

CHAPTER 6



Database Design using the E-R Model

Practice Exercises

- 6.1 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. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.
- 6.2 Consider a database that includes the entity sets *student*, *course*, and *section* from the university schema and that additionally records the marks that students receive in different exams of different sections.
 - a. Construct an E-R diagram that models exams as entities and uses a ternary relationship as part of the design.
 - b. Construct an alternative E-R diagram that uses only a binary relationship between *student* and *section*. Make sure that only one relationship exists between a particular *student* and *section* pair, yet you can represent the marks that a student gets in different exams.
- 6.3 Design an E-R diagram for keeping track of the scoring statistics of your favorite sports team. You should store the matches played, the scores in each match, the players in each match, and individual player scoring statistics for each match. Summary statistics should be modeled as derived attributes with an explanation as to how they are computed.
- 6.4 Consider an E-R diagram in which the same entity set appears several times, with its attributes repeated in more than one occurrence. Why is allowing this redundancy a bad practice that one should avoid?

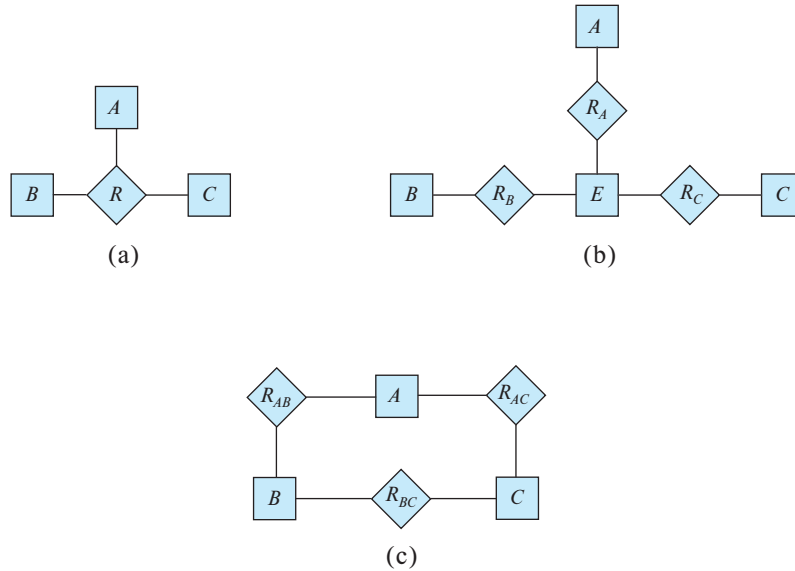
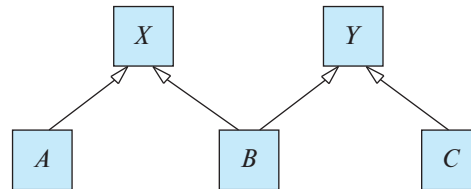


Figure 6.29 Representation of a ternary relationship using binary relationships.

- 6.5 An E-R diagram can be viewed as a graph. What do the following mean in terms of the structure of an enterprise schema?
- The graph is disconnected.
 - The graph has a cycle.
- 6.6 Consider the representation of the ternary relationship of Figure 6.29a using the binary relationships illustrated in Figure 6.29b (attributes not shown).
- 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 .
 - Modify the E-R diagram of Figure 6.29b 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 .
 - Modify the preceding translation to handle total participation constraints on the ternary relationship.
- 6.7 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.
- 6.8 Consider a relation such as *sec_course*, generated from a many-to-one relationship set *sec_course*. Do the primary and foreign key constraints created on the relation enforce the many-to-one cardinality constraint? Explain why.

- 6.9 Suppose the *advisor* relationship set were one-to-one. What extra constraints are required on the relation *advisor* to ensure that the one-to-one cardinality constraint is enforced?
- 6.10 Consider a many-to-one relationship R between entity sets A and B . Suppose the relation created from R is combined with the relation created from A . In SQL, attributes participating in a foreign key constraint can be null. Explain how a constraint on total participation of A in R can be enforced using **not null** constraints in SQL.
- 6.11 In SQL, foreign key constraints can reference only the primary key attributes of the referenced relation or other attributes declared to be a superkey using the **unique** constraint. As a result, total participation constraints on a many-to-many relationship set (or on the “one” side of a one-to-many relationship set) cannot be enforced on the relations created from the relationship set, using primary key, foreign key, and not null constraints on the relations.
- Explain why.
 - Explain how to enforce total participation constraints using complex check constraints or assertions (see Section 4.4.8). (Unfortunately, these features are not supported on any widely used database currently.)
- 6.12 Consider the following lattice structure of generalization and specialization (attributes not shown).



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 .

- 6.13 An E-R diagram usually models the state of an enterprise at a point in time. Suppose we wish to track *temporal changes*, that is, changes to data over time. For example, Zhang may have been a student between September 2015 and May 2019, while Shankar may have had instructor Einstein as advisor from May 2018 to December 2018, and again from June 2019 to January 2020. Similarly, attribute values of an entity or relationship, such as *title* and *credits* of *course*, *salary*, or even *name* of *instructor*, and *tot_cred* of *student*, can change over time.
- One way to model temporal changes is as follows: We define a new data type called **valid_time**, which is a time interval, or a set of time intervals. We then

associate a *valid_time* attribute with each entity and relationship, recording the time periods during which the entity or relationship is valid. The end time of an interval can be infinity; for example, if Shankar became a student in September 2018, and is still a student, we can represent the end time of the *valid_time* interval as infinity for the Shankar entity. Similarly, we model attributes that can change over time as a set of values, each with its own *valid_time*.

- a. Draw an E-R diagram with the *student* and *instructor* entities, and the *advisor* relationship, with the above extensions to track temporal changes.
- b. Convert the E-R diagram discussed above into a set of relations.

It should be clear that the set of relations generated is rather complex, leading to difficulties in tasks such as writing queries in SQL. An alternative approach, which is used more widely, is to ignore temporal changes when designing the E-R model (in particular, temporal changes to attribute values), and to modify the relations generated from the E-R model to track temporal changes.