C H A P T E R 2

Entity Relationship Model

This chapter introduces the entity-relationship model in detail. The chapter covers numerous features of the model, several of which can be omitted depending on the planned coverage of the course. Weak entity sets (Section 2.6), design constraints (Section 2.7.4) and aggregation (Section 2.7.5), and the corresponding subsections of Section 2.9 (Reduction of an E-R Schema to Tables) can be omitted if time is short. We recommend covering specialization (Section 2.7.1) at least in some detail, since it is an important concept for object-oriented databases (Chapter 8).

The E-R model itself and E-R diagrams are used often in the text. It is important that students become comfortable with them. The E-R model is an excellent context for the introduction of students to the complexity of database design. For a given enterprise there are often a wide variety of E-R designs. Although some choices are arbitrary, it is often the case that one design is inherently superior to another. Several of the exercises illustrate this point. The evaluation of the goodness of an E-R design requires an understanding of the enterprise being modeled and the applications to be run. It is often possible to lead students into a debate of the relative merits of competing designs and thus illustrate by example that understanding the application is often the hardest part of database design.

Considerable emphasis is placed on the construction of tables from E-R diagrams. This serves to build intuition for the discussion of the relational model in the subsequent chapters. It also serves to ground abstract concepts of entities and relationships into the more concrete concepts of relations. Several other texts places this material along with the relational data model, rather than in the E-R model chapter. Our motivation for placing this material here is help students to appreciate how E-R data models get used in reality, while studying the E-R model rather than later on.

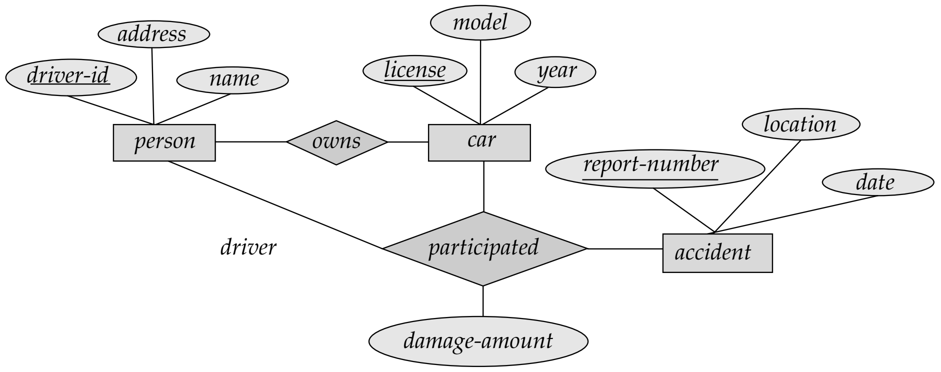

Figure 2.1 E-R diagram for a Car-insurance company.

Exercises

2.1 Explain the distinctions among the terms primary key, candidate key, and superkey.
Answer: A *superkey* is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. A superkey may contain extraneous attributes. If *K* is a superkey, then so is any superset of *K*. A superkey for which no proper subset is also a superkey is called a *candidate key*. It is possible that several distinct sets of attributes could serve as candidate keys. The *primary key* is one of the candidate keys that is chosen by the database designer as the principal means of identifying entities within an entity set.

2.2 Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
Answer: See Figure 2.1

2.3 Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.
Answer: See Figure 2.2

2.4 A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.
Answer: See Figure 2.3.
In the answer given here, the main entity sets are *student*, *course*, *course-offering*,

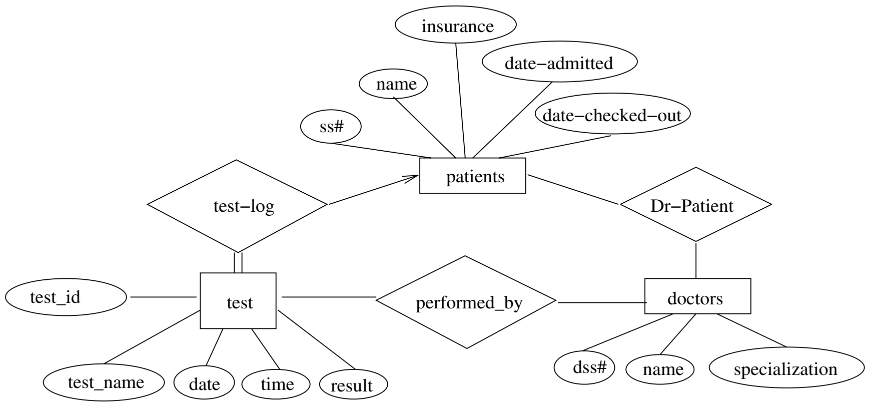

Figure 2.2 E-R diagram for a hospital.

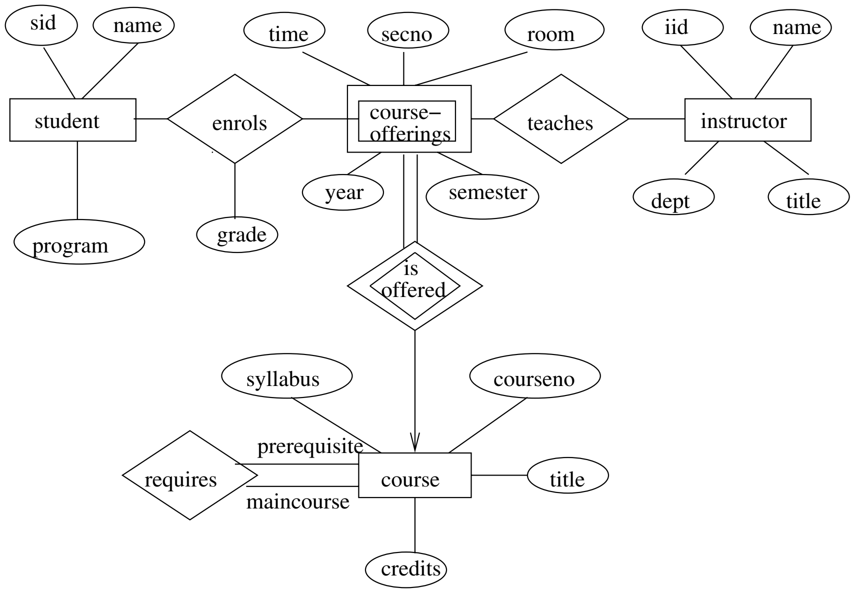

Figure 2.3 E-R diagram for a university.

and *instructor*. The entity set *course-offering* is a weak entity set dependent on *course*. The assumptions made are :

a. a class meets only at one particular place and time. This E-R diagram cannot model a class meeting at different places at different times.

b. There is no guarantee that the database does not have two classes meeting at the same place and time.

2.5 Consider a database used to record the marks that students get in different exams of different course offerings.

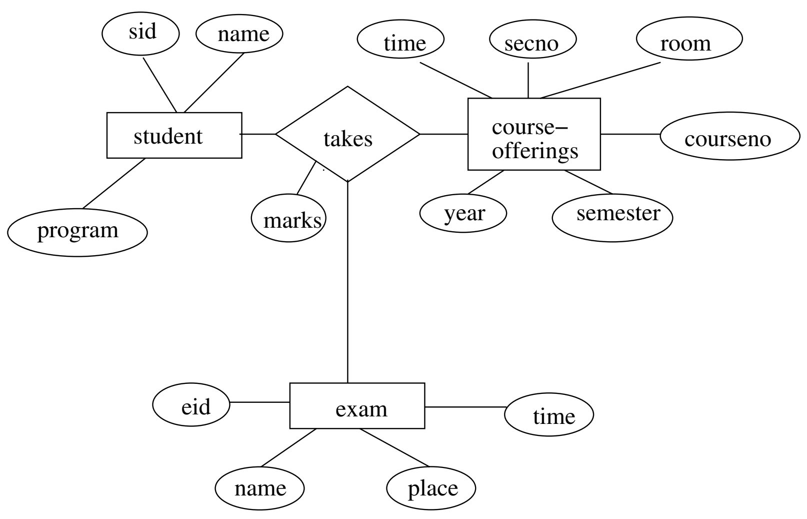

Figure 2.4 E-R diagram for marks database.

a. Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

b. Construct an alternative E-R diagram that uses only a binary relationship between *students* and *course-offerings*. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.

Answer:

a. See Figure 2.4

b. See Figure 2.5

2.6 Construct appropriate tables for each of the E-R diagrams in Exercises 2.2 to 2.4.

Answer:

a. Car insurance tables:

person (<u>driver-id</u>, name, address)
car (<u>license</u>, year, model)
accident (<u>report-number</u>, date, location)
participated(<u>driver-id</u>, <u>license</u>, <u>report-number</u>, damage-amount)

b. Hospital tables:

patients (<u>patient-id</u>, name, insurance, date-admitted, date-checked-out)
doctors (<u>doctor-id</u>, name, specialization)
test (<u>testid</u>, testname, date, time, result)
doctor-patient (<u>patient-id</u>, <u>doctor-id</u>)
test-log (<u>testid</u>, <u>patient-id</u>) performed-by (<u>testid</u>, <u>doctor-id</u>)

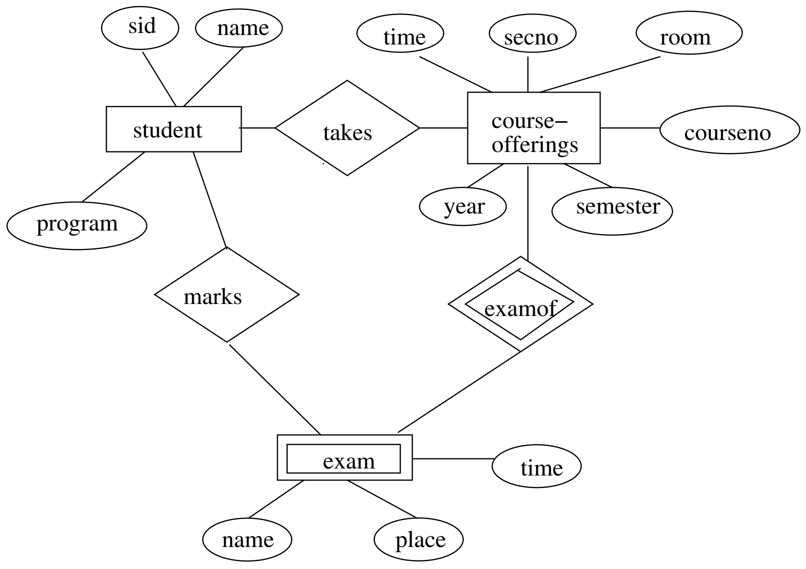

Figure 2.5 Another E-R diagram for marks database.

c. University registrar's tables:

student (<u>student-id</u>, name, program)
course (<u>courseno</u>, title, syllabus, credits)
course-offering (<u>courseno</u>, <u>secno</u>, <u>year</u>, <u>semester</u>, time, room)
instructor (<u>instructor-id</u>, name, dept, title)
enrols (<u>student-id</u>, <u>courseno</u>, <u>secno</u>, <u>semester</u>, <u>year</u>, grade)
teaches (<u>courseno</u>, <u>secno</u>, <u>semester</u>, <u>year</u>, <u>instructor-id</u>)
requires (<u>maincourse</u>, <u>prerequisite</u>)

2.7 Design an E-R diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes.
Answer: See Figure 2.6

2.8 Extend the E-R diagram of the previous question to track the same information for all teams in a league.
Answer: See Figure 2.7 Note that a player can stay in only one team during a season.

2.9 Explain the difference between a weak and a strong entity set.
Answer: A strong entity set has a primary key. All tuples in the set are distinguishable by that key. A weak entity set has no primary key unless attributes of the strong entity set on which it depends are included. Tuples in a weak entity set are partitioned according to their relationship with tuples in a strong entity

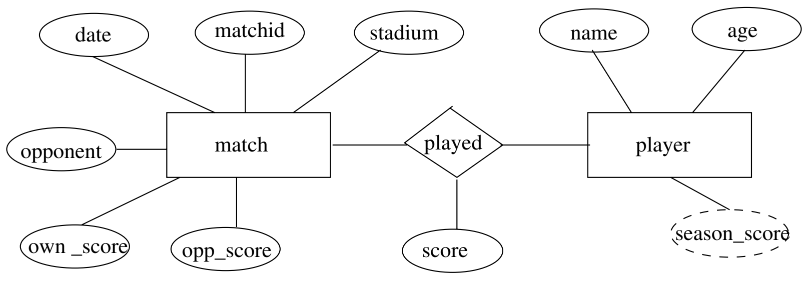

Figure 2.6 E-R diagram for favourite team statistics.

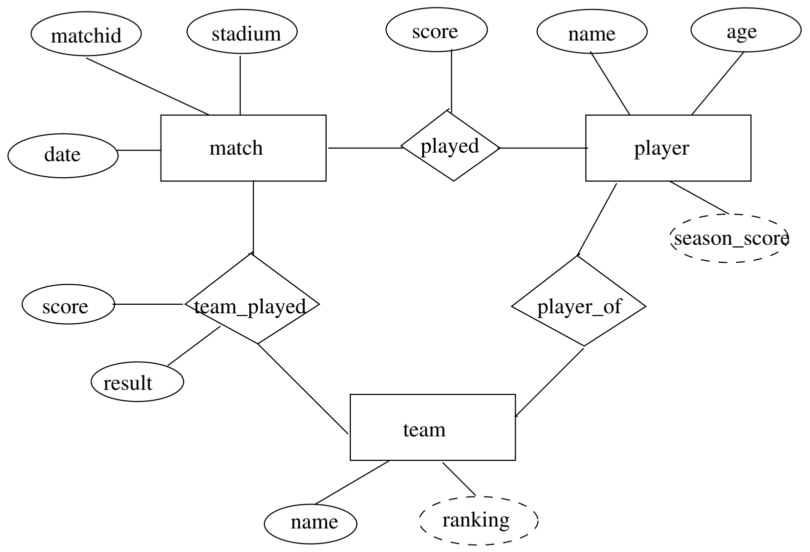

Figure 2.7 E-R diagram for all teams statistics.

set. Tuples within each partition are distinguishable by a discriminator, which is a set of attributes.

2.10 We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?
Answer: We have weak entities for several reasons:

- We want to avoid the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity.
- Weak entities reflect the logical structure of an entity being dependent on another entity.
- Weak entities can be deleted automatically when their strong entity is deleted.
- Weak entities can be stored physically with their strong entities.

2.11 Define the concept of aggregation. Give two examples of where this concept is useful.

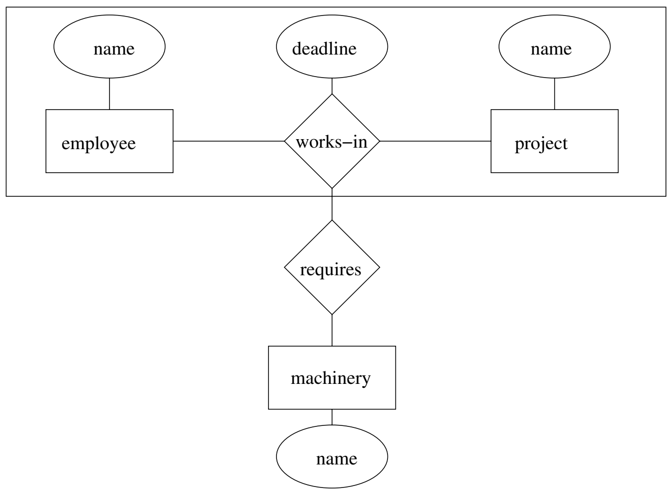

Figure 2.8 E-R diagram Example 1 of aggregation.

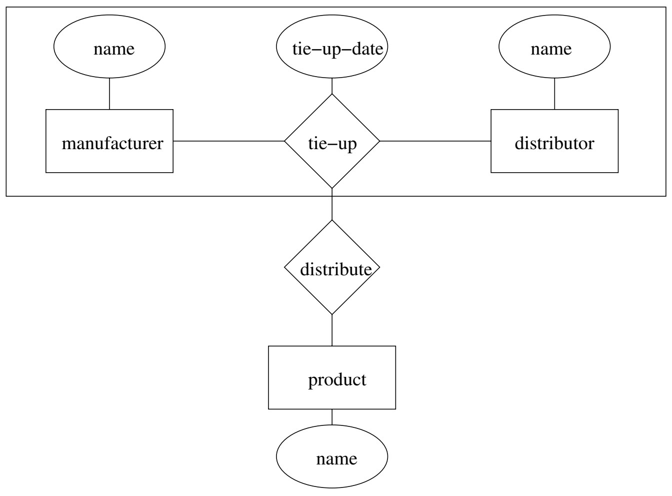

Figure 2.9 E-R diagram Example 2 of aggregation.

Answer: Aggregation is an abstraction through which relationships are treated as higher-level entities. Thus the relationship between entities A and B is treated as if it were an entity C. Some examples of this are:

a. Employees work for projects. An employee working for a particular project uses various machinery. See Figure 2.8

b. Manufacturers have tie-ups with distributors to distribute products. Each tie-up has specified for it the set of products which are to be distributed. See Figure 2.9

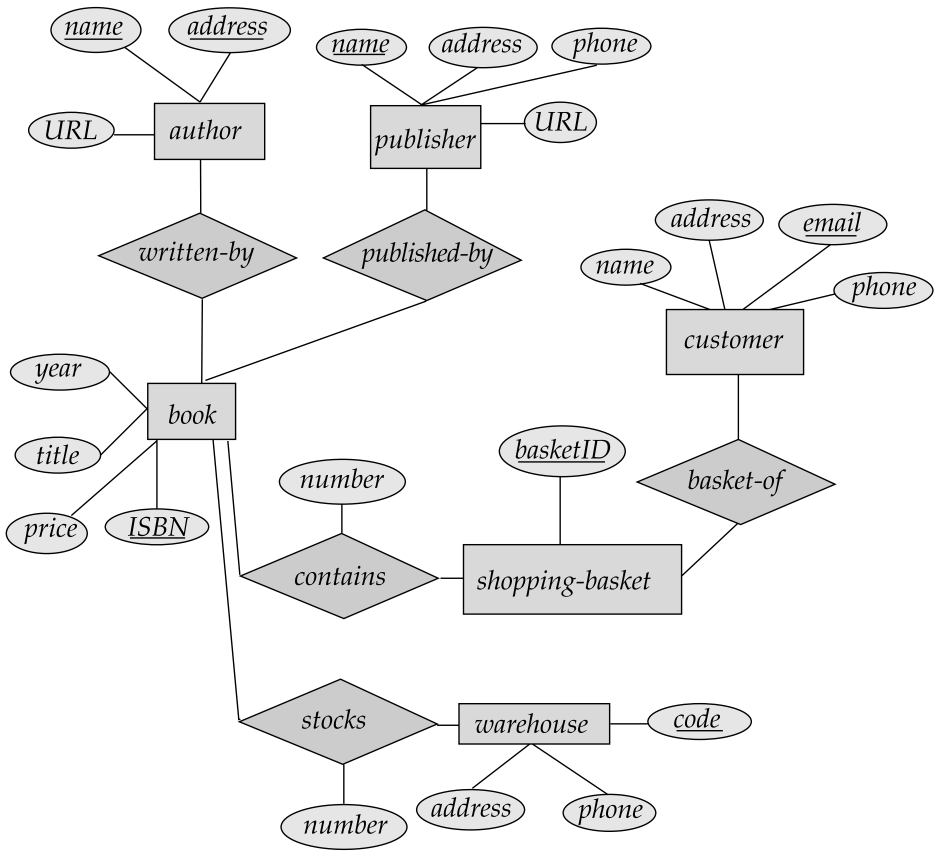

Figure 2.10 E-R diagram for Exercise 2.12.

2.12 Consider the E-R diagram in Figure 2.10, which models an online bookstore.

a. List the entity sets and their primary keys.

b. Suppose the bookstore adds music cassettes and compact disks to its collection. The same music item may be present in cassette or compact disk format, with differing prices. Extend the E-R diagram to model this addition, ignoring the effect on shopping baskets.

c. Now extend the E-R diagram, using generalization, to model the case where a shopping basket may contain any combination of books, music cassettes, or compact disks.

Answer:

2.13 Consider an E-R diagram in which the same entity set appears several times. Why is allowing this redundancy a bad practice that one should avoid whenever possible?

Answer: By using one entity set many times we are missing relationships in

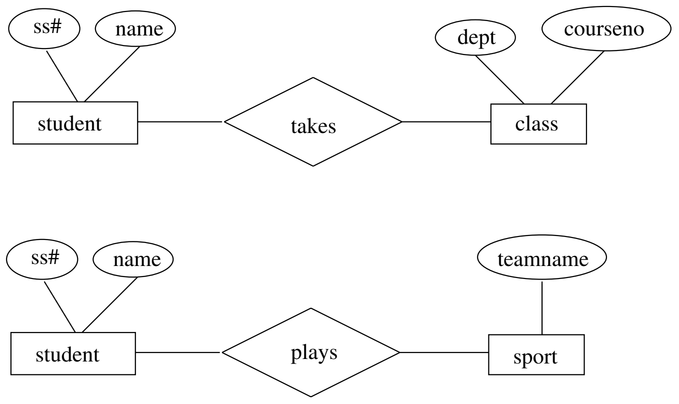

Figure 2.11 E-R diagram with entity duplication.

the model. For example, in the E-R diagram in Figure 2.11: the students taking classes are the same students who are athletes, but this model will not show that.

2.14 Consider a university database for the scheduling of classrooms for final exams. This database could be modeled as the single entity set *exam*, with attributes *course-name*, *section-number*, *room-number*, and *time*. Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the *exam* entity set, as

- *course* with attributes *name*, *department*, and *c-number*
- *section* with attributes *s-number* and *enrollment*, and dependent as a weak entity set on *course*
- *room* with attributes *r-number*, *capacity*, and *building*

a. Show an E-R diagram illustrating the use of all three additional entity sets listed.

b. Explain what application characteristics would influence a decision to include or not to include each of the additional entity sets.

Answer:

a. See Figure 2.12

b. The additional entity sets are useful if we wish to store their attributes as part of the database. For the *course* entity set, we have chosen to include three attributes. If only the primary key (*c-number*) were included, and if courses have only one section, then it would be appropriate to replace the *course* (and *section*) entity sets by an attribute (*c-number*) of *exam*. The reason it is undesirable to have multiple attributes of *course* as attributes of *exam* is that it would then be difficult to maintain data on the courses, particularly if a course has no exam or several exams. Similar remarks apply to the *room* entity set.

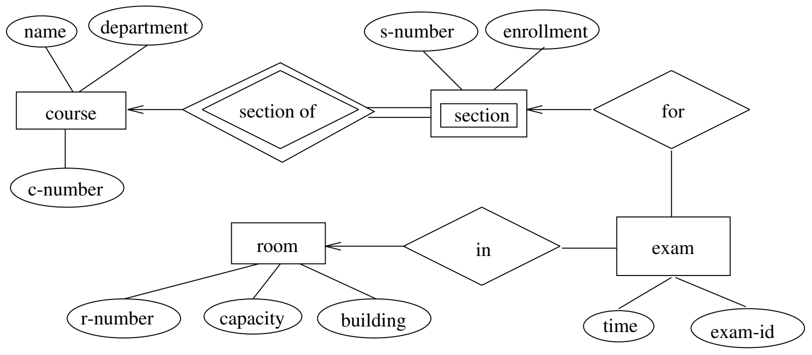

Figure 2.12 E-R diagram for exam scheduling.

2.15 When designing an E-R diagram for a particular enterprise, you have several alternatives from which to choose.

a. What criteria should you consider in making the appropriate choice?
b. Design three alternative E-R diagrams to represent the university registrar's office of Exercise 2.4. List the merits of each. Argue in favor of one of the alternatives.

Answer:

a. The criteria to use are intuitive design, accurate expression of the real-world concept and efficiency. A model which clearly outlines the objects and relationships in an intuitive manner is better than one which does not, because it is easier to use and easier to change. Deciding between an attribute and an entity set to represent an object, and deciding between an entity set and relationship set, influence the accuracy with which the real-world concept is expressed. If the right design choice is not made, inconsistency and/or loss of information will result. A model which can be implemented in an efficient manner is to be preferred for obvious reasons.
b. Consider three different alternatives for the problem in Exercise 2.4.
- See Figure 2.13
- See Figure 2.14
- See Figure 2.15

Each alternative has merits, depending on the intended use of the database. Scheme 2.13 has been seen earlier. Scheme 2.15 does not require a separate entity for *prerequisites*. However, it will be difficult to store all the prerequisites(being a multi-valued attribute). Scheme 2.14 treats prerequisites as well as classrooms as separate entities, making it useful for gathering data about prerequisites and room usage. Scheme 2.13 is in between the others, in that it treats prerequisites as separate entities but not classrooms. Since a registrar's office probably has to answer general questions about the number of classes a student is taking or what are all the prerequisites of a course, or where a specific class meets, scheme 2.14 is probably the best choice.

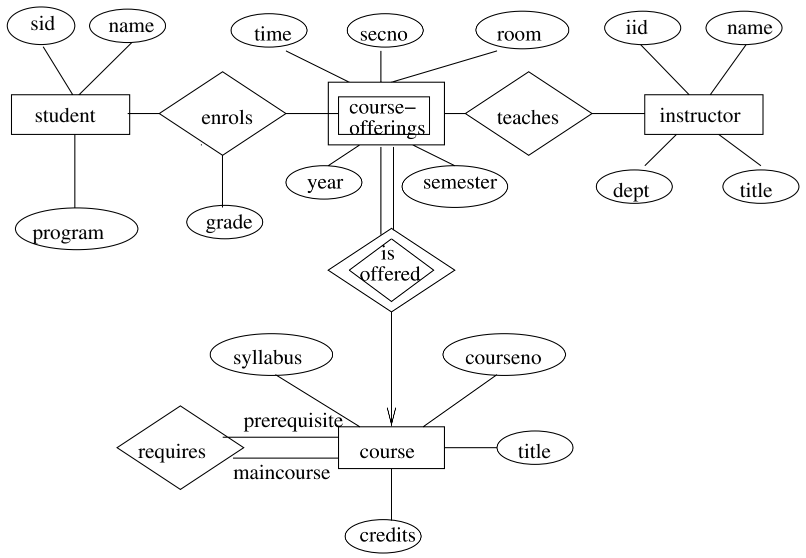

Figure 2.13 E-R diagram for University(a) .

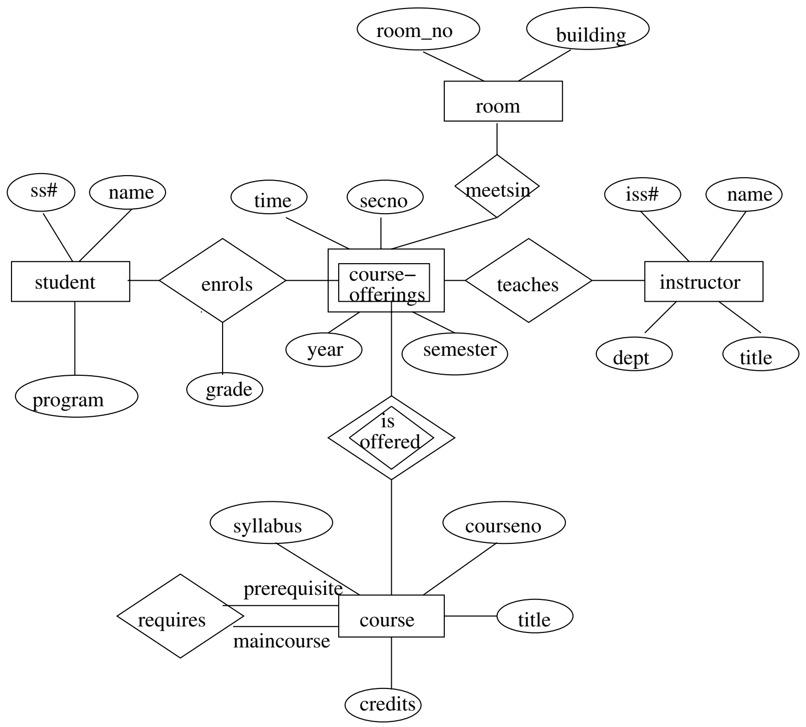

Figure 2.14 E-R diagram for University(b).

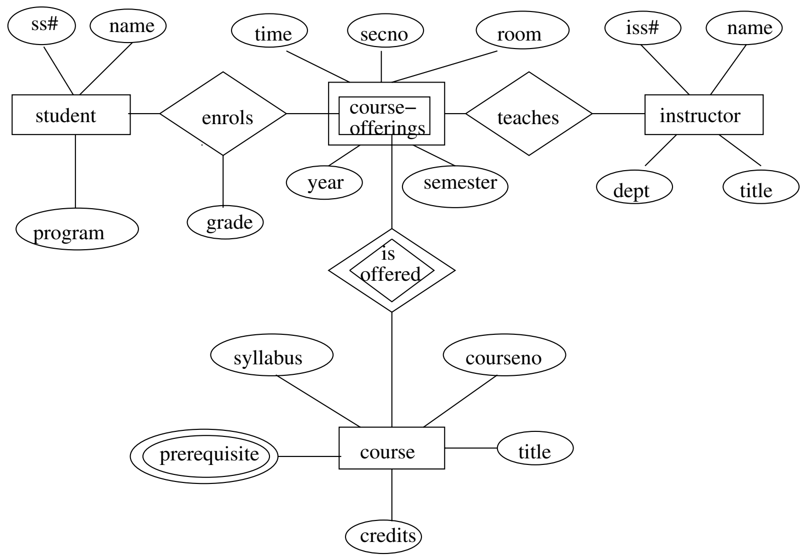

Figure 2.15 E-R diagram for University(c).

2.16 An E-R diagram can be viewed as a graph. What do the following mean in terms of the structure of an enterprise schema?

a. The graph is disconnected.
b. The graph is acyclic.

Answer:

a. If a pair of entity sets are connected by a path in an E-R diagram, the entity sets are related, though perhaps indirectly. A disconnected graph implies that there are pairs of entity sets that are unrelated to each other. If we split the graph into connected components, we have, in effect, a separate database corresponding to each connected component.
b. As indicated in the answer to the previous part, a path in the graph between a pair of entity sets indicates a (possibly indirect) relationship between the two entity sets. If there is a cycle in the graph then every pair of entity sets on the cycle are related to each other in at least two distinct ways. If the E-R diagram is acyclic then there is a unique path between every pair of entity sets and, thus, a unique relationship between every pair of entity sets.

2.17 In Section 2.4.3, we represented a ternary relationship (Figure 2.16a) using binary relationships, as shown in Figure 2.16b. Consider the alternative shown in Figure 2.16c. Discuss the relative merits of these two alternative representations of a ternary relationship by binary relationships.
Answer: The model of Figure 2.16c will not be able to represent all ternary relationships. Consider the *ABC* relationship set below.

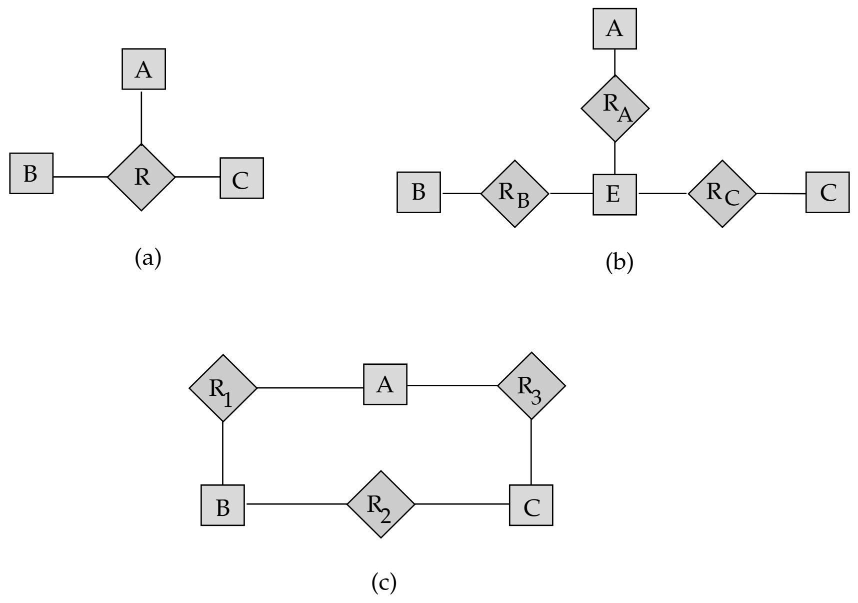

Figure 2.16 E-R diagram for Exercise 2.17 (attributes not shown.)

A	*B*	C
1	2	3
4	2	7
4	8	3

If *ABC* is broken into three relationships sets *AB*, *BC* and *AC*, the three will imply that the relation (4, 2, 3) is a part of *ABC*.

2.18 Consider the representation of a ternary relationship using binary relationships as described in Section 2.4.3 (shown in Figure 2.16b.)

a. Show a simple instance of E, A, B, C, R_A, R_B, and R_C that cannot correspond to any instance of A, B, C, and R.

b. Modify the E-R diagram of Figure 2.16b to introduce constraints that will guarantee that any instance of E, A, B, C, R_A, R_B, and R_C that satisfies the constraints will correspond to an instance of A, B, C, and R.

c. Modify the translation above to handle total participation constraints on the ternary relationship.

d. The above representation requires that we create a primary key attribute for E. Show how to treat E as a weak entity set so that a primary key attribute is not required.

Answer:

a. Let $E = \{e_1, e_2\}$, $A = \{a_1, a_2\}$, $B = \{b_1\}$, $C = \{c_1\}$, $R_A = \{(e_1, a_1), (e_2, a_2)\}$, $R_B = \{(e_1, b_1)\}$, and $R_C = \{(e_1, c_1)\}$. We see that because of the tuple (e_2, a_2), no instance of R exists which corresponds to E, R_A, R_B and R_C.

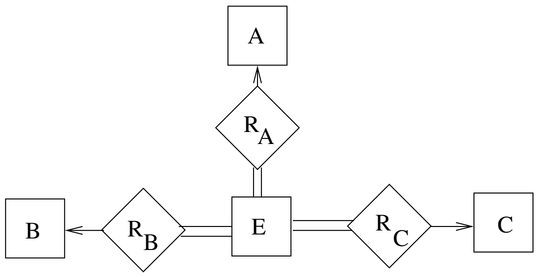

Figure 2.17 E-R diagram to Exercise 2.17b.

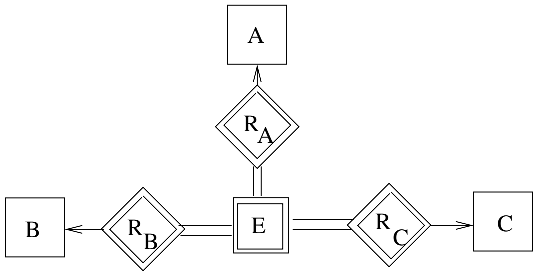

Figure 2.18 E-R diagram to Exercise 2.17d.

b. See Figure 2.17. The idea is to introduce total participation constraints between E and the relationships R_A, R_B, R_C so that every tuple in E has a relationship with A, B and C.

c. Suppose A totally participates in the relationhip R, then introduce a total participation constraint between A and R_A.

d. Consider E as a weak entity set and R_A, R_B and R_C as its identifying relationship sets. See Figure 2.18.

2.19 A weak entity set can always be made into a strong entity set by adding to its attributes the primary key attributes of its identifying entity set. Outline what sort of redundancy will result if we do so.
Answer: The primary key of a weak entity set can be inferred from its relationship with the strong entity set. If we add primary key attributes to the weak entity set, they will be present in both the entity set and the relationship set and they have to be the same. Hence there will be redundancy.

2.20 Design a generalization–specialization hierarchy for a motor-vehicle sales company. The company sells motorcycles, passenger cars, vans, and buses. Justify your placement of attributes at each level of the hierarchy. Explain why they should not be placed at a higher or lower level.
Answer: Figure 2.19 gives one possible hierarchy, there could be many different solutions. The generalization–specialization hierarchy for the motor-vehicle company is given in the figure. *model*, *sales-tax-rate* and *sales-volume* are attributes necessary for all types of vehicles. Commercial vehicles attract commercial vehi-

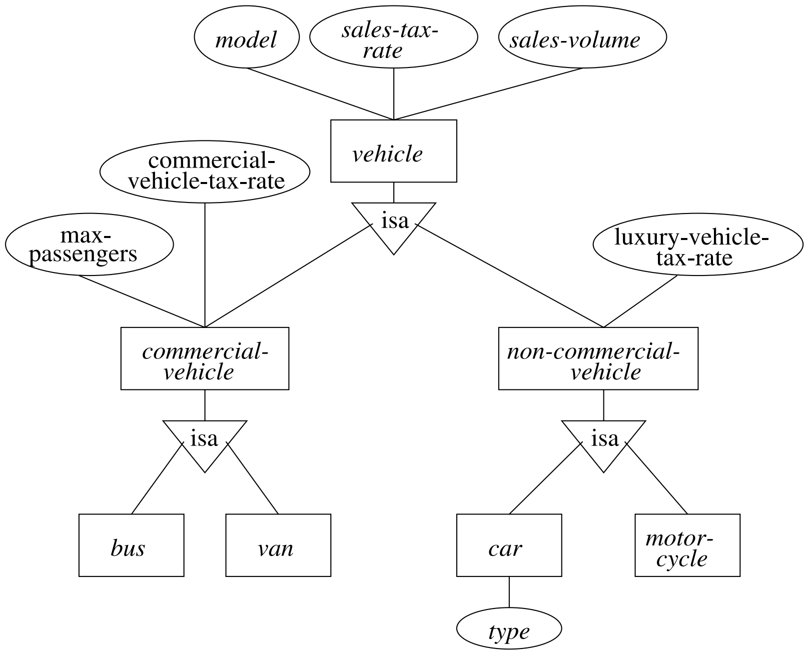

Figure 2.19 E-R diagram of motor-vehicle sales company.

cle tax, and each kind of commercial vehicle has a passenger carrying capacity specified for it. Some kinds of non-commercial vehicles attract luxury vehicle tax. Cars alone can be of several types, such as sports-car, sedan, wagon etc., hence the attribute *type*.

2.21 Explain the distinction between condition-defined and user-defined constraints. Which of these constraints can the system check automatically? Explain your answer.
Answer: In a generalization–specialization hierarchy, it must be possible to decide which entities are members of which lower level entity sets. In a condition-defined design constraint, membership in the lower level entity-sets is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate.User-defined lower-level entity sets are not constrained by a membership condition; rather, entities are assigned to a given entity set by the database user.

Condition-defined constraints alone can be automatically handled by the system. Whenever any tuple is inserted into the database, its membership in the various lower level entity-sets can be automatically decided by evaluating the respective membership predicates. Similarly when a tuple is updated, its membership in the various entity sets can be re-evaluated automatically.

2.22 Explain the distinction between disjoint and overlapping constraints.
Answer: In a disjointness design constraint, an entity can belong to not more

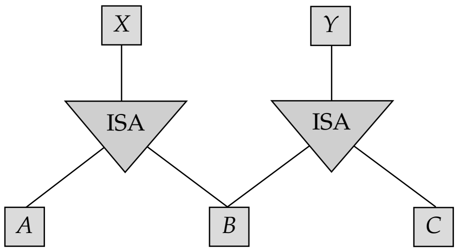

Figure 2.20 E-R diagram for Exercise 2.24 (attributes not shown).

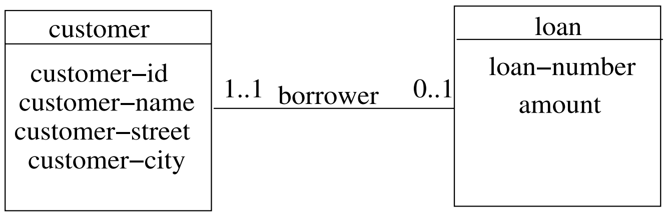

Figure 2.21 UML equivalent of Figure 2.9c.

than one lower-level entity set. In overlapping generalizations, the same entity may belong to more than one lower-level entity sets. For example, in the employee-workteam example of the book, a manager may participate in more than one work-team.

2.23 Explain the distinction between total and partial constraints.
Answer: In a total design constraint, each higher-level entity must belong to a lower-level entity set. The same need not be true in a partial design constraint. For instance, some employees may belong to no work-team.

2.24 Figure 2.20 shows a lattice structure of generalization and specialization. For entity sets A, B, and C, explain how attributes are inherited from the higher-level entity sets X and Y. Discuss how to handle a case where an attribute of X has the same name as some attribute of Y.
Answer: *A* inherits all the attributes of *X* plus it may define its own attributes. Similarly C inherits all the attributes of *Y* plus its own attributes. *B* inherits the attributes of both *X* and *Y*. If there is some attribute *name* which belongs to both *X* and *Y*, it may be referred to in *B* by the qualified name *X.name* or *Y.name*.

2.25 Draw the UML equivalents of the E-R diagrams of Figures 2.9c, 2.10, 2.12, 2.13 and 2.17.
Answer: See Figures 2.21 to 2.25

2.26 Consider two separate banks that decide to merge. Assume that both banks use exactly the same E-R database schema—the one in Figure 2.22. (This assumption is, of course, highly unrealistic; we consider the more realistic case in

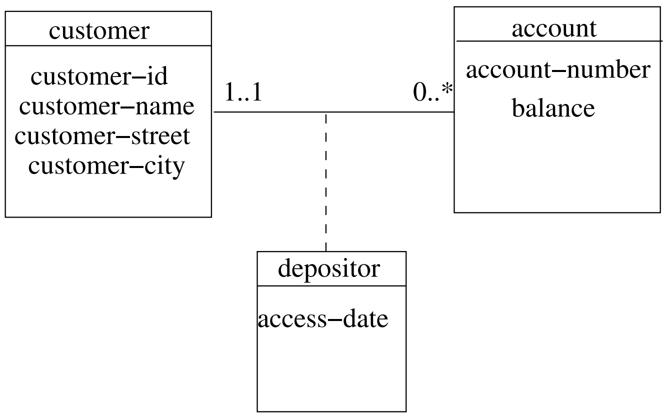

Figure 2.22 UML equivalent of Figure 2.10

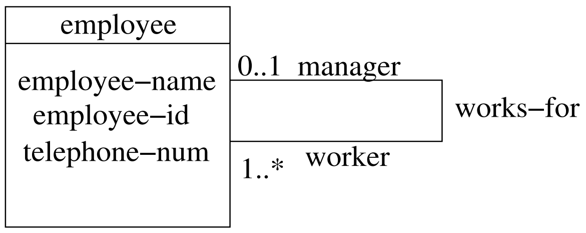

Figure 2.23 UML equivalent of Figure 2.12

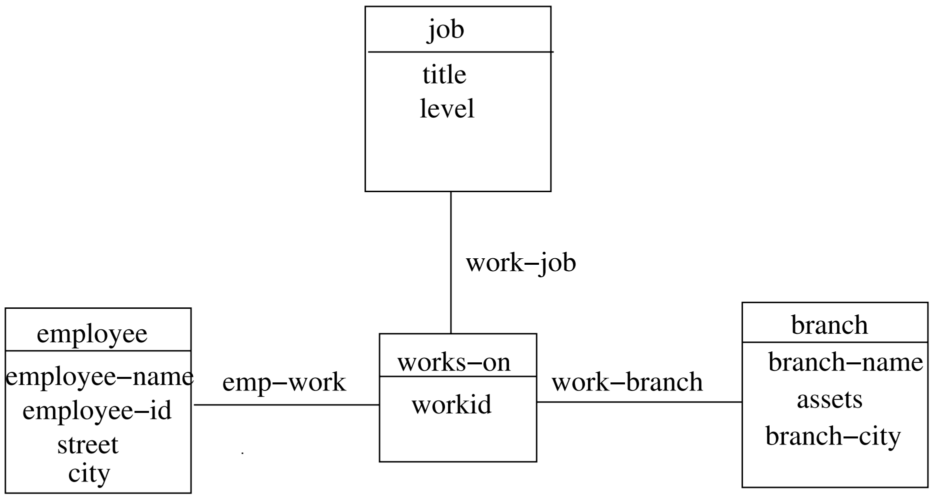

Figure 2.24 UML equivalent of Figure 2.13

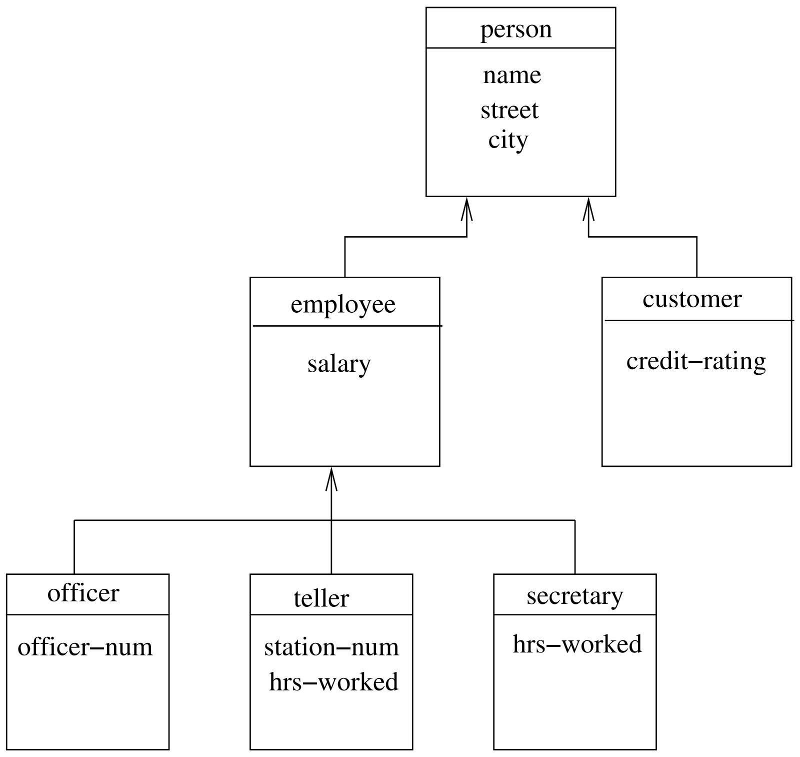

Figure 2.25 UML equivalent of Figure 2.17

Section 19.8.) If the merged bank is to have a single database, there are several potential problems:

- The possibility that the two original banks have branches with the same name
- The possibility that some customers are customers of both original banks
- The possibility that some loan or account numbers were used at both original banks (for different loans or accounts, of course)

For each of these potential problems, describe why there is indeed a potential for difficulties. Propose a solution to the problem. For your solution, explain any changes that would have to be made and describe what their effect would be on the schema and the data.

Answer: In this example, we assume that both banks have the shared identifiers for customers, such as the social security number. We see the general solution in the next exercise.

Each of the problems mentioned does have potential for difficulties.

a. *branch-name* is the primary-key of the *branch* entity set. Therefore while merging the two banks' entity sets, if both banks have a branch with the same name, one of them will be lost.

b. customers participate in the relationship sets *cust-banker*, *borrower* and *depositor*. While merging the two banks' *customer* entity sets, duplicate tuples of the same customer will be deleted. Therefore those relations in the three mentioned relationship sets which involved these deleted tuples will have to be updated. Note that if the tabular representation of a relationship set is obtained by taking a union of the primary keys of the participating entity sets, no modification to these relationship sets is required.

c. The problem caused by *loans* or *accounts* with the same number in both the banks is similar to the problem caused by branches in both the banks with the same *branch-name*.

To solve the problems caused by the merger, no schema changes are required. Merge the *customer* entity sets removing duplicate tuples with the same *social-security* field. Before merging the *branch* entity sets, prepend the old bank name to the *branch-name* attribute in each tuple. The *employee* entity sets can be merged directly, and so can the *payment* entity sets. No duplicate removal should be performed. Before merging the *loan* and *account* entity sets, whenever there is a number common in both the banks, the old number is replaced by a new unique number, in one of the banks.

Next the relationship sets can be merged. Any relation in any relationship set which involves a tuple which has been modified earlier due to the merger, is itself modified to retain the same meaning. For example let 1611 be a loan number common in both the banks prior to the merger, and let it be replaced by a new unique number 2611 in one of the banks, say bank 2. Now all the relations in *borrower*, *loan-branch* and *loan-payment* of bank 2 which refer to loan number 1611 will have to be modified to refer to 2611. Then the merger with bank 1's corresponding relationship sets can take place.

2.27 Reconsider the situation described for Exercise 2.26 under the assumption that one bank is in the United States and the other is in Canada. As before, the banks use the schema of Figure 2.22, except that the Canadian bank uses the *social-insurance* number assigned by the Canadian government, whereas the U.S. bank uses the social-security number to identify customers. What problems (beyond those identified in Exercise 2.24) might occur in this multinational case? How would you resolve them? Be sure to consider both the scheme and the actual data values in constructing your answer.

Answer: This is a case in which the schemas of the two banks differ, so the merger becomes more difficult. The identifying attribute for persons in the US is *social-security*, and in Canada it is *social-insurance*. Therefore the merged schema cannot use either of these. Instead we introduce a new attribute *person-id*, and use this uniformly for everybody in the merged schema. No other change to the schema is required. The values for the *person-id* attribute may be obtained by several ways. One way would be to prepend a country code to the old *social-security* or *social-insurance* values ("U" and "C" respectively, for instance), to get the corresponding *person-id* values. Another way would be to assign fresh numbers starting from 1 upwards, one number to each *social-security* and *social-insurance* value in the old databases.

Once this has been done, the actual merger can proceed as according to the answer to the previous question. If a particular relationship set, say *borrower*, involves only US customers, this can be expressed in the merged database by specializing the entity-set *customer* into *us-customer* and *canada-customer*, and making only *us-customer* participate in the merged *borrower*. Similarly *employee* can be specialized if needed.