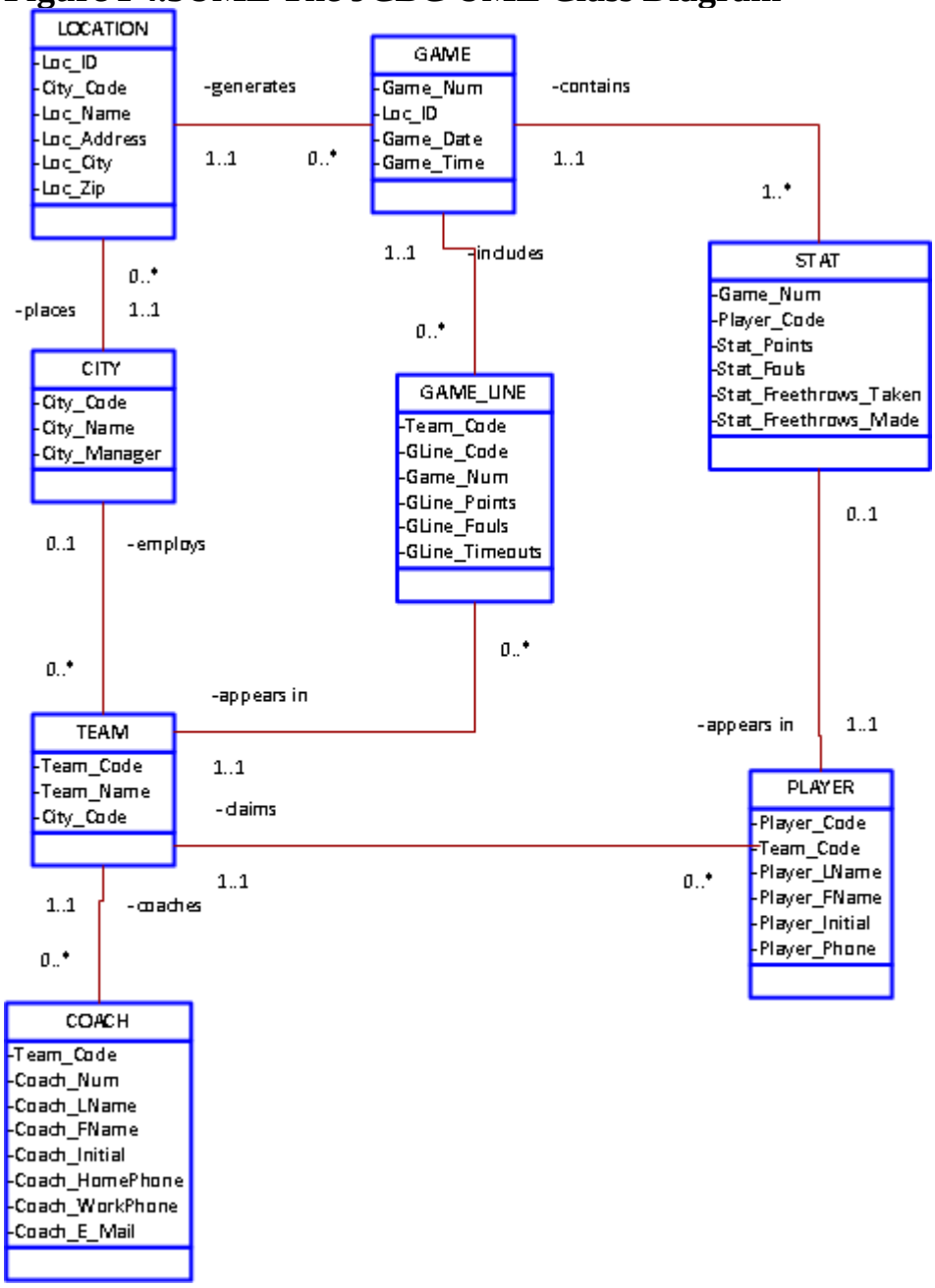


Figure P4.3UML The JCBC UML Class Diagram



NOTE

You may wonder why we examined this solution in such detail. (The sample implementation is shown in the database named Ch04_JCBC_Version2.) After all, mere games hardly seem to merit this level of database design attention.

Actually, there is the proverbial method in the madness. The basketball – or any other game environment -- is likely to be familiar to your students. Therefore, it becomes easier for you to show the design and implementation of recursive relationships – which are actually rather complex things. Fortunately, even complex design issues become manageable in a familiar data environment.

Recursive relationships are common enough – or should be – to merit attention and the development of expertise in their implementation. In many manufacturing industries, incredibly detailed part tracking is mandatory. For example, the implementation of the recursive relationship "PART contains PART" is especially desirable in the aviation manufacturing businesses. Such businesses are required by federal law to maintain absolute parts tracing records. If a complex part fails, it must be possible to follow all the trails to all the component parts that may have been involved in the part's failure.

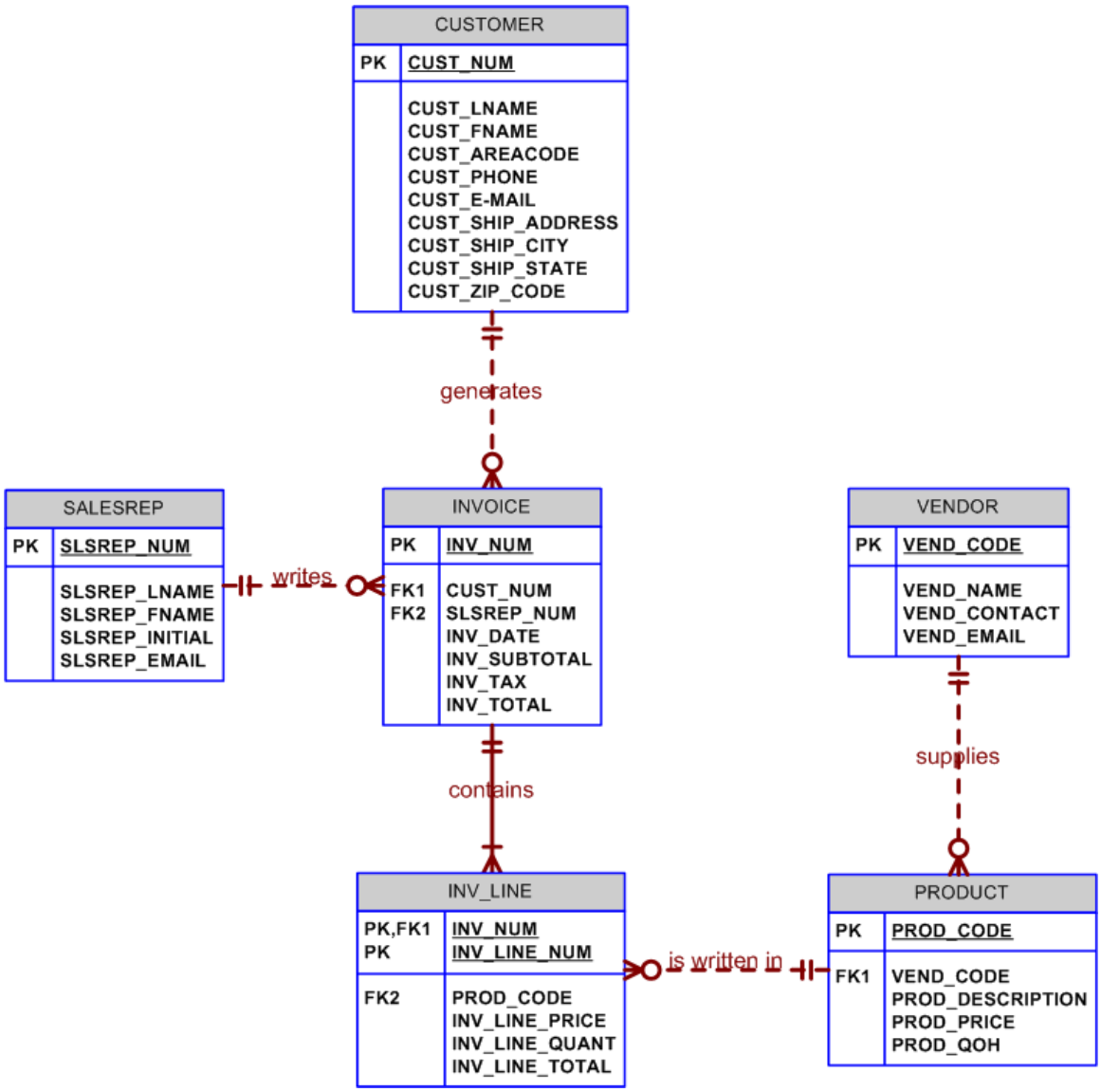
4. **Create an ERD based on the Crow's Foot model, using the following requirements:**
   - **An INVOICE is written by a SALESREP. Each sales representative can write many invoices, but each invoice is written by a single sales representative.**
   - **The INVOICE is written for a single CUSTOMER. However, each customer can have many invoices.**
   - **An INVOICE can include many detail lines (LINE), each of which describes one product bought by the customer.**
   - **The product information is stored in a PRODUCT entity.**
   - **The product's vendor information is found in a VENDOR entity.**

NOTE

The ERD must reflect business rules that you are free to define (within reason). Make sure that your ERD reflects the conditions you require. Finally, make sure that you include the attributes that would permit the model to be successfully implemented.

The Crow's Foot ERD solution is shown in Figure 4.4.

Figure P4.4 The Crow's Foot ERD Solution for Problem 4



NOTE

Keep in mind that the preceding ER diagram reflects a set of business rules that may easily be modified. For example, if customers are supplied via a commercial customer list, many of the customers on that list will not (yet!) have bought anything, so INVOICE would be optional to CUSTOMER. We are assuming here that many vendors can supply a product and that each vendor can supply many products. The PRODUCT may be optional to VENDOR if the vendor list includes potential vendors from which you have not (yet) ordered anything. Some products may never sell, so LINE is optional to PRODUCT... because an unsold product will never appear in an invoice line. You may also want to show the students how the composite entities may be represented at the final implementation level. For example, LINE is shown as weak to INVOICE, because it borrows the invoice number as part of its primary key and it is existence-dependent on INVOICE. The modified ER diagram is shown next. The point of this exercise is that the design's final iteration depends on the exact nature of the business rules and the desired level of implementation detail.

5. **The Hudson Engineering Group (HEG) has contacted you to create a conceptual model whose application will meet the expected database requirements for the company's training program. The HEG administrator gives you the description (see below) of the training group's operating environment. (Hint: Some of the following sentences identify the volume of data rather than cardinalities. Can you tell which ones?)**

   The HEG has 12 instructors and can handle up to 30 trainees per class. HEG offers five Advanced Technology courses, each of which may generate several classes. If a class has fewer than ten trainees, it will be canceled. Therefore, it is possible for a course not to generate any classes. Each class is taught by one instructor. Each instructor may teach up to two classes or may be assigned to do research only. Each trainee may take up to two classes per year.
   Given that information, do the following:

   a. **Define all of the entities and relationships. (Use Table 4.4 as your guide.)**

      The HEG entities and relationships are shown in Table P4.5a.

Table P4.5a The Components of the HEG ERD

ENTITY	RELATIONSHIP	CONNECTIVITY	ENTITY
INSTRUCTOR	teaches	1:M	CLASS
COURSE	generates	1:M	CLASS
CLASS	is listed in	1:M	ENROLL
TRAINEE	is written in	1:M	ENROLL

As you examine the summary in Table P4.5a, it is reasonable to assume that many of the relationships are optional and that some are mandatory. (Remember a point we made earlier: when in doubt, assume an optional relationship.)

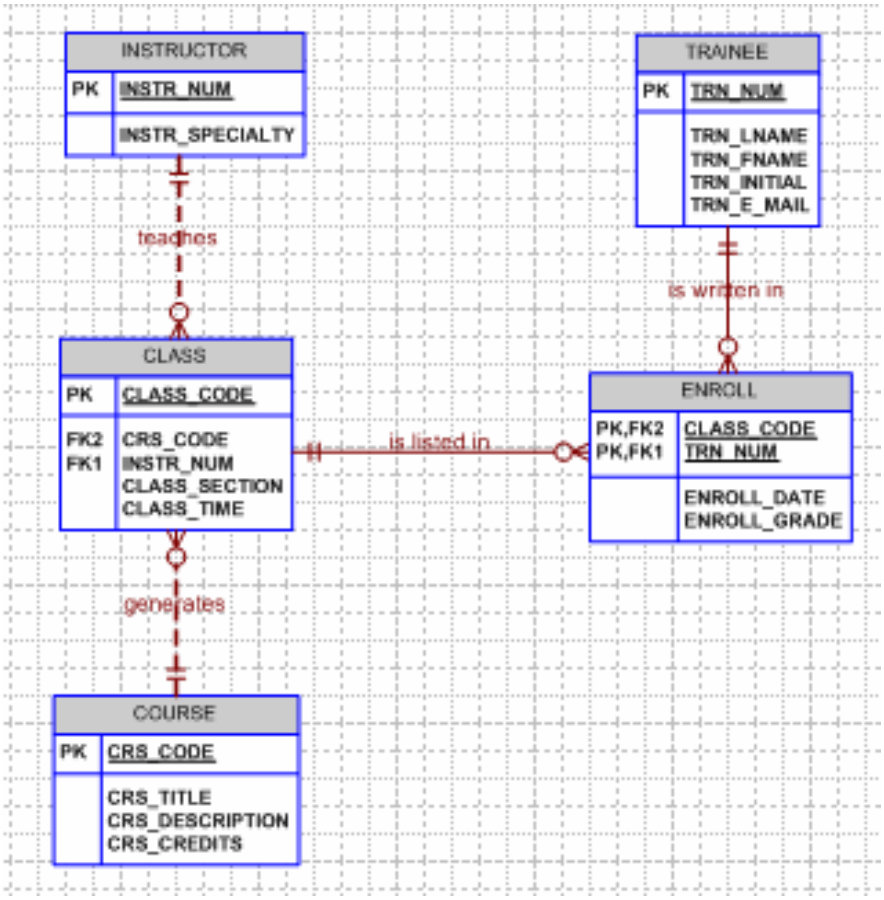
- A COURSE does not necessarily generate a class during each training period. (Some courses may be taught every other period or during some other specified time frames. Therefore, it is reasonable to assume that CLASS is optional to COURSE.
- Each CLASS must be related to a COURSE. (The class must cover designated course material!) Therefore, COURSE is mandatory to CLASS.
- Some instructors may teach a class every other period or even rarely. Therefore, it is reasonable to assume that CLASS is optional to INSTRUCTOR during any enrollment period. This optionality makes sense from an implementation point of view, too. For example, if you appoint a new instructor, that instructor will not – yet – have taught a class.
- Not all trainees are likely to be enrolled in classes during some time period. In fact, in a real world setting, many trainees are likely to get informal "on the job" training without going to formal classes. Therefore, it is reasonable to assume that ENROLL is optional to TRAINEE.
- You cannot create an enrollment record without having a trainee. Therefore, TRAINEE is mandatory to ENROLL. (Discussion point: What about making TRAINEE optional to ENROLL? In any case, optional relationships may be used for operational reasons, whether or not they are directly derived from a business rule.)

Note that a real world database design requires the explicit recognition of each relationship's characteristics. When in doubt, ask the end users!

b. Describe the relationship between instructor and class in terms of connectivity, cardinality, and existence-dependence.

Both questions (a) and (b) have been addressed in the ER diagram shown in Figure P4.4b.

Figure P4.5b The HEG ERD



As you discuss Figure P4.5b, keep the discussion in part (a) in mind. Also, note the following points:

- A trainee can take more than one class, and each class contains many (10 or more) trainees, so there is a M:N relationship between TRAINEE and CLASS. (Therefore, a composite entity is used to serve as the bridge between TRAINEE and CLASS.)
- A class is taught by only one instructor, but an instructor can teach up to two classes. Therefore, there is a 1:M relationship between INSTRUCTOR and CLASS.
- Finally, a COURSE may generate more than one CLASS, while each CLASS is based on one COURSE, so there is a 1:M relationship between COURSE and CLASS.

These relationships are all reflected in the ER diagram shown in Figure P4.4b. Note the optional and mandatory relationships:

- To exist, a CLASS must have TRAINEEs enrolled in it, but TRAINEEs do not necessarily take CLASSes. (Some may take "on the job training.")
- An INSTRUCTOR may not be teaching any CLASSes during some enrollment periods. For example, an instructor may be assigned to duties other than training. However, each CLASS must have an INSTRUCTOR.
- If an insufficient number of people sign up for a CLASS, a COURSE may not generate any CLASSes, but each CLASS must represent a COURSE.

NOTE

The sentences "HEG has twelve instructors." and "HEG offers five advanced technology courses." are not reflected in the ER diagram. Instead, they represent additional information concerning the volume of data (number of entities in an entity set), rather than information concerning entity relationships.

Because the HEG description in Problem 4 leaves room for different interpretations of optional vs. mandatory relationships, we like to give the student the benefit of the doubt. Therefore, unless the question or problem description is sufficiently precise to leave no doubt about the existence of optional/mandatory relationships, we base the student grade on two criteria:

1. **Was the basic nature of the relationship – 1:1, 1:M, or M:N – selected and displayed properly?**
2. **Given the student's rendering of such a relationship, are the cardinalities appropriate?**

You can add substantial detail to the ERD by including sample attributes for each of the entities. Using Visio Professional, you can also let your student declare the nature – weak or strong – of the relationships among the entities. Finally, remind your students that the order in which the attributes appear in each entity is immaterial. Therefore, the (composite) PK of the can be written as either CLASS_CODE + TRN_NUM or as TRN_NUM + CLASS_CODE. That's why it is also immaterial which one of the foreign key attributes is FK1 or FK2.

As you discuss the ERD shown in Figure P4.5b, note that the basic components of this problem are found in the text's Figure 4.35. Note also that the ENROLL entity in Figure P4.5b uses a composite PK (TRN_NUM + CLASS_CODE) and that, therefore the relationships between ENROLL and CLASS and TRAINEE are strong. Finally, discuss the reason for the weak relationship between COURSE and CLASS – the CLASS entity's PK (CLASS_CODE) does not "borrow" the PK of the parent COURSE entity. If the CLASS entity's PK had been composed of CRS_CODE + CLASS_SECTION, the relationship between COURSE and CLASS would have been strong.

Discussion: Review the text to show the two possible relationship strengths between COURSE and CLASS. Emphasize that the choice of the PK component(s) is usually a designer option, but that single-attribute PKs tend to yield more design options than composite PKs. Even the composite ENROLL entity can be modified to have a single-attribute PK such as ENROLL_NUM. Given that choice, CLASS_CODE + TRN_NUM constitute a candidate key – CLASS_CODE and TRN_NUM continue to serve as foreign keys to CLASS and TRAINEE, respectively. Given the latter scenario, you can create a (unique) composite index to prevent duplicate enrollments.

6. **Automata Inc. produces specialty vehicles by contract. The company operates several departments, each of which builds a particular vehicle, such as a limousine, a truck, a van, or an RV.**

Before a new vehicle is built, the department places an order with the purchasing department to request specific components. Automata's purchasing department is interested in creating a database to keep track of orders and to accelerate the process of delivering materials.

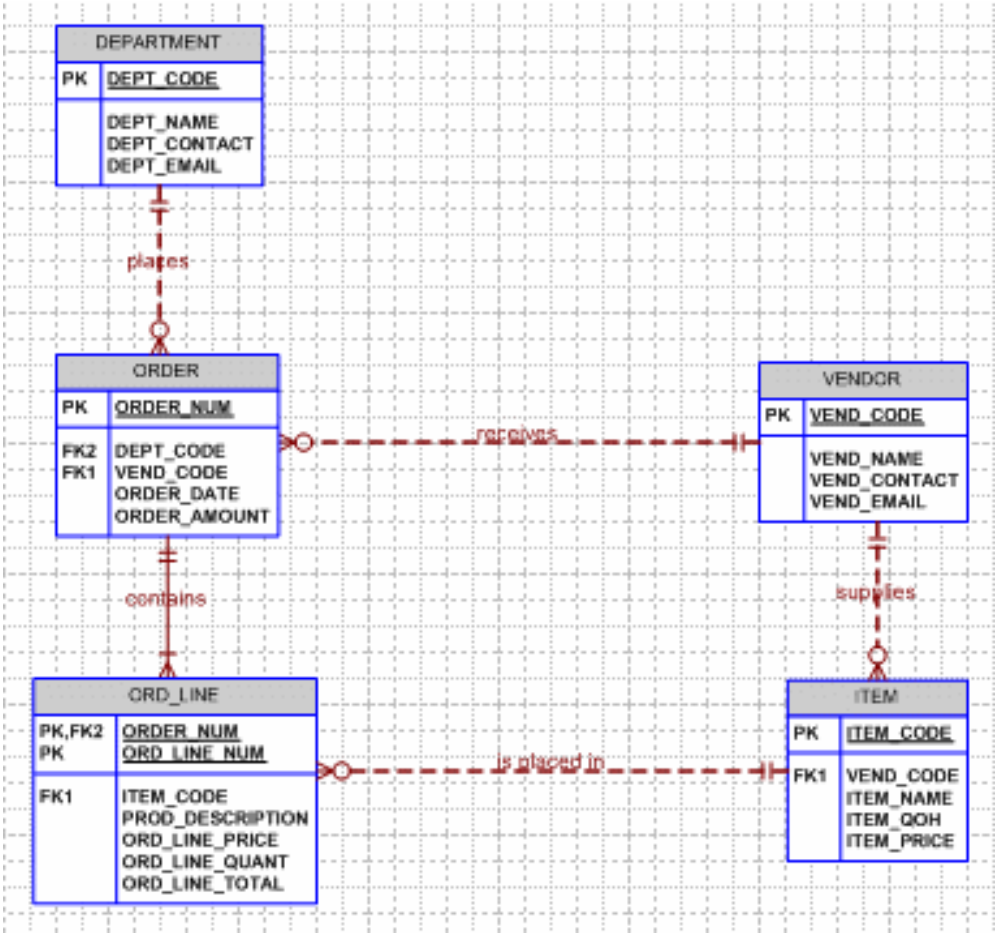
The order received by the purchasing department may contain several different items. An inventory is maintained so that the most frequently requested items are delivered almost immediately. When an order comes in, it is checked to determine whether the requested item is in inventory. If an item is not in inventory, it must be ordered from a supplier. Each item may have several suppliers.

Given that functional description of the processes encountered at Automata's purchasing department, do the following:

a. **Identify all of the main entities.**
b. **Identify all of the relations and connectivities among entities.**
c. **Identify the type of existence dependency in all the relationships.**
d. **Give at least two examples of the types of reports that can be obtained from the database.**

The initial Crow's Foot ERD is shown in Figure P4.6init. The discussion preceding Figure P4.6rev explains why the revision was made.

Figure P4.6init Initial Automata Crow's Foot ERD



As you explain the development of the Crow's Foot ERD shown in Figure P4.6init, several points are worth stressing:

- The ORDER and ORD_LINE entities are perfect reflections of the INVOICE and INV_LINE entities the students have encountered before. This kind of 1:M relationship is quite common in a business environment and you will see it recur throughout the book and in its many problems. Note that the ORD_LINE entity is weak, because it inherits part of its PK from its ORDER "parent" entity. Therefore, the "contains" relationship between ORDER and ORD_LINE is properly shown as an identifying (strong) relationship. (The relationship line is solid, rather than dashed.) Finally, note that ORD_LINE is mandatory to ORDER; it is not possible to have an ORDER that does not contain at least one order line. And, of course, ORDER is mandatory to ORD_LINE, because an ORD_LINE occurrence cannot exist without referencing an ORDER.
- The ORDER entity is shown as optional to DEPARTMENT, indicating that it is quite possible that a department has not (yet) placed an order. Aside from the fact that such an optionality makes common sense, it also makes operational sense from a database point of view. For example, if the ORDER entity were mandatory to the DEPARTMENT entity, the creation of a new department would require the creation of an order, so you might have to create a "dummy" order when you create a new department. Also, keep in mind that an

order cannot be written by a department that does not (yet) exist.

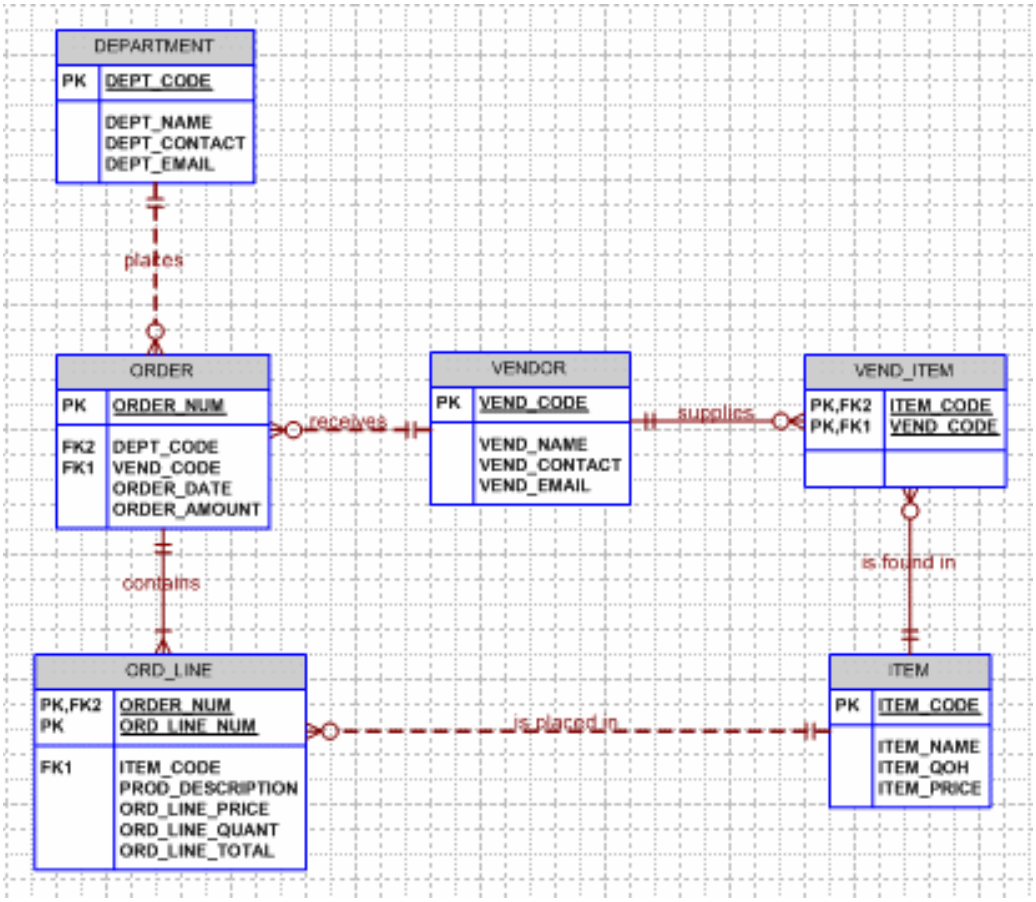
- Note also that the VENDOR may not (yet) have received and order, so ORDER is optional to VENDOR. The VENDOR entity may contain vendors who are simply potential suppliers if items and you may want to have such potential vendors available just in case your "usual" vendor(s) run(s) out of items that you need.

The other optionalities should be discussed, too – using the same basic scenarios that were described in bullets 2 and 3.

NOTE

In this presentation, the relationship between VENDOR and ITEM is shown as 1:M. Therefore, each vendor can supply many items, but only one vendor can supply each item. If it is possible for items to be supplied by more than one vendor, there is a M:N relationship between VENDOR and ITEM and this relationship would have to be implemented through a composite (bridge) entity. Actually, such an M:N relationship is specified in the brief description of the Automata company's data environment. Therefore, the following Figure P4.6rev more accurately reflects the problem description.

Figure P4.6rev Revised Automata Crow's Foot ERD

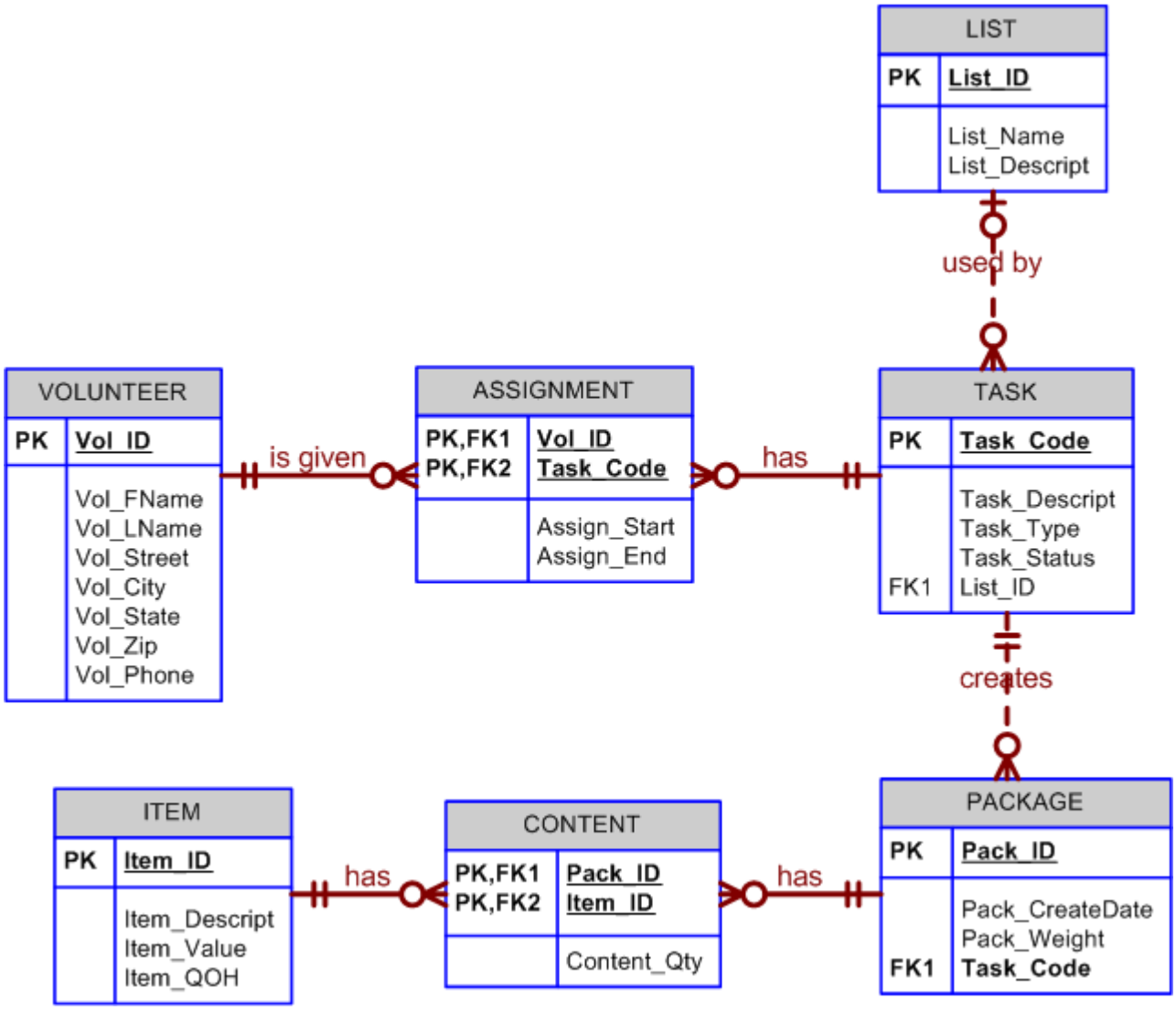


7. **United Helpers is a nonprofit organization that provides aid to people after natural disasters. Based on the following brief description of operations, create the appropriate fully labeled Crow's Foot ERD.**

   - **Individuals volunteer their time to carry out the tasks of the organization. For each volunteer, their name, address, and telephone number are tracked. Each volunteer may be assigned to several tasks during the time that they are doing volunteer work, and some tasks require many volunteers. It is possible for a volunteer to be in the system without having been assigned a task yet. It is possible to have tasks that no one has been assigned. When a volunteer is assigned to a task, the system should track the start time and end time of that assignment.**
   - **For each task, there is a task code, task description, task type, and a task status. For example, there may be a task with task code "101," description of "answer the telephone," a type of "recurring," and a status of "ongoing." There could be another task with a code of "102," description of "prepare 5000 packages of basic medical supplies," a type of "packing," and a status of "open."**
   - **For all tasks of type "packing," there is a packing list that specifies the contents of the packages. There are many different packing lists to produce different packages, such as basic medical packages, child care packages, food packages, etc. Each packing list has a packing list ID number, packing list name, and a packing list description, which describes the items that ideally go into making that type of package. Every packing task is associated with only one packing list. A packing list may not be associated with any tasks, or may be associated with many tasks. Tasks that are not packing tasks are not associated with any packing list.**
   - **Packing tasks result in the creation of packages. Each individual package of supplies that is produced by the organization is tracked. Each package is assigned an ID number. The date the package was created, and total weight of the package is recorded. A given package is associated with only one task. Some tasks (e.g., "answer the phones") will not have produced any packages, while other tasks (e.g., "prepare 5000 packages of basic medical supplies") will be associated with many packages.**
   - **The packing list describes the ideal contents of each package, but it is not always possible to include the ideal number of each item. Therefore, the actual items included in each package should be tracked. A package can contain many different items, and a given item can be used in many different packages.**
   - **For each item that the organization provides, there is an item ID number, item description, item value, and item quantity on hand stored in the system. Along with tracking the actual items that are placed in each package, the quantity of each item placed in the package must be tracked too. For example, a packing list may state that basic medical packages should include 100 bandages, 4 bottles of iodine, and 4 bottles of hydrogen peroxide. However, because of the limited supply of items, a given package may include only 10 bandages, 1 bottle of iodine, and no hydrogen peroxide. The fact that this package includes bandages and iodine needs to be recorded along with the quantity of each that is included. It is possible for the organization to have items donated that have not been included in any package yet, but every package will contain at least one item.**

The ERD for United Helpers is shown in Figure P4.6a.

FIGURE P4.7a United Helpers ERD



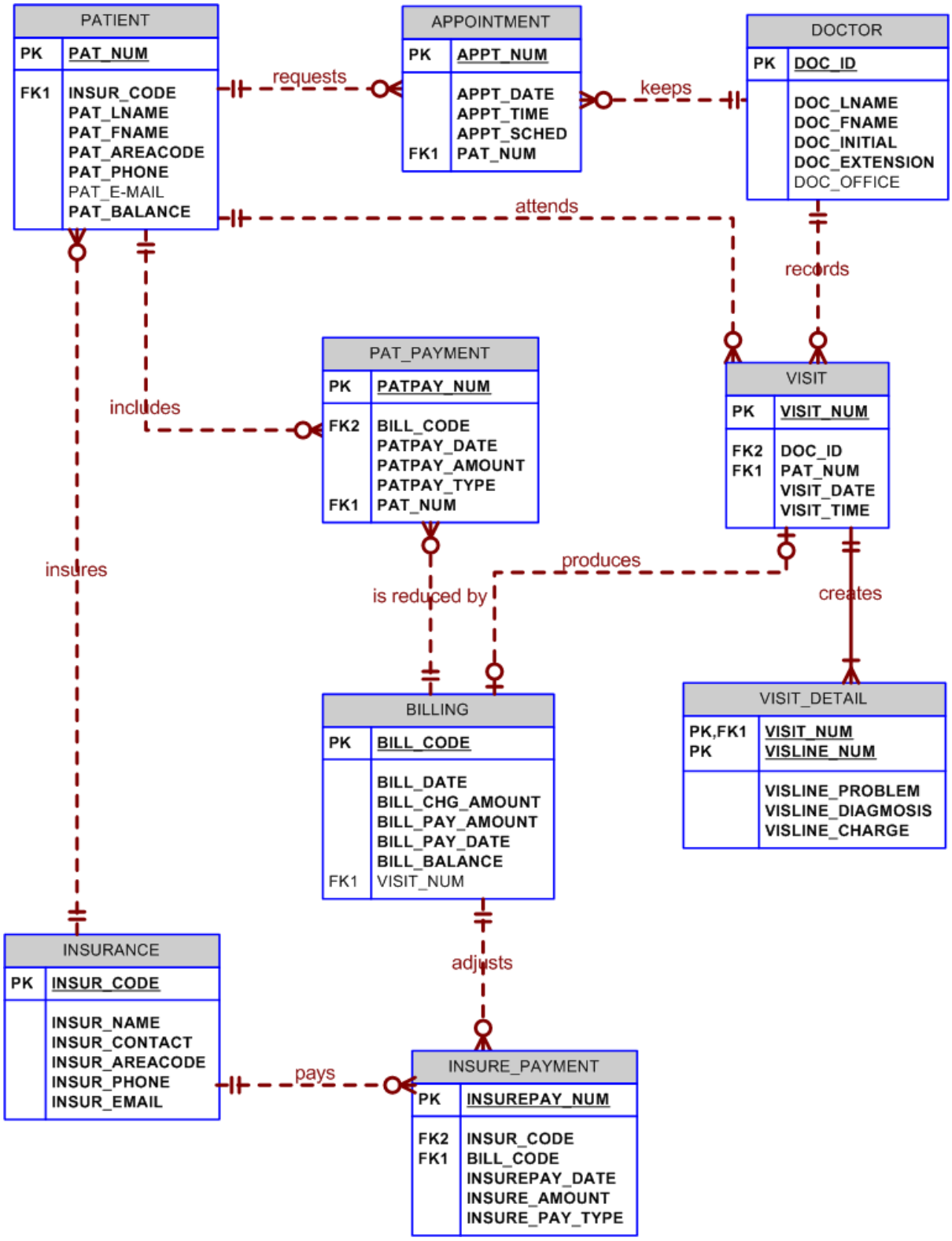
This problem, however, does leave room for interesting discussion with the students regarding the need to verify requirements with the business users. In fact, getting **unambiguous** business rules can be one of the most difficult parts of the design process. In this problem, the potential for a relationship between the packing list (LIST) and the items (ITEM) stocked by the organization can be a source for discussion. Students may envision that a LIST can specify many ITEMs and an ITEM can be specified in many LISTs. This would imply the need for a M:N relationship between ITEM and LIST. However, the business users may not intend for the packing list to be that

specific. For example, the packing list may specify that "2 liter of iodine" should be included in a given type of package without specifying whether it should be two 1-liter bottles of iodine or four 500ml bottles of iodine. Note that "1-liter bottle of iodine" and "500ml bottle of iodine" would have to be separate entity instances in ITEM because they have different values. If it is the case that the packing list is intentionally generic in its description of the ideal contents, then a relationship between LIST and ITEM would not be appropriate.

8. **Using the Crow's Foot methodology, create an ERD that can be implemented for a medical clinic, using at least the following business rules:**
   a. **A patient can make many appointments with one or more doctors in the clinic, and a doctor can accept appointments with many patients. However, each appointment is made with only one doctor and one patient.**
   b. **Emergency cases do not require an appointment. However, for appointment management purposes, an emergency is entered in the appointment book as "unscheduled."**
   c. **If kept, an appointment yields a visit with the doctor specified in the appointment. The visit yields a diagnosis and, when appropriate, treatment.**
   d. **With each visit, the patient's records are updated to provide a medical history**
   e. **Each patient visit creates a bill. Each patient visit is billed by one doctor, and each doctor can bill many patients.**
   f. **Each bill must be paid. However, a bill may be paid in many installments, and a payment may cover more than one bill.**
   g. **A patient may pay the bill directly, or the bill may be the basis for a claim submitted to an insurance company.**
   h. **If the bill is paid by an insurance company, the deductible is submitted to the patient for payment.**

The ERD solution is shown in Figure P4.8.

Figure P4.8 The Medical Clinic's Crow's Foot ERD



Case Solutions

9. The administrators of Tiny College are so pleased with your design and implementation of their student registration/tracking system that they want you to expand the design to include the database for their motor vehicle pool. A brief description of operations follows:

A brief description of operations follows:

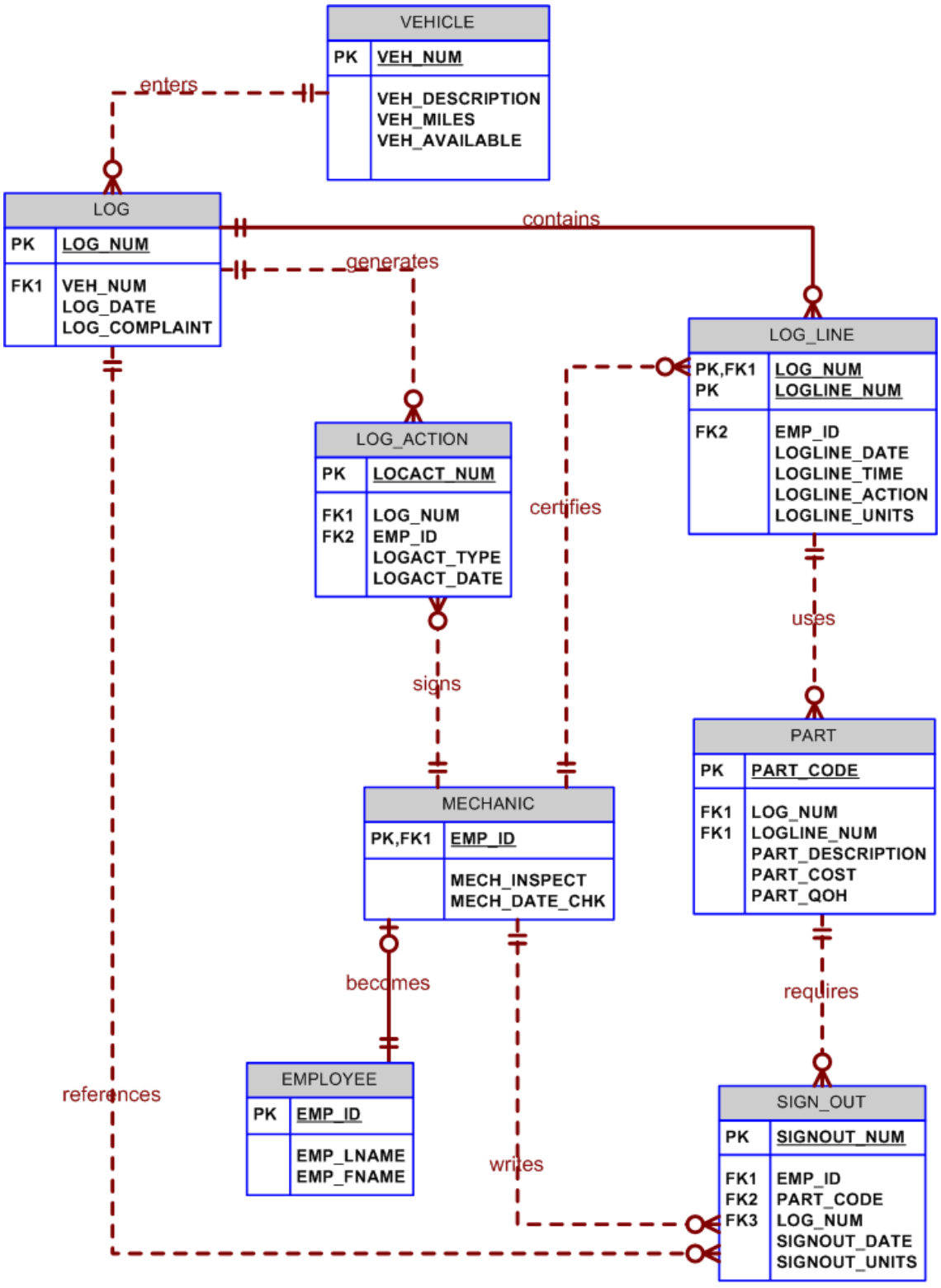
- **Faculty members may use the vehicles owned by Tiny College for officially sanctioned travel. For example, the vehicles may be used by faculty members to travel to off-campus learning centers, to travel to locations at which research papers are presented, to transport students to officially sanctioned locations, and to travel for public service purposes. The vehicles used for such purposes are managed by Tiny College's TFBS (Travel Far But Slowly) Center.**
- **Using reservation forms, each department can reserve vehicles for its faculty, who are responsible for filling out the appropriate trip completion form at the end of a trip. The reservation form includes the expected departure date, vehicle type required, destination, and name of the authorized faculty member. The faculty member arriving to pick up a vehicle must sign a checkout form to log out the vehicle and pick up a trip completion form. (The TFBS employee who releases the vehicle for use also signs the checkout form.) The faculty member's trip completion form includes the faculty member's identification code, the vehicle's identification, the odometer readings at the start and end of the trip, maintenance complaints (if any), gallons of fuel purchased (if any), and the Tiny College credit card number used to pay for the fuel. If fuel is purchased, the credit card receipt must be stapled to the trip completion form. Upon receipt of the faculty trip completion form, the faculty member's department is billed at a mileage rate based on the vehicle type (sedan, station wagon, panel truck, minivan, or minibus) used. (Hint: Do not use more entities than are necessary. Remember the difference between attributes and entities!)**
- **All vehicle maintenance is performed by TFBS. Each time a vehicle requires maintenance, a maintenance log entry is completed on a prenumbered maintenance log form. The maintenance log form includes the vehicle identification, a brief description of the type of maintenance required, the initial log entry date, the date on which the maintenance was completed, and the identification of the mechanic who released the vehicle back into service. (Only mechanics who have an inspection authorization may release the vehicle back into service.)**
- **As soon as the log form has been initiated, the log form's number is transferred to a maintenance detail form; the log form's number is also forwarded to the parts department manager, who fills out a parts usage form on which the maintenance log number is recorded. The maintenance detail form contains separate lines for each maintenance item performed, for the parts used, and for identification of the mechanic who performed the maintenance item. When all maintenance items have been completed, the maintenance detail form is stapled to the maintenance log form, the maintenance log form's completion date is filled out, and the mechanic who releases the vehicle back into service signs the form. The stapled forms are then filed, to be used later as the source for various maintenance reports.**

- **TFBS maintains a parts inventory, including oil, oil filters, air filters, and belts of various types. The parts inventory is checked daily to monitor parts usage and to reorder parts that reach the "minimum quantity on hand" level. To track parts usage, the parts manager requires each mechanic to sign out the parts that are used to perform each vehicle's maintenance; the parts manager records the maintenance log number under which the part is used.**
- **Each month TFBS issues a set of reports. The reports include the mileage driven by vehicle, by department, and by faculty members within a department. In addition, various revenue reports are generated by vehicle and department. A detailed parts usage report is also filed each month. Finally, a vehicle maintenance summary is created each month.**

Given that brief summary of operations, draw the appropriate (and fully labeled) ERD. Use the Chen methodology to indicate entities, relationships, connectivities, and cardinalities.

The solution is shown in Figure P4.9.

Figure P4.9 The Tiny-College TFBS Maintenance ERD



10. During peak periods, Temporary Employment Corporation (TEC) places temporary workers in companies. TEC's manager gives you the following description of the business:

- **TEC has a file of candidates who are willing to work.**
- **If the candidate has worked before, that candidate has a specific job history. (Naturally, no job history exists if the candidate has never worked.) Each time the candidate works, one additional job history record is created.**
- **Each candidate has earned several qualifications. Each qualification may be earned by more than one candidate. (For example, it is possible for more than one candidate to have earned a BBA degree or a Microsoft Network Certification. And clearly, a candidate may have earned both a BBA and a Microsoft Network Certification.)**
- **TEC offers courses to help candidates improve their qualifications.**
- **Every course develops one specific qualification; however, TEC does not offer a course for every qualification. Some qualifications have multiple courses that develop that qualification.**
- **Some courses cover advanced topics that require specific qualifications as prerequisites. Some courses cover basic topics that do not require any prerequisite qualifications. A course can have several prerequisites. A qualification can be a prerequisite for more than one course.**
- **Courses are taught during training sessions. A training session is the presentation of a single course. Over time, TEC will offer many training sessions for each course; however, new courses may not have any training sessions scheduled right away.**
- **Candidates can pay a fee to attend a training session. A training session can accommodate several candidates, although new training sessions will not have any candidates registered at first.**
- **TEC also has a list of companies that request temporaries.**
- **Each time a company requests a temporary employee, TEC makes an entry in the Openings folder. That folder contains an opening number, a company name, required qualifications, a starting date, an anticipated ending date, and hourly pay.**
- **Each opening requires only one specific or main qualification.**
- **When a candidate matches the qualification, the job is assigned, and an entry is made in the Placement Record folder. That folder contains an opening number, a candidate number, the total hours worked, etc. In addition, an entry is made in the job history for the candidate.**
- **An opening can be filled by many candidates, and a candidate can fill many openings.**
- **TEC uses special codes to describe a candidate's qualifications for an opening. The list of codes is shown in Table P4.10.**

TABLE P4.10 TEC QUALIFICATION CODES

CODE	DESCRIPTION
SEC-45	Secretarial work, at least 45 words per minute
SEC-60	Secretarial work, at least 60 words per minute
CLERK	General clerking work
PRG-VB	Programmer, Visual Basic
PRG-C++	Programmer, C++
DBA-ORA	Database Administrator, Oracle
DBA-DB2	Database Administrator, IBM DB2
DBA-SQLSERV	Database Administrator, MS SQL Server
SYS-1	Systems Analyst, level 1
SYS-2	Systems Analyst, level 2
NW-NOV	Network Administrator, Novell experience
WD-CF	Web Developer, ColdFusion

TEC's management wants to keep track of the following entities:

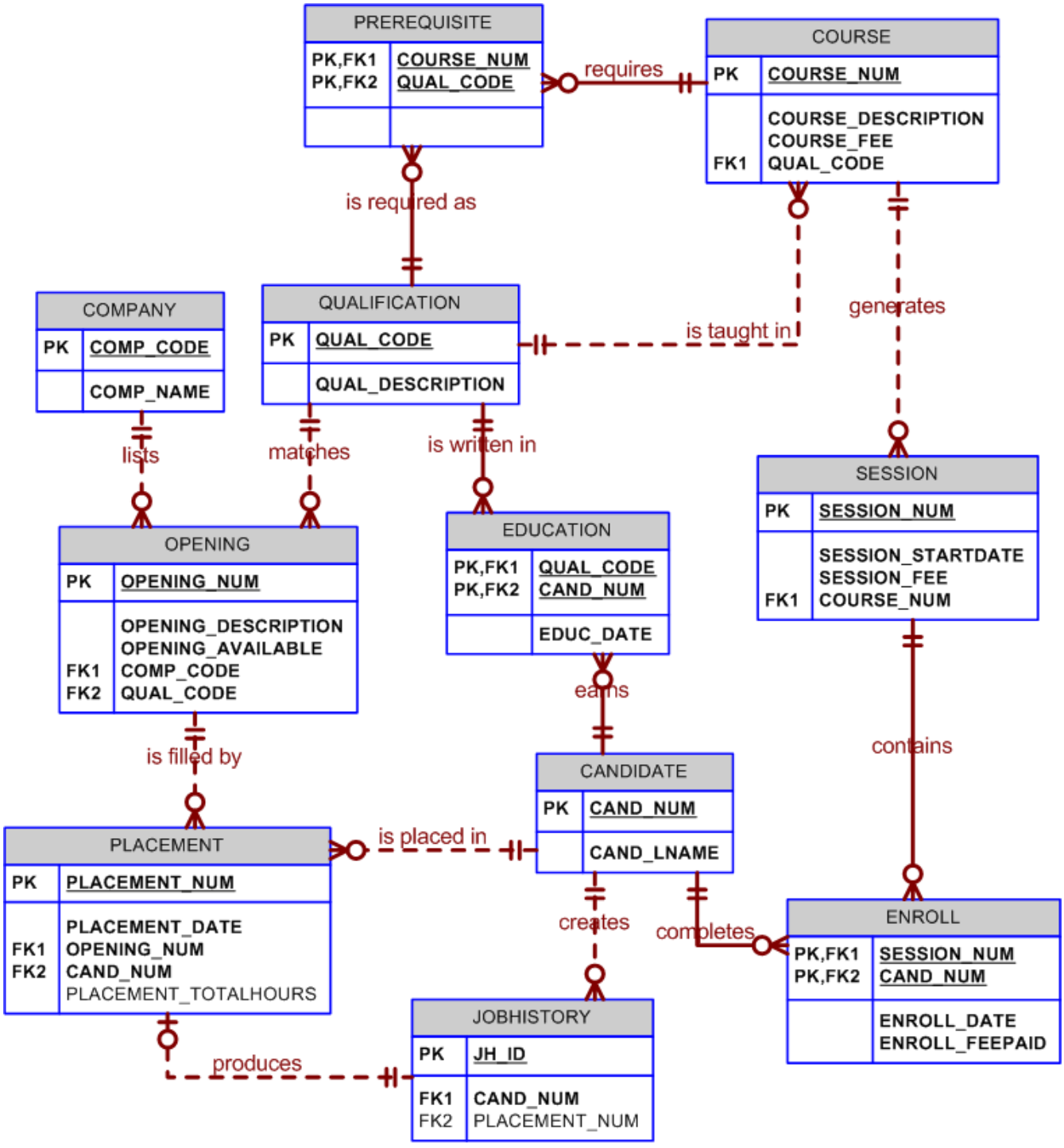
- **COMPANY**
- **OPENING**
- **QUALIFICATION**
- **CANDIDATE**
- **JOB_HISTORY**
- **PLACEMENT**
- **COURSE**
- **SESSION**

Given that information, do the following:

a. Draw the Crow's Foot ERDs for this enterprise.
b. Identify all possible relationships.
c. Identify the connectivity for each relationship.
d. Identify the mandatory/optional dependencies for the relationships.
e. Resolve all M:N relationships.

The solutions for problems 10a-10e are shown in Figure P4.10.

Figure P4.10 TEC Solution ERD



To help the students understand Figure P4.10's ER diagram's components better, the following discussion is likely to be useful:

- Each COMPANY may list one or more OPENINGs. Because we will maintain COMPANY data even if a company has not (yet!) hired any of TEC's candidates, OPENING is an

optional entity in the COMPANY lists OPENING relationship.

- OPENING is existence-dependent on COMPANY, because there cannot be an opening unless a company lists it. If you decide to use the COMPANY primary key as a component of the OPENING's primary key, you have satisfied the conditions that will allow you to classify OPENING as a weak entity and the relationship between COMPANY and OPENING will be strong or identifying. In other words, the OPENING entity is weak if its PK is the combination of OPENING_NUM and COMP_CODE. (The COMP_CODE remains the FK in the OPENING entity.)

Note that there is a 1:M relationship between COMPANY and OPENING, because a company can list multiple job openings. The next table segment shows that the WEST Company has two available job openings and the EAST Company has one available job opening. Naturally, the actual table would have additional attributes in it – but we're merely illustrating the behavior of the PK components here.

COMP_CODE	OPENING_NUM
West	1
West	2
East	1

However, if the OPENING's PK is defined to be a single OPENING attribute such as a unique OPENING_NUM, OPENING is no longer a weak entity. We have decided to use the latter approach in Figure P4.10. (If you use Microsoft Access to implement this design, OPENING_NUM may be declared as an autonumber.) **Note that this decision causes the relationship between COMPANY and OPENING to be weak**. (The relationship line is dashed.) In this case, the COMP_CODE attribute would continue to be the FK pointing to the COMPANY table, but it would no longer be a part of the OPENING entity PK. The next table segment shows what such an arrangement would look like:

OPENING_NUM	COMP_CODE
10025	West
10026	West
10027	East

- Similarly, the relationship between PLACEMENT and OPENING may be defined as strong or weak. We have used a weak relationship between OPENING and PLACEMENT.
- A job candidate may have had many jobs -- remember that TEC is a temp employer. Therefore, a candidate may have many entries in HISTORY. But keep in mind that a candidate may just have completed job training and, therefore, may not have had job experience (i.e., no job history) yet. In short, HISTORY is optional to CANDIDATE.
- To enable TEC or its clients to trace the entire employment record of any candidate, it is reasonable to expect that the HISTORY entity also records the job(s) held by the candidate before that candidate was placed by TEC. Only the portion of the job history created through TEC placement is reflected in the PLACEMENT entity. Therefore, PLACEMENT is optional to HISTORY.
- The semantics of the problem seem to suggest that the HISTORY is an entity that exists in a

1:1 relationship with PLACEMENT. After all, each placement generates one (and only one) entry in the candidate's history.

- Because each placement must generate an entry in the HISTORY entity, one would reasonably conclude that HISTORY is mandatory to PLACEMENT. Note that PLACEMENT is redundant, because a job placement obviously creates a job history entry. However, such a redundancy can be justified on the basis that PLACEMENT may be used to track job placement details that are of interest to TEC management.
- HISTORY is clearly existence-dependent on CANDIDATE; it is not possible to make an entry in HISTORY without having a CANDIDATE to generate that history. Given this scenario, the CANDIDATE entity's primary key may be used as one of the components of the HISTORY entity's primary key, thus making HISTORY a weak entity.
- Each CANDIDATE may have earned one or more QUALIFICATIONs. Although a company may list a qualification, there may not be a matching candidate because it is possible that none of the candidates have this qualification. For instance, it is possible that none of the available candidates is a Pascal programmer. Therefore, CANDIDATE is optional to QUALIFICATION. However, many candidates may have a given qualification. For example, many candidates may be C++ programmers. Keep in mind that each qualification may be matched to many job candidates, so the relationship between CANDIDATE and QUALIFICATION is M:N. This relationship must be decomposed into two 1:M relationships with the help of a composite entity we will name EDUCATION. The EDUCATION entity will contain the qualification code, the candidate identification, the date on which the candidate earned the qualification, and so on. A few sample data entries might look like this:

QUAL_CODE	CAND_NUM	EDUC_DATE
PRG-VB	4358	12-Dec-00
PRG-C++	4358	05-Mar-03
DBA-ORA	4358	23-Nov-01
DBA-DB2	2113	02-Jun-85
DBA-ORA	2113	26-Jan-02

Note that the preceding table contents illustrate that candidate 4358 has three listed qualifications, while candidate 2113 has two listed qualifications. Note also that the qualification code DBA-ORA occurred more than once. Clearly, the PK must be a combination of QUAL_CODE and CAND_NUM, thus making the relationships between QUALIFICATION and EDUCATION and between EDUCATION and CANDIDATE strong. **In this example, the EDUCATION entity is both weak and composite**.

NOTE

If you use Visio to create the ERD, you only enter the EDUC_DATE column in the EDUCATION entity. Do not type the foreign key attributes under the column headings – Visio will automatically create the FK entries as you declare the relationships.

In this ERD, select the relationships between QUALIFICATION and EDUCATION and between EDUCATION and CANDIDATE to be strong, thus ensuring that the relationship lines will be solid, rather than dashed. The QUAL_CODE and the CAND_NUM will automatically be inserted as PKs and as FKs in the EDUCATION entity. If you declare the QUAL_CODE and the CAND_NUM attributes in the EDUCATION entity and you then create the relationship lines, Visio will write duplicate QUAL_CODE and the CAND_NUM attributes into the EDUCATION entity as PKs and FKs. Clearly, this result is undesirable if the entity is implemented as a table.

Follow this simple rule: Never declare a FK in an entity by creating it yourself. Let Visio write all the FKs as you establish the relationships.

- Each job OPENING requires one QUALIFICATION, and any given qualification may fit many openings, thus producing a 1:M relationship between QUALIFICATION and OPENING. For example, a job opening for a C++ programmer requires an applicant to have the C++ programming qualification, but there may be many job openings for C++ programmers! However, a qualification does not require an opening. (After all, if there is no listing with a C++ requirement, a candidate who has the C++ qualification does not match the listing!) Therefore, OPENING is optional to QUALIFICATION.

  In the ERD shown in Figure P4.10a, we decided to define the OPENING entity's PK to be OPENING_NUM. **This decision produces a non-identifying (weak) relationship between OPENING and QUALIFICATION**. However, if you want to ensure that there cannot be a listed opening unless it also lists the required qualification for that opening, the OPENING is existence-dependent on QUALIFICATION. If you then decide to let the OPENING entity inherit QUAL_CODE from QUALIFICATION as part of its PK, OPENING is properly classified as a weak entity to QUALIFICATION.
- One or more candidates may fill a listed job opening. Also, keep in mind that, during some period of time, a candidate may fill many openings. (TEC supplies temporaries, remember?) Therefore, the relationship between OPENING and CANDIDATE is M:N. We will decompose this M:N relationship into two 1:M relationships, using the composite entity named PLACEMENT as the bridge between CANDIDATE and OPENING.
- Because a candidate is not necessarily placed, PLACEMENT is optional to CANDIDATE. Similarly, since an opening may be listed even when there is no available candidate, PLACEMENT is optional to OPENING.

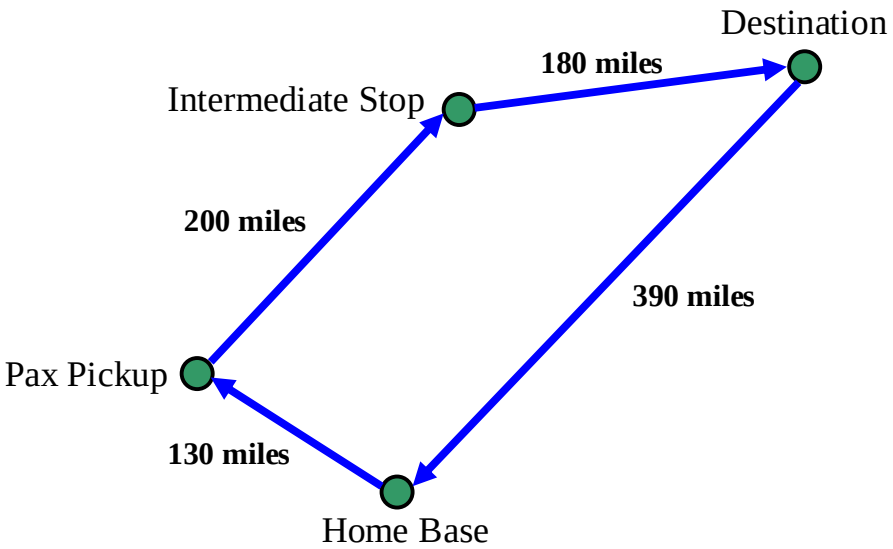
11. Use the following description of the operations of the RC_Charter2 Company to complete this exercise.

The RC_Charter2 Company operates a fleet of aircraft under the Federal Air Regulations Part 135 (air taxi or charter) certificate, enforced by the FAA. The aircraft are available for air taxi (charter) operations within the United States and Canada.

Charter companies provide so-called "unscheduled" operations—that is, charter flights take place only after a customer reserves the use of an aircraft to fly at a customer-designated date and time to one or more customer-designated destinations, transporting passengers, cargo, or some combination of passengers and cargo. A customer can, of course, reserve many different charter flights (trips) during any time frame. However, for billing purposes, each charter trip is reserved by one and only one customer. Some of RC_Charter2's customers do not use the company's charter operations; instead, they purchase fuel, use maintenance services, or use other RC_Charter2 services. However, this database design will focus on the charter operations only.

Each charter trip yields revenue for the RC_Charter2 Company. That revenue is generated by the charges that a customer pays upon the completion of a flight. The charter flight charges are a function of aircraft model used, distance flown, waiting time, special customer requirements, and crew expenses. The distance flown charges are computed by multiplying the round-trip miles by the model's charge per mile. Round-trip miles are based on the actual navigational path flown. The sample route traced in Figure P4.10 illustrates the procedure. Note that the number of round-trip miles is calculated to be 130 + 200 + 180 + 390 = 900.

FIGURE P4.11a ROUND-TRIP MILE DETERMINATION



Depending on whether a customer has RC_Charter2 credit authorization, the customer may:

- **Pay the entire charter bill upon the completion of the charter flight.**

- **Pay a part of the charter bill and charge the remainder to the account. The charge amount may not exceed the available credit.**
- **Charge the entire charter bill to the account. The charge amount may not exceed the available credit.**

Customers may pay all or part of the existing balance for previous charter trips. Such payments may be made at any time and are not necessarily tied to a specific charter trip. The charter mileage charge includes the expense of the pilot(s) and other crew required by FAR 135. However, if customers request additional crew not required by FAR 135, those customers are charged for the crew members on an hourly basis. The hourly crew-member charge is based on each crew member's qualifications.

The database must be able to handle crew assignment. Each charter trip requires the use of an aircraft, and a crew flies each aircraft. The smaller piston engine-powered charter aircraft require a crew consisting of only a single pilot. Larger aircraft (that is, aircraft having a gross takeoff weight of 12,500 pounds or more) and jet-powered aircraft require a pilot and a copilot, while some of the larger aircraft used to transport passengers may require flight attendants as part of the crew. Some of the older aircraft require the assignment of a flight engineer, and larger cargo-carrying aircraft require the assignment of a loadmaster. In short, a crew can consist of more than one person and not all crew members are pilots.

The charter flight's aircraft waiting charges are computed by multiplying the hours waited by the model's hourly waiting charge. Crew expenses are limited to meals, lodging, and ground transportation.

The RC_Charter2 database must be designed to generate a monthly summary of all charter trips, expenses, and revenues derived from the charter records. Such records are based on the data that each pilot in command is required to record for each charter trip: trip date(s) and time(s), destination(s), aircraft number, pilot (and other crew) data, distance flown, fuel usage, and other data pertinent to the charter flight. Such charter data are then used to generate monthly reports that detail revenue and operating cost information for customers, aircraft, and pilots. All pilots and other crew members are RC_Charter2 Company employees; that is, the company does not use contract pilots and crew.

FAR Part 135 operations are conducted under a strict set of requirements that govern the licensing and training of crew members. For example, pilots must have earned either a Commercial license or an Airline Transport Pilot (ATP) license. Both licenses require appropriate ratings. Ratings are specific competency requirements. For example:

- **To operate a multiengine aircraft designed for takeoffs and landings on land only, the appropriate rating is MEL, or Multiengine Landplane. When a multiengine aircraft can take off and land on water, the appropriate rating is MES, or Multiengine Seaplane.**
- **The instrument rating is based on a demonstrated ability to conduct all flight operations with sole reference to cockpit instrumentation. The instrument rating is required to operate an aircraft under Instrument Meteorological Conditions (IMC), and all such operations are governed under FAR-specified Instrument Flight Rules (IFR). In contrast,**

operations conducted under "good weather" or visual flight conditions are based on the FAR Visual Flight Rules (VFR).

- **The type rating is required for all aircraft with a takeoff weight of more than 12,500 pounds or for aircraft that are purely jet-powered. If an aircraft uses jet engines to drive propellers, that aircraft is said to be turboprop-powered. A turboprop—that is, a turbo propeller-powered aircraft—does not require a type rating unless it meets the 12,500-pound weight limitation.**

Although pilot licenses and ratings are not time-limited, exercising the privilege of the license and ratings under Part 135 requires both a current medical certificate and a current Part 135 checkride. The following distinctions are important:

- **The medical certificate may be Class I or Class II. The Class I medical is more stringent than the Class II, and it must be renewed every six months. The Class II medical must be renewed yearly. If the Class I medical is not renewed during the six-month period, it automatically reverts to a Class II certificate. If the Class II medical is not renewed within the specified period, it automatically reverts to a Class III medical, which is not valid for commercial flight operations.**
- **A Part 135 checkride is a practical flight examination that must be successfully completed every six months. The checkride includes all flight maneuvers and procedures specified in Part 135.**

Nonpilot crew members must also have the proper certificates in order to meet specific job requirements. For example, loadmasters need an appropriate certificate, as do flight attendants. In addition, crew members such as loadmasters and flight attendants, who may be required in operations that involve large aircraft (more than a 12,500-pound. takeoff weight and passenger configurations over 19) are also required periodically to pass a written and practical exam. The RC_Charter2 Company is required to keep a complete record of all test types, dates, and results for each crew member, as well as pilot medical certificate examination dates.

In addition, all flight crew members are required to submit to periodic drug testing; the results must be tracked, too. (Note that nonpilot crew members are not required to take pilot-specific tests such as Part 135 checkrides. Nor are pilots required to take crew tests such as loadmaster and flight attendant practical exams.) However, many crew members have licenses and/or certifications in several areas. For example, a pilot may have an ATP and a loadmaster certificate. If that pilot is assigned to be a loadmaster on a given charter flight, the loadmaster certificate is required. Similarly, a flight attendant may have earned a commercial pilot's license. Sample data formats are shown in Table P4.13.

TABLE P4.11 SAMPLE DATA FORMATS

Part A Tests

TEST CODE	TEST DESCRIPTION	TEST FREQUENCY
1	Part 135 Flight Check	6 months
2	Medical, Class 1	6 months
3	Medical, Class 2	12 months
4	Loadmaster Practical	12 months
5	Flight Attendant Practical	12 months
6	Drug test	Random
7	Operations, written exam	6 months

Part B Results

EMPLOYEE	TEST CODE	TEST DATE	TEST RESULT
101	1	12-Nov-13	Pass-1
103	6	23-Dec-13	Pass-1
112	4	23-Dec-13	Pass-2
103	7	11-Jan-14	Pass-1
112	7	16-Jan-14	Pass-1
101	7	16-Jan-14	Pass-1
101	6	11-Feb-14	Pass-2
125	2	15-Feb-14	Pass-1

Part C Licenses and Certificates

LICENSE OR CERTIFICATE	LICENSE OR CERTIFICATE DESCRIPTION
ATP	Airline Transport Pilot
Comm	Commercial license
Med-1	Medical certificate, class 1
Med-2	Medical certificate, class 2
Instr	Instrument rating
MEL	Multiengine Land aircraft rating
LM	Loadmaster
FA	Flight Attendant

Part D Licenses and Certificates Held by Employees

EMPLOYEE	LICENSE OR CERTIFICATE	DATE EARNED
101	Comm	12-Nov-93
101	Instr	28-Jun-94
101	MEL	9-Aug-94
103	Comm	21-Dec-95
112	FA	23-Jun-02
103	Instr	18-Jan-96
112	LM	27-Nov-05

Pilots and other crew members must receive recurrency training appropriate to their work assignments. Recurrency training is based on an FAA-approved curriculum that is job-specific. For example, pilot recurrency training includes a review of all applicable Part 135 flight rules and regulations, weather data interpretation, company flight operations requirements, and specified flight procedures. The RC_Charter2 Company is required to keep a complete record of all recurrency training for each crew member subject to the training.

The RC_Charter2 Company is required to maintain a detailed record of all crew credentials and all training mandated by Part 135. The company must keep a complete record of each requirement and of all compliance data.

To conduct a charter flight, the company must have a properly maintained aircraft available. A pilot who meets all of the FAA's licensing and currency requirements must fly the aircraft as Pilot in Command (PIC). For those aircraft that are powered by piston engines or turboprops and have a gross takeoff weight under 12,500 pounds, single-pilot operations are permitted under Part 135 as long as a properly maintained autopilot is available. However, even if FAR Part 135 permits single-pilot operations, many customers require the presence of a copilot who is capable of conducting the flight operations under Part 135.

The RC_Charter2 operations manager anticipates the lease of turbojet-powered aircraft, and those aircraft are required to have a crew consisting of a pilot and copilot. Both pilot and copilot must meet the same Part 135 licensing, ratings, and training requirements.

The company also leases larger aircraft that exceed the 12,500-pound gross takeoff weight. Those aircraft can carry the number of passengers that requires the presence of one or more flight attendants. If those aircraft carry cargo weighing over 12,500 pounds, a loadmaster must be assigned as a crew member to supervise the loading and securing of the cargo. The database must be designed to meet the anticipated additional charter crew assignment capability.

a. Given this incomplete description of operations, write all applicable business rules to establish entities, relationships, optionalities, connectivities, and cardinalities. (Hint: Use the following five business rules as examples, writing the remaining business rules in the same format.)

- **A customer may request many charter trips.**

- **Each charter trip is requested by only one customer.**
- **Some customers have not (yet) requested a charter trip.**
- **An employee may be assigned to serve as a crew member on many charter trips.**
- **Each charter trip may have many employees assigned to it to serve as crew members.**

b. Draw the fully labeled and implementable Crow's Foot ERD based on the business rules you wrote in Part a of this problem. Include all entities, relationships, optionalities, connectivities, and cardinalities.

The following business rules can be derived from the description of operations:

- A customer may request many charter trips.
- Each charter trip is requested by only one customer.
- Some customers have not (yet) requested a charter trip.
- Every charter trip is requested by at least one customer.
- An employee may be assigned to serve as a crew member on many charter trips.
- Each charter trip may have many employees assigned to it to serve as crew members.
- An employee may not yet have been assigned to serve as a crew member on any charter trip.
- A charter trip may not yet have any employee assigned to serve as a crew member.
- Each customer may make many payments.
- Some customers have not made any payments yet.
- Every payment is made by only one customer.
- Every payment must have been made by a customer.
- A payment may be toward many charter trips.
- A payment may not be in reference to any charter trip.
- Every charter trip must have a payment made.
- Each charter trip has only one payment.
- Every charter trip involves the use of a single aircraft.
- Every charter trip requires at least one aircraft.
- An aircraft may be used for many charter trips.
- An aircraft may not yet have been used for any charter trip.
- Each aircraft is only one model airplane.
- Every aircraft has a model designation.
- An airplane model is not required to be associated with any aircraft that the company owns.
- The company may own many aircraft of a given model.
- A given flight assignment may be given to many crew members.
- Some flight assignments may not have ever been given to any crew member.
- Every crew member assignment is associated with a flight assignment.
- Every crew member assignment is associated with only one flight assignment.
- An employee may have taken many tests.
- Some employees may have taken no tests yet.
- A test may be taken by many employees.
- A test may not have been taken by any employee yet.
- Each employee has one job with the company.
- Every employee has only one job with the company.
- A job may be done by many employees.

- A job may be currently unfilled and not be associated with any employee.
- An employee may be a pilot, and every pilot is an employee.
- A pilot may have earned many ratings.
- Some pilots have not earned any rating yet.
- A rating may be earned by many pilots.
- Some ratings are not held by any pilots.
- A pilot may have many licenses.
- A pilot may not have any license yet.
- A license may be held by many pilots.
- A license may not be held by any pilot yet.
- Every employee can have many qualifications.
- Some employees do not have any qualifications.
- Each qualification can be held by many employees.
- Some qualifications are not held by any employee.

The completed ERD is shown in Figure P4.11b.

Figure P4.11b The RC_Charter2 Flight Department Crow's Foot ERD