Chapter 5

Data Modelling With Entity Relationship Diagrams

Selected 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 5.10 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 5.10.)

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 5.8, 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 5.8.) **However, regardless of how the designer classifies the relationship – weak or strong – CLASS is always existence-dependent on COURSE**.

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:* 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

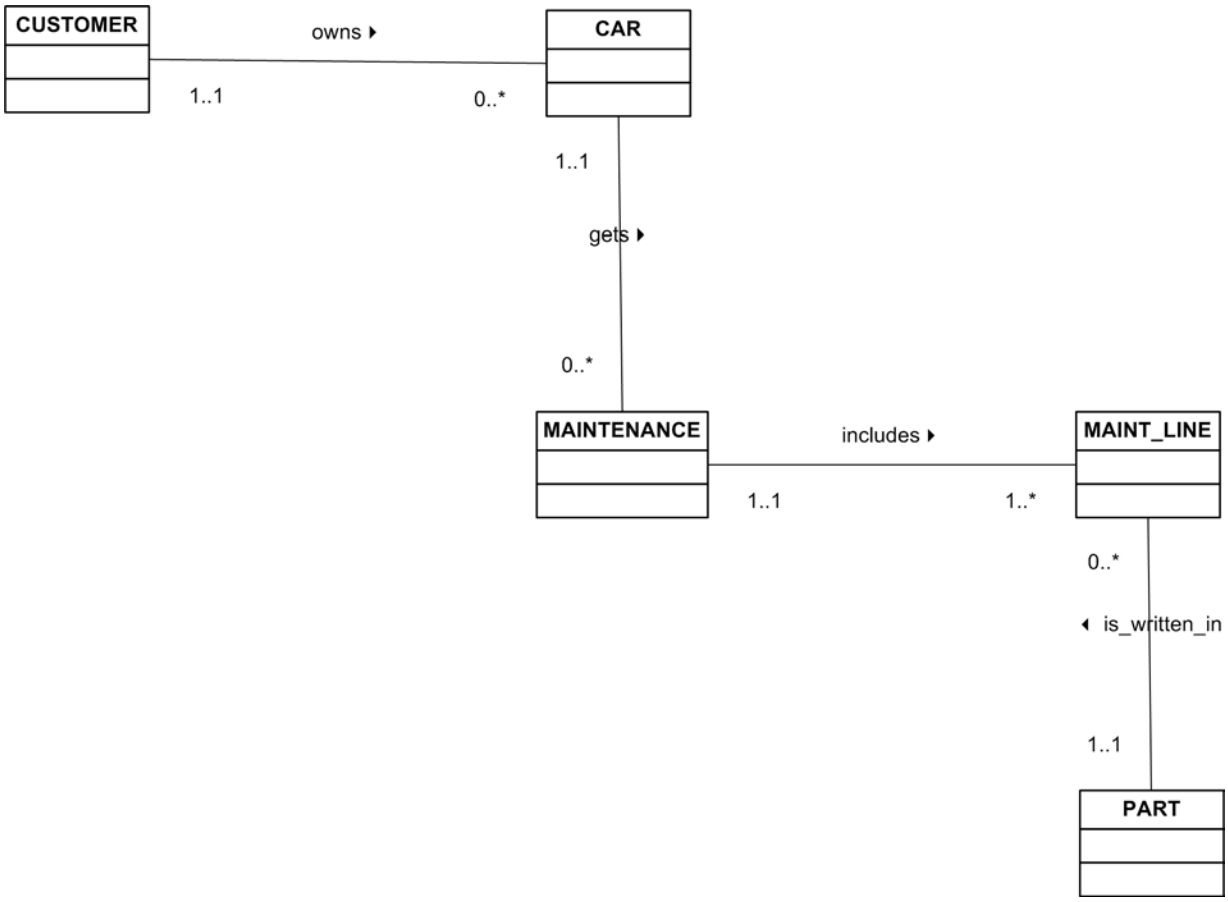
Table name: DEGREE

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.)

5. Suppose you are working within the framework of the conceptual model in Figure Q5.5.

Figure Q5.5 The Conceptual Model for Question 5



Given the conceptual model in Figure Q5.5:

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

Even a simple ERD such as the one shown in Figure Q5.5 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 Q5.5, 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 *:* 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 *:* relationship between MAINTENANCE and PART has been broken up to produce the two 1:* 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-2006
10002	15-Mar-2006
10003	16-Mar-2006

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

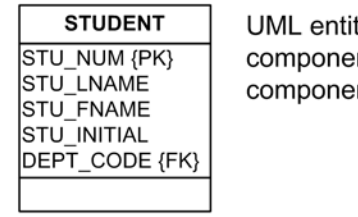
b. Identify all of the cardinalities.

The Visio-generated UML ERD, shown in Figure Q4=5.5, shows the cardinalities directly.

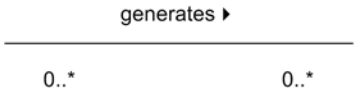
7. How would you (graphically) identify each of the following ERM components in a UML model?

The answers to questions (a) through (b) are illustrated with the help of Figure Q5.7.

FIGURE Q5.7 UML ERM Components



UML entity box showing the attribute component and the empty method component.



Visually representing a 0:* multiplicity

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 UML ERD – as represented in Visio Professional – does not distinguish among the various entity types such as weak entities and composite entities. Instead, the UML 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 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.

.

9. What two courses of action are available to a designer when he or she encounters a multivalued attribute?

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

11. How is a relationship between entities indicated in an ERD? Give an example, using the UML notation.

Use Figure Q5.7 as the basis for your answer.

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 *:* relationships – and the composite entity can be used to break up such relationships into 1:* 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 Q512b is far better than that of the CLASS entity structure shown in Figure Q512a. Suppose, for example, that you want to design a class enrollment entity to serve as the "bridge" between STUDENT and CLASS in the *:* 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 *:* 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-2013	322453	1215	07:18 a.m.	04:23 p.m.
10016	12-Mar-2013	322453	1215	07:18 a.m.	04:23 p.m.
10017	12-Mar-2013	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 *:* relationship is converted to a set of two 1:* 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.

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.

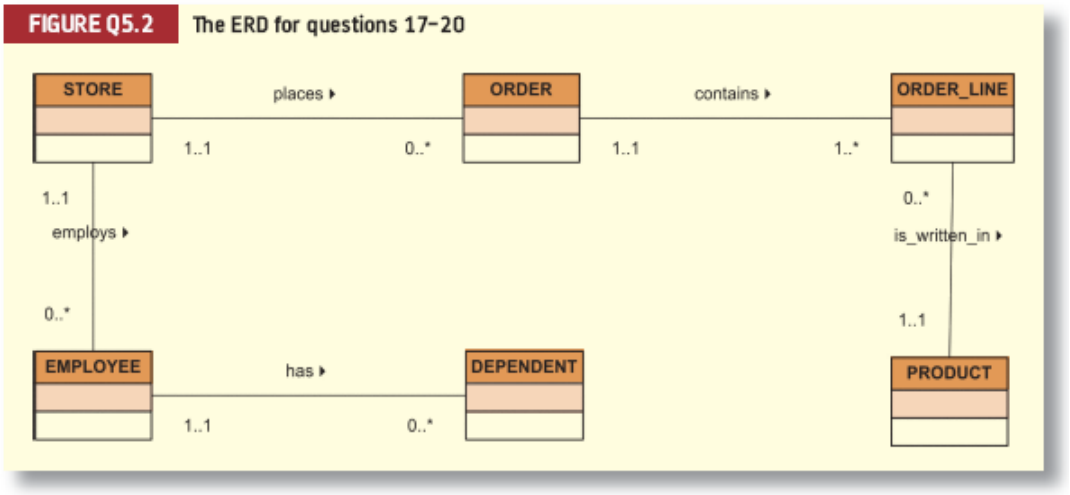
A simple attribute is one that cannot be decomposed into its component pieces. For example, a person's sex is classified as either M or F and there is no reasonable way to decompose M or F. Similarly, a person's first name cannot be decomposed into meaningful components. (In contrast, if a phone number 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 UKS2M231109154321 might be decomposed this way:

UK country = United Kingdom
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.

The final four questions are based on the ERD in Figure Q5.2.

FIGURE Q5.2 The ERD For Questions 17−20



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

The cardinalities are indicated in the Figure Q5.2 and should be explained by the student. They should be able to read the direction of the relationship both ways.

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 Q5.2, note that an order is represented by two entities, ORDER and ORDER_LINE. Note also that the STORE's 1:* relationship with ORDER and the ORDER's 1:* relationship with ORDER_LINE reflect the conceptual *:* relationship between STORE and PRODUCT. The original business rules probably read:

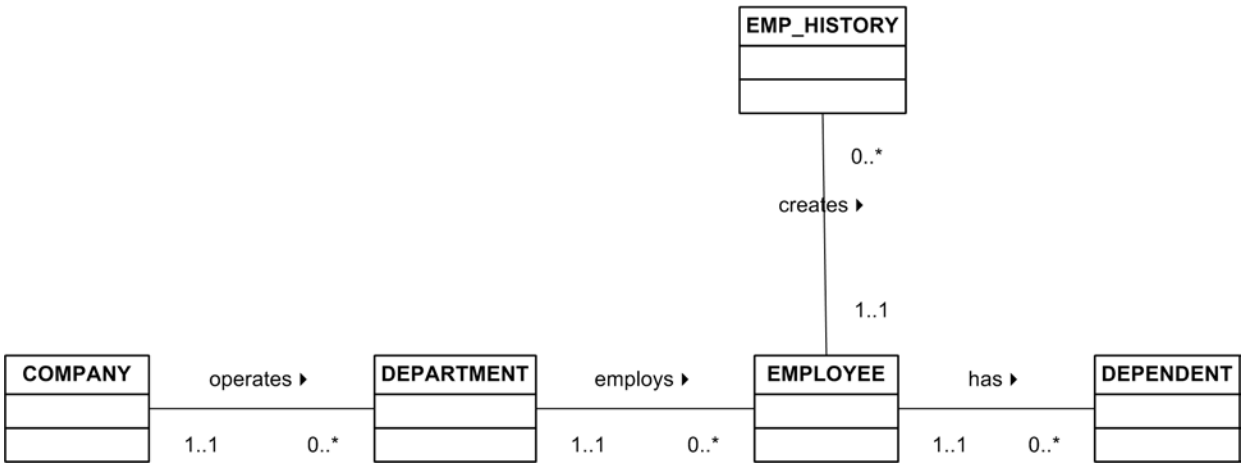
- A store can order many products
- A product can be ordered by many stores.

Problem Solutions

1. **Given the following business rules, create the appropriate ERD using UML notation.**
   a. **A company operates many departments.**
   b. **Each department employs one or more employees.**
   c. **Each of the employees may or may not have one or more dependents.**
   d. **Each employee may or may not have an employment history.**

   The solution is presented in Figure P5.1.

FIGURE P5.1 Solution to Problem 5.1

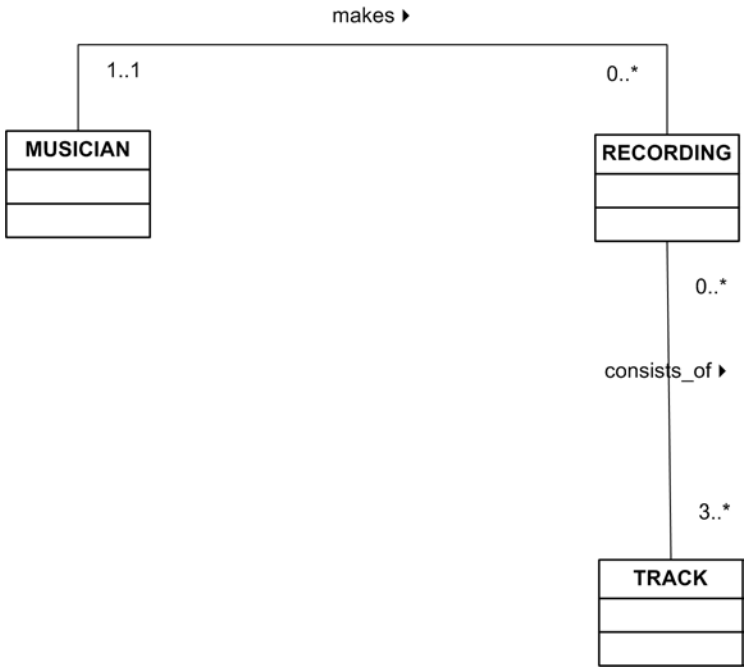


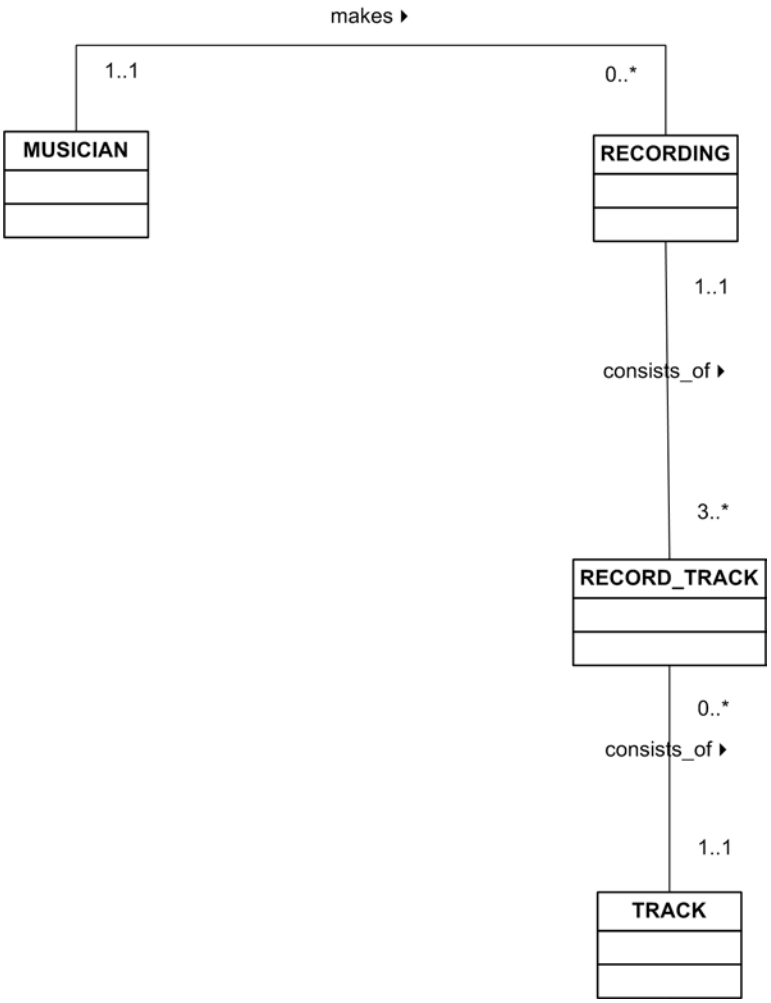
3. Given the following business rules, create an initial ERD using UML notation.

   a. **A musician makes at least one recording, but may over a period of time make many recordings.**
   b. **One recording consists if at least 3 or more tracks.**
   c. **A track can appear on more than one recording**.

The solution is presented in Figure P5.3.

FIGURE P5.3 Solution to Problem 5.3





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 10 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 5.4 as your guide.)**

The HEG entities and relationships are shown in Table P5.5a.

Table P5.5a The Components of the HEG ERD

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

As you examine the summary in Table P5.5a, 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.)

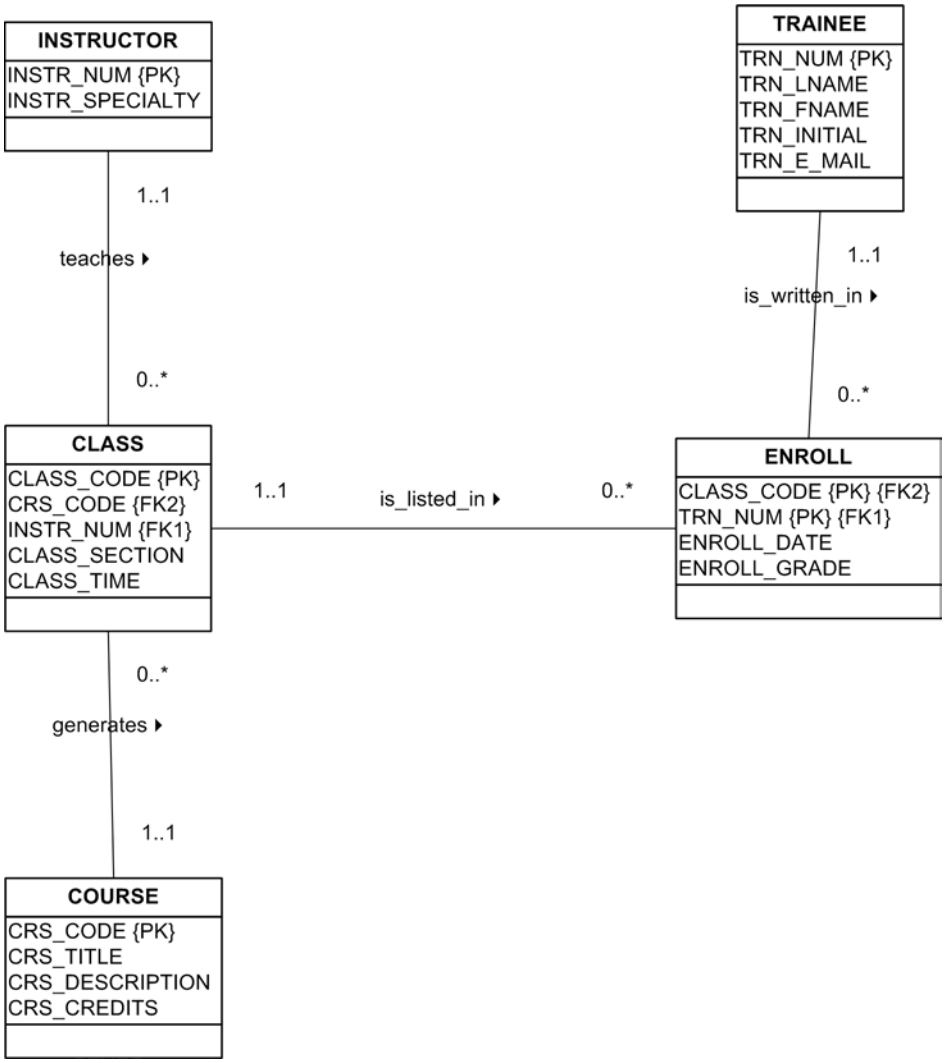
- A COURSE does not necessarily generate a class during each training period. (Some courses may be taught every other period or during some other specified time frames. Therefore, it is reasonable to assume that CLASS is optional to COURSE.
- Each CLASS must be related to a COURSE. (The class must cover 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 P5.5b.

Figure P5.5b The HEG ERD



As you discuss Figure P5.5b, 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 *:* 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:* 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:* relationship between COURSE and CLASS.

These relationships are all reflected in the ER diagram shown in Figure P5.2b. 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 2 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:*, or *:* – 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. 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 P5.5b, note also that the ENROLL entity in Figure P5.2b 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 Section 5.1.6 in 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.

7. 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 worked, one additional job history record was 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 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, (s)he is given the job 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 filed 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 P5.7.**

TABLE P5.7 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:

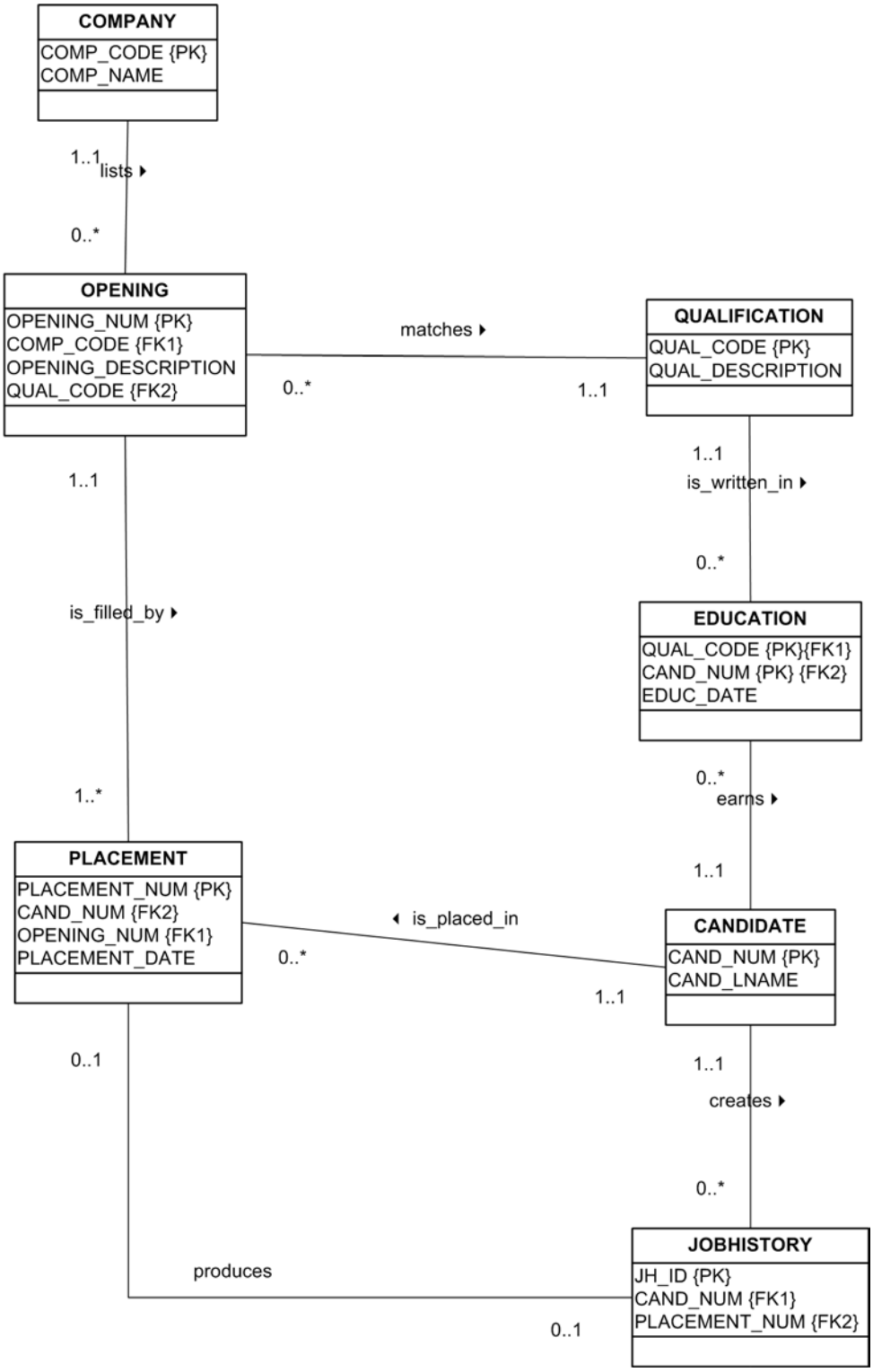
- **COMPANY**
- **OPENING**
- **QUALIFICATION**
- **CANDIDATE**
- **JOB_HISTORY**
- **PLACEMENT**

Given that information, do the following:

a. Draw the ERD using UML notation 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 *:* relationships.

The solutions for problems 7a-7e are shown in Figure P5.7.

Figure P5.7 The TEC ERD



To help the students understand Figure P5.7'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:* 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 P5.5a. (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. However, when you use Visio to generate the 1:1 relationship between PLACEMENT and HISTORY, with PLACEMENT optional to HISTORY, Visio will automatically set HISTORY optional to PLACEMENT, too. This action reflects the argument 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 *:*. This relationship must be decomposed into two 1:* 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**.

- Each job OPENING requires one QUALIFICATION, and any given qualification may fit many openings, thus producing a 1:* 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 P5.7, 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 *:*. We will decompose this *:* relationship into two 1:* 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.

9. 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.

When 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.

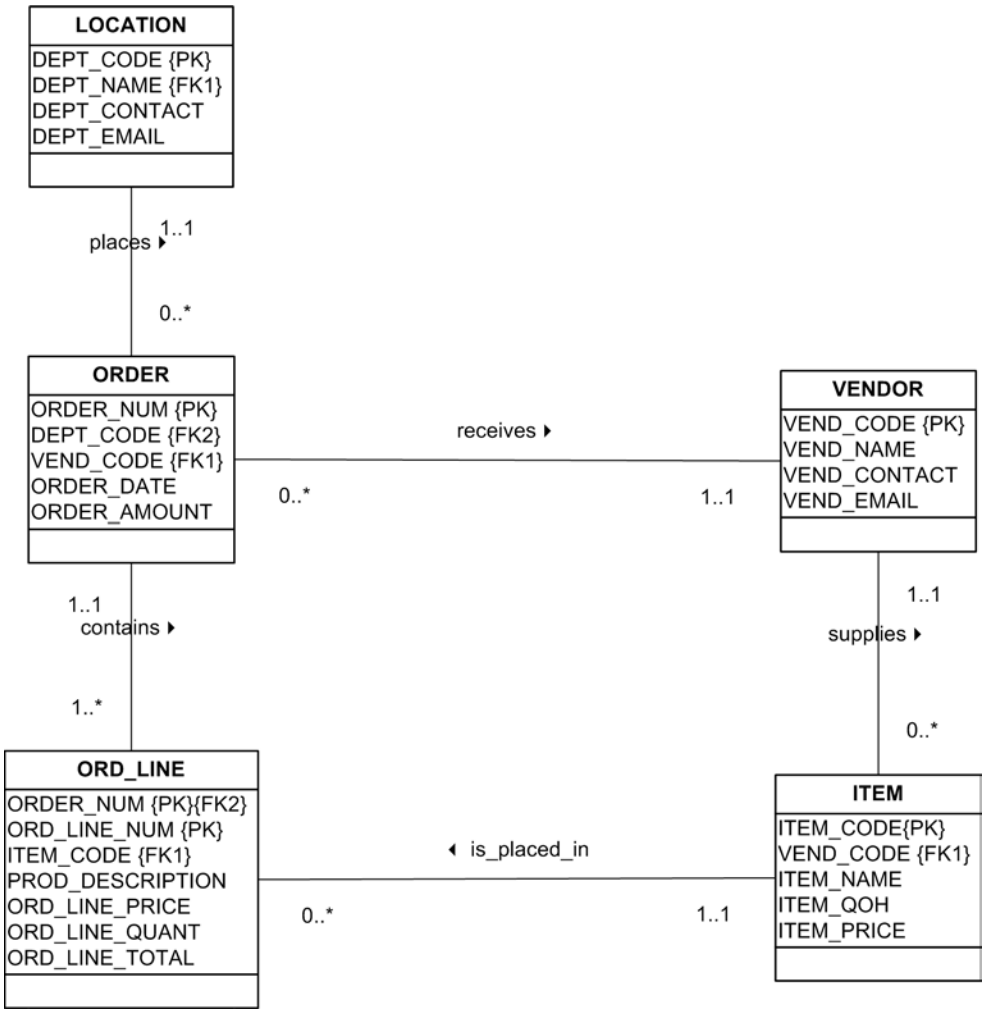
The order received by the purchasing department may contain several different items. An inventory is maintained so that the most frequently requested items are delivered almost immediately. When an order comes in, it is checked to determine whether the requested item is in inventory. If an item is not 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 multiplicities among entities.
c. Identify the type of existence dependency in all relations.
d. Give at least two examples of the types of reports that can be obtained from the database.

The initial UML ERD is shown in Figure P5.9. The discussion preceding Figure P5.9rev explains why the revision was made.

Figure P5.9 Initial Automata ERD



As you explain the development of the ERD shown in Figure P5.9, 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:* 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.

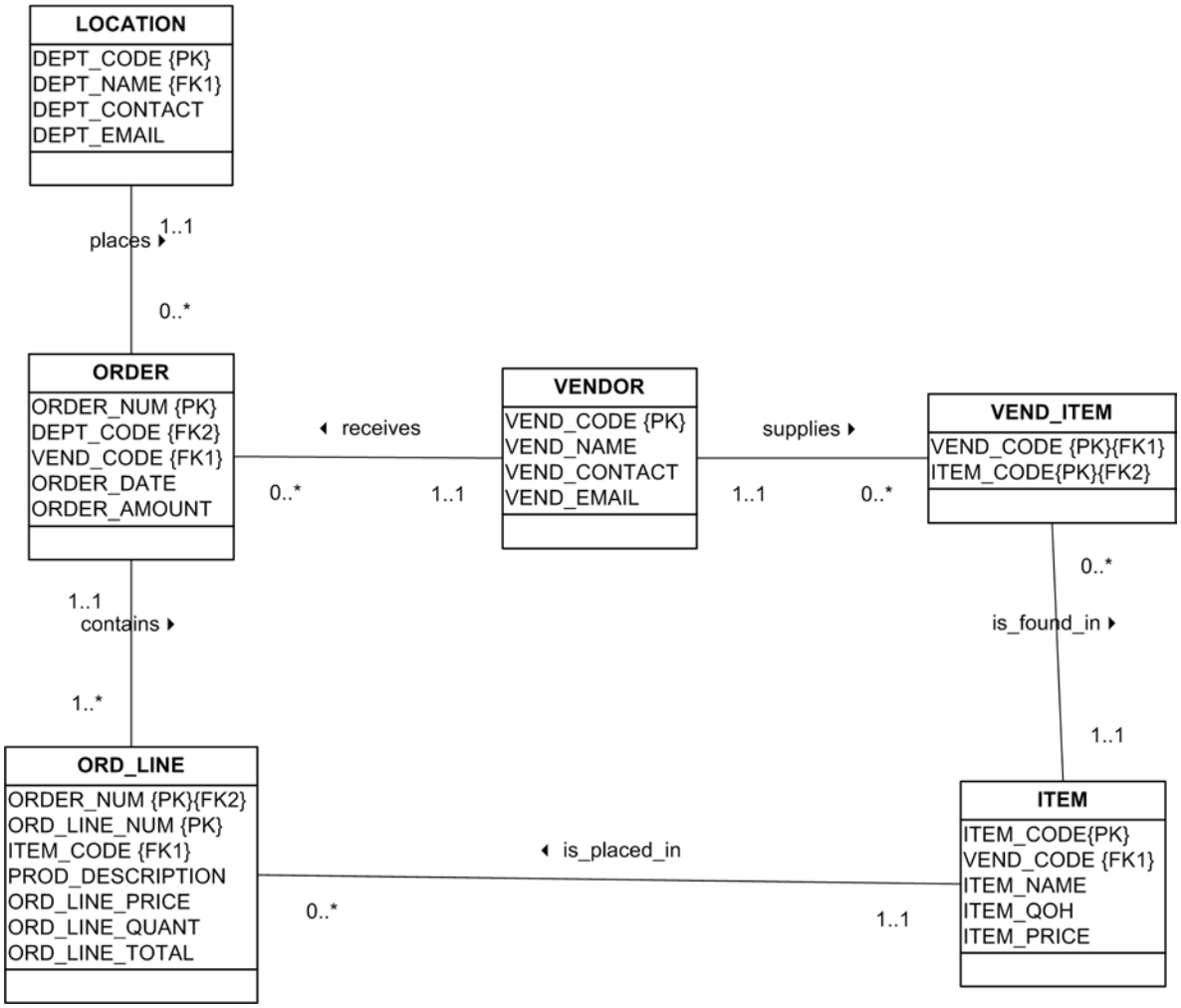
- Note also that the VENDOR may not (yet) have received and order, so ORDER is optional to VENDOR. The VENDOR entity may contain vendors who are simply *potential* suppliers if items and you may want to have such potential vendors available just in case your "usual" vendor(s) run(s) out of items that 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:*. 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 *:* relationship between VENDOR and ITEM and this relationship would have to be implemented through a composite (bridge) entity. Actually, such an *:* relationship is specified in the brief description of the Automata company's data environment. Therefore, the following Figure P5.9rev more accurately reflects the problem description.

Figure P5.9rev Revised Automata ERD



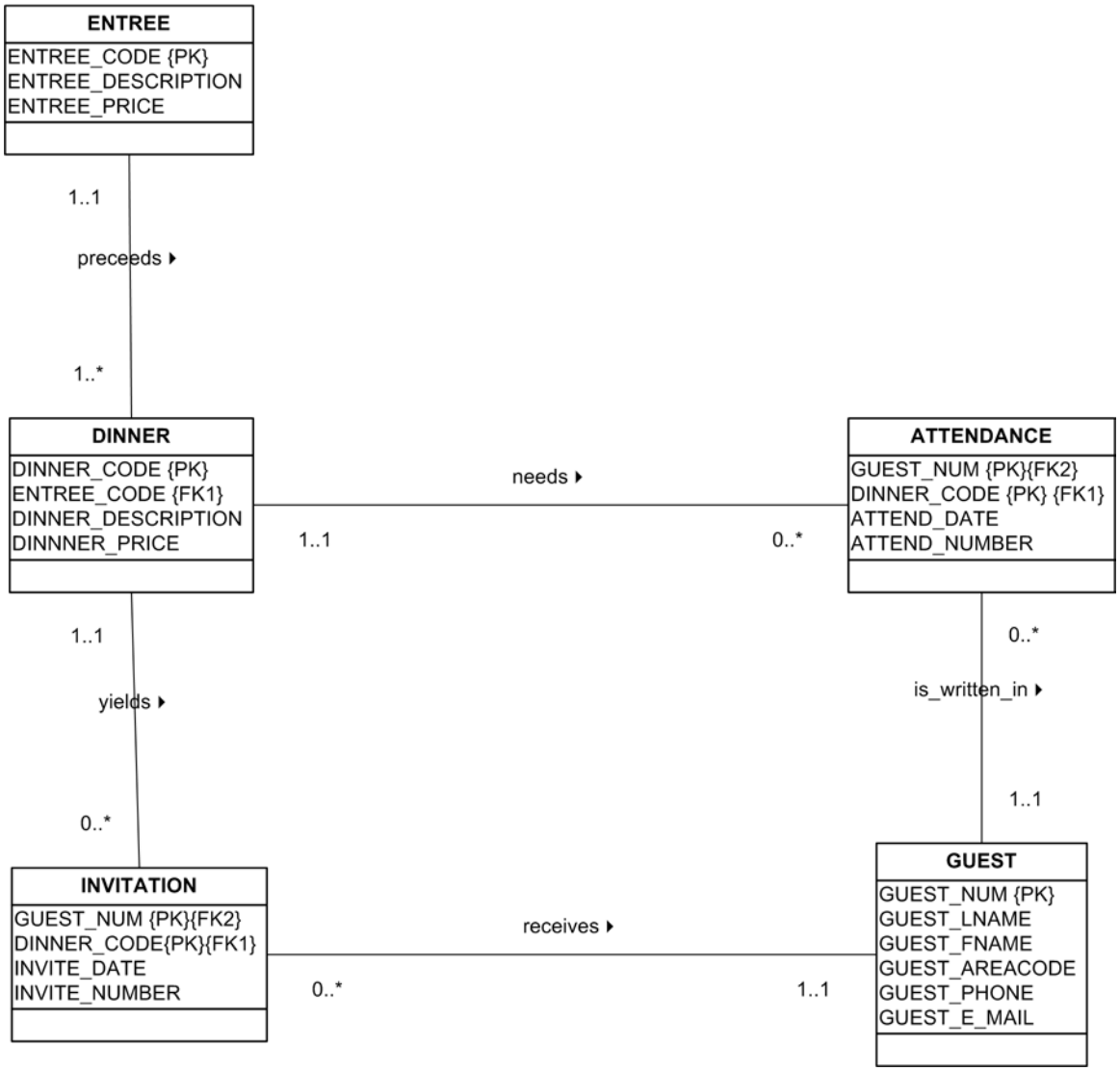
11. Given the following brief summary of business rules for the ROBCOR catering service, draw the fully labeled ERD. Make sure you include all appropriate entities, relationships, connectivities, and cardinalities.

> **NOTE**
>
> **Limit your ERD to entities and relationships based on the business rules shown here. In other words, do *not* add "realism" to your design by expanding or refining the business rules. However, make sure you include the attributes that would permit the model to be successfully implemented.**

Each dinner is based on a single entree, but each entree can be served at many dinners. A guest can attend many dinners, and each dinner can be attended by many guests. Each dinner invitation can be mailed to many guests, and each guest can receive many invitations.

The UML ERD solution is shown in Figure P5.11. (The problem note limits this design. In previous problems, there was some flexibility to enhance the basic components.)

Figure P5.11 The ROBCOR Catering Service ERD



13. Tiny University is so pleased with your design and implementation of its student registration/tracking system that it wants you to expand the design to include its motor pool.

A brief description of operations follows:

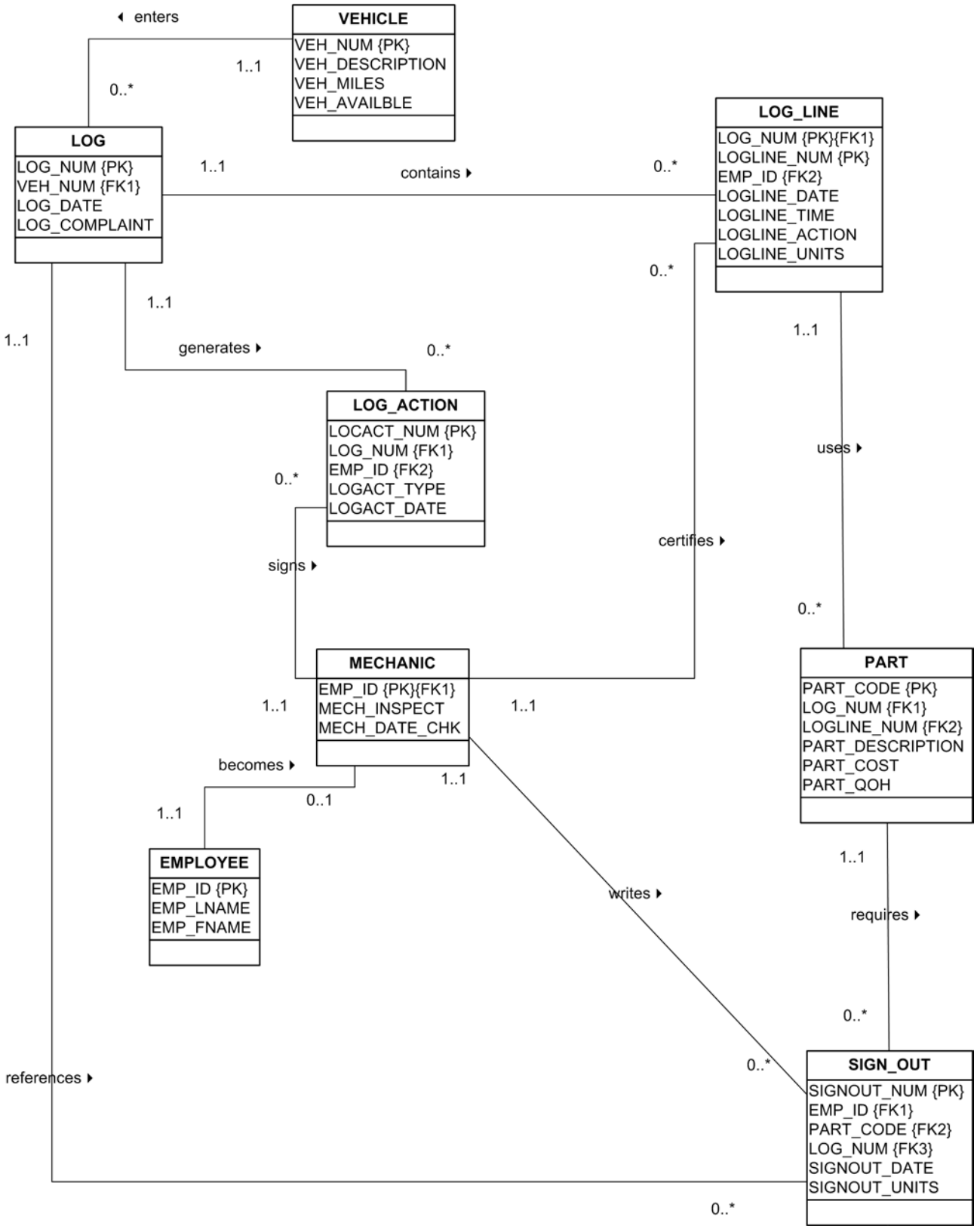
- **Faculty members may use the vehicles owned by Tiny University for officially sanctioned travel. For example, the vehicles may be used by faculty members to travel to off-campus learning centers, to travel to locations at which research papers are presented, to transport students to officially sanctioned locations, and to travel for public service purposes. The vehicles used for such purposes are managed by Tiny University'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. When the faculty member arrives to pick up the vehicle, (s)he 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 UML ERD notation to indicate entities, relationships, and multiplicities.

The solution is shown in Figure P5.13.

Figure P5.13 The Tiny-University TFBS Maintenance ERD



15. Use the following descriptions of the operations of RC_Models Company to complete this exercise.

RC_Models Company sells its products—plastic models (aircraft, ships, and cars) and "add-on" decals for those models—through its Internet Web site (*www.rc_models.com*). Models and decals are available in scales that vary from 1/144 to 1/32.

Customers use the Web site to select the products and to pay by credit card. If a product is not currently available, it is placed on back order at the customer's discretion. (Back orders are not charged to a customer until the order is shipped.) When a customer completes his/her transactions, the invoice is printed and the products listed on the invoice are pulled from inventory for shipment. (The invoice includes a shipping charge.) The printed invoice is enclosed in the shipping container. The customer credit card charges are transmitted to the CC Bank, at which RC_Models Company maintains a commercial account. (*Note:* The CC Bank is *not* part of the RC_Models database.)

RC_Models Company tracks customer purchases and periodically sends out promotional materials. Because management at RC_Models Company requires detailed information to conduct its operations, numerous reports are available. Those reports include, but are not limited to, customer purchases by product category and amount, product turnover, and revenues by product and customer. If a product has not recorded a sale within four weeks of being stocked, it is removed from inventory and scrapped.

Many of the customers on the RC_Models customer list have bought RC_Models products. However, RC_Models Company also has purchased a copy of the *FineScale Modeler* magazine subscription list to use in marketing its products to customers who have not yet bought from RC_Models Company. In addition, customer data are recorded when potential customers request product information.

RC_Models Company orders its products directly from the manufacturers. For example, the plastic models are ordered from Tamiya, Academy, Revell/Monogram, and others. Decals are ordered from Aeromaster, Tauro, WaterMark, and others. (*Note:* Not all manufacturers in the RC_Models Company database have received orders.) All orders are placed via the manufacturers' Web sites, and the order amounts are automatically handled through RC_Models' commercial bank account with the CC Bank. Orders are automatically placed when product inventory reaches the specified minimum quantity on hand. (The number of product units ordered depends on the minimum order quantity specified for each product.)

a. Given that brief and incomplete description of operations for RC_Models Company, write all applicable business rules to establish entities, relationships, optionalities, and multiplicities. (*Hint:* Use the following three business rules as examples, writing the remaining business rules in the same format.)
   - **A customer may generate many invoices.**

- **Each invoice is generated by only one customer.**
- **Some customers have not (yet) generated an invoice.**

b. Draw the fully labeled and implementable ERD based on the business rules you wrote in Part a of this problem. Include all entities, relationships, optionalities, and multiplicities.

The solution is shown in Figure P5.15.

Figure P5.15 The RC Models UML ERD