EXERCISE 5-1

The class in Figure A-18 records information about a department. What other options are there for modeling information about the manager and location of a department?

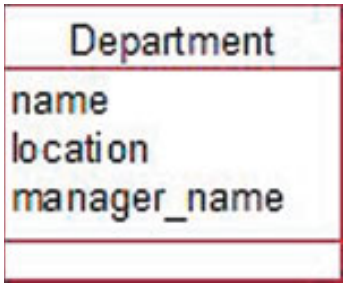


Figure A-18. *Initial attempt at modeling the information about a department*

Let's first think about location. What might we want to do with this information? It may be useful to be able to find all the departments that are in the same location—to let them know the CEO is visiting, say. If we want to retrieve an object based on the value of an attribute, then we must ensure that the value is stored consistently in each object. Creating a new class can help with that. If we introduce a `Location` class, then we can store information about each location and set up a relationship between departments and location. Introducing a `Location` class also allows us to keep additional information about the location: address, phone, and so on.

It is unlikely that we will regularly want to retrieve department objects based on the manager's name, as for the most part managers will only be attached to a single department. However, there is a great deal of additional information we need to know about a manager. How to contact him would be a start. Do we already have this information? The company will surely have information stored about all its employees, so here we should model manager as a relationship to an existing `Employee` class.

A better model is shown in Figure A-19. If you are keen, try developing this new model to account for previous managers.

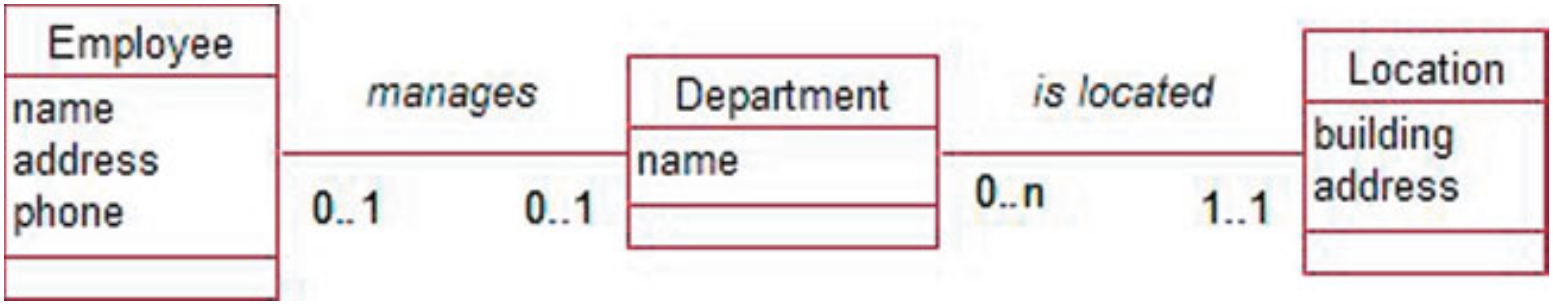


Figure A-19. *More flexible model for department information*

EXERCISE 5-2

A university wants to model information about the teaching of courses. A number of staff members may contribute to teaching a course and one staff member is denoted as the course supervisor. Suggest an initial data model.

We have two obvious classes for this situation: `Course` and `StaffMember`. There are two different relationships between the classes: `teaches` and `supervises`. A possible model is shown in Figure A-20. Reading from right to left, each course has one or more teachers and each course has a single supervisor. From left to right, staff members might teach and/or supervise any number of courses. In this model we haven't considered any historical information; nor have we considered other possible constraints, such as, does the supervisor need to be one of the teachers?

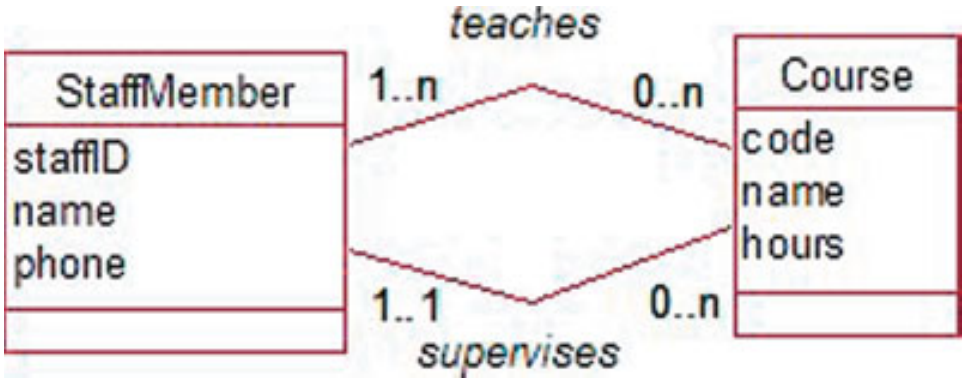


Figure A-20. *Modeling the teaching and supervision of courses with two relationships*

EXERCISE 5-3

How would you model the information about marriages? Think about all the different situations that could eventuate (for simplicity, do not worry at this stage about the gender of the participants).

We need to keep information about people—their names, at least. One person marries another person, so a self relationship is required as in Figure A-21.

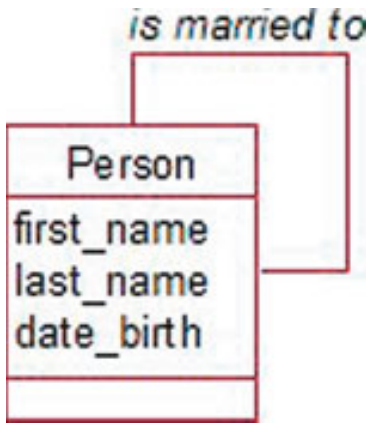


Figure A-21. *Modeling marriage with a self relationship*

Optionalities are easy—people are not required to be married to someone else. What about the cardinalities Generally it is one marriage at a time (but not always). However, historically a person may have many marriages. This would require a Many–Many relationship. When we have a Many–Many relationship it is always prudent to ask, "What additional information could be useful about a particular pair of objects?" In this case the dates of the marriage would be useful. These can be kept in an intermediate class, `Marriage`. We could also keep details on the cause of the end of the marriage (divorce, separation, death of spouse, etc.). A class, `EndType`, to maintain these categories accurately would be useful, and something like the data model in Figure A-22 would capture much of this detail. Each marriage involves exactly two people (the two lines between `Marriage` and `Person`) and each person can be involved in any number of marriages. Some marriages have ended and so have an optional category associated with them.

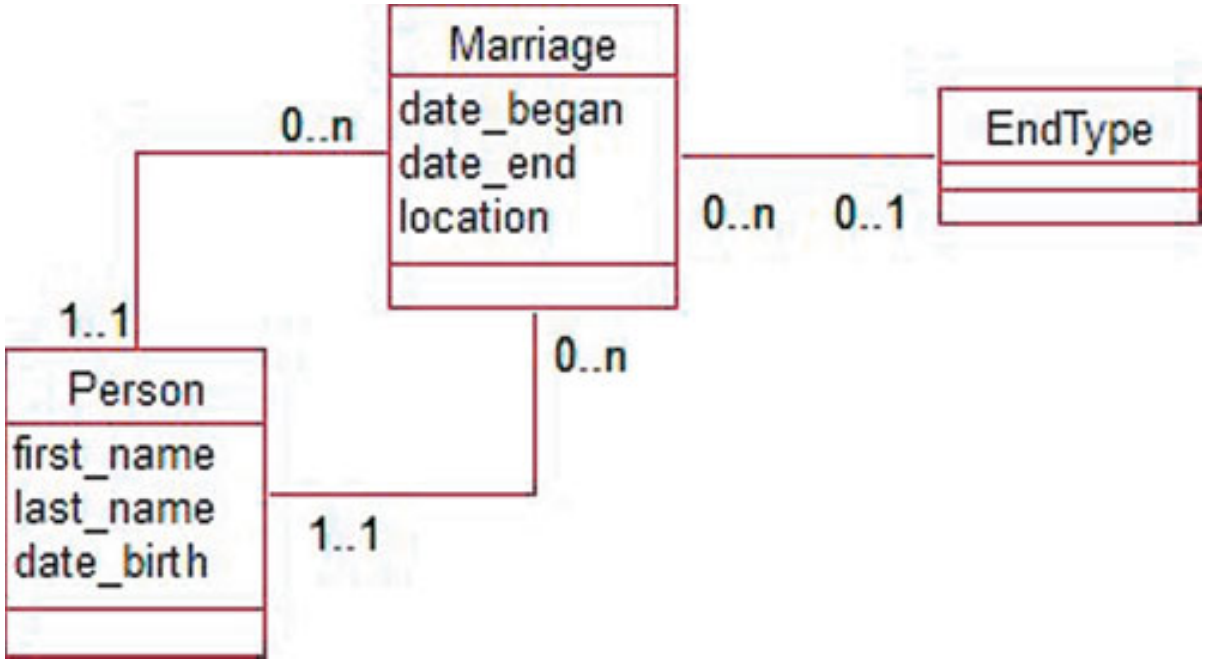


Figure A-22. *Including additional classes to capture marriage information*

EXERCISE 5-4

An orchestra keeps information about its musicians, repertoire, and concerts. A partial data model is shown in Figure A-23.

What false information could be deduced from this initial model?

Amend the model so that it can maintain the following information correctly:

- Which players are involved in particular works in a concert

- The works being presented at a concert
- The fee a player receives for appearing in a particular concert

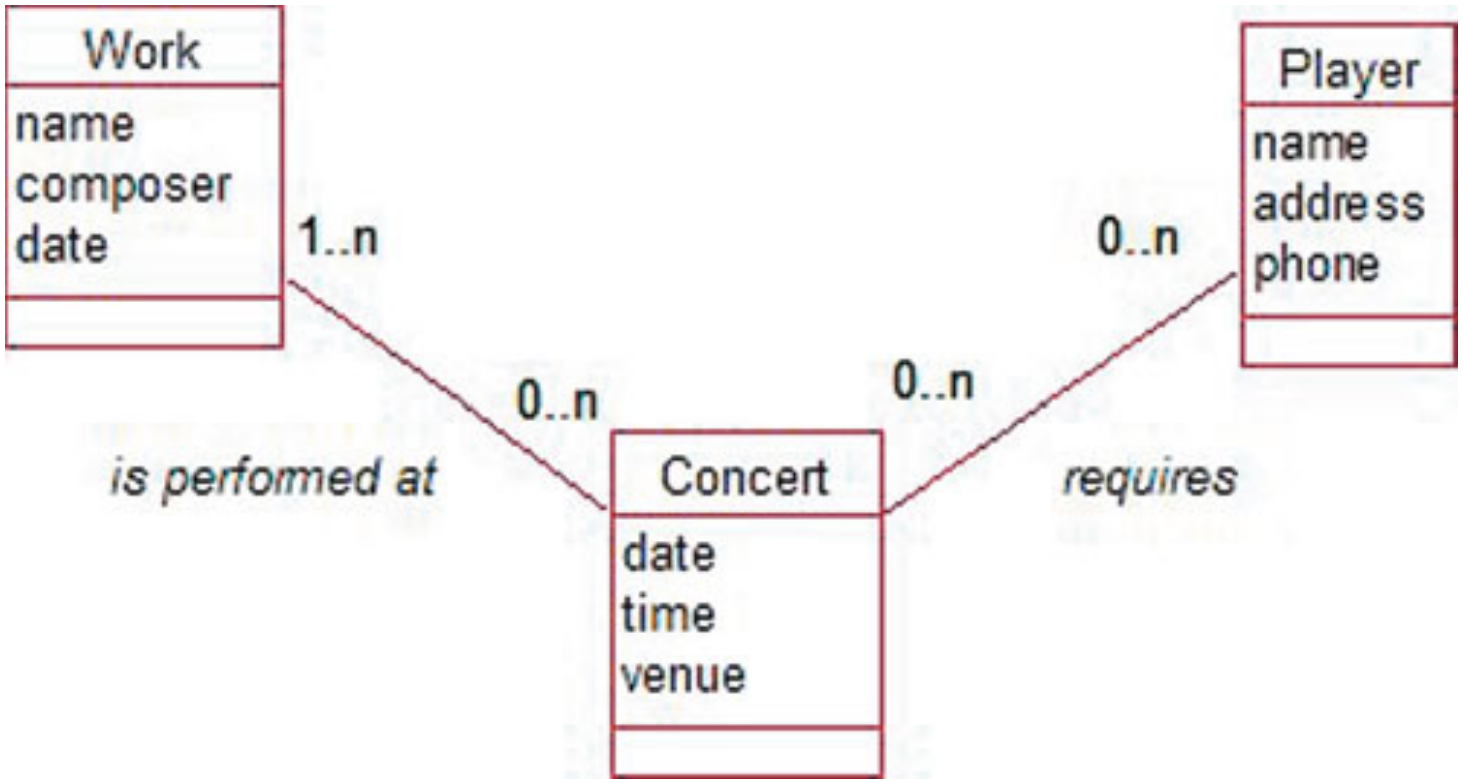
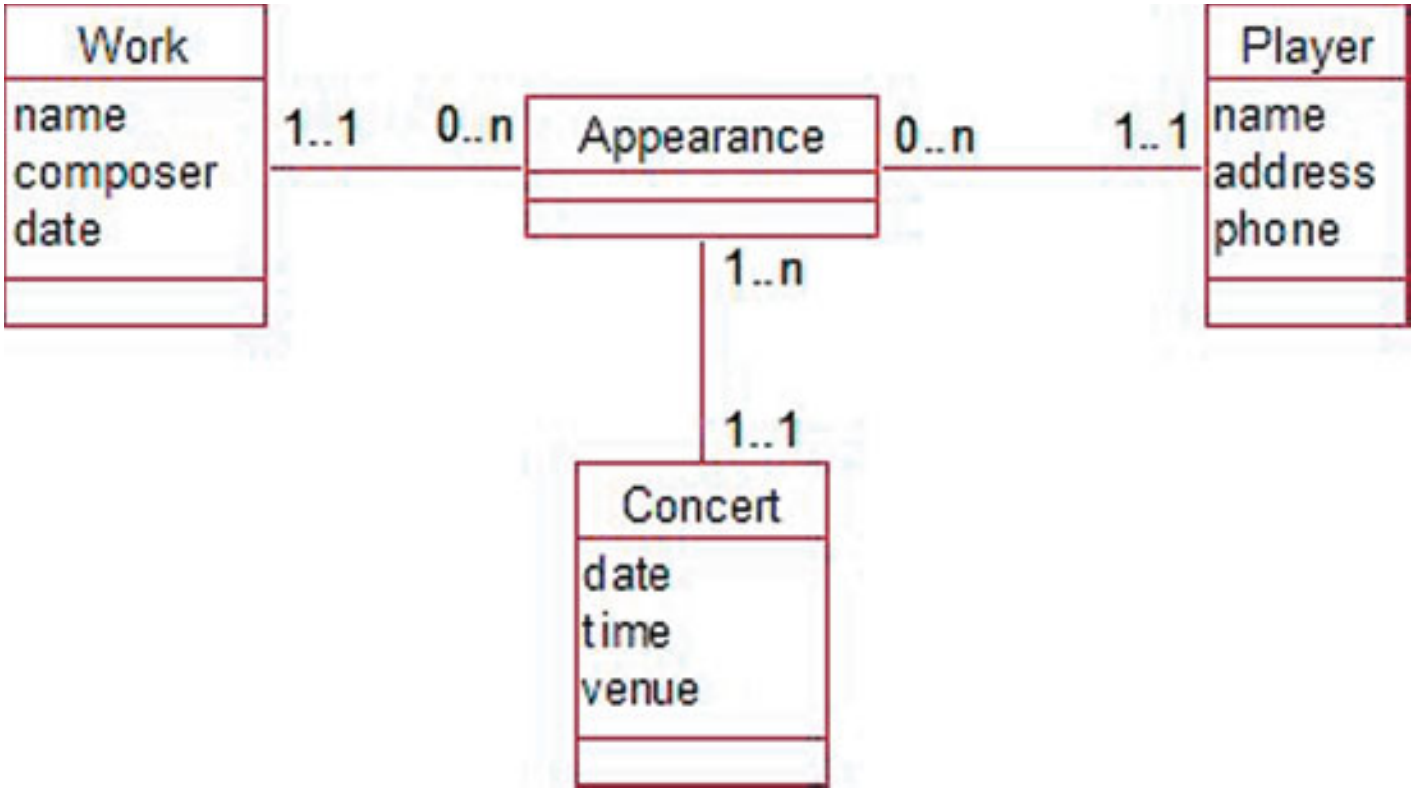


Figure A-23. *A partial model for an orchestra's concert information*

This model contains a fan trap: there are Many cardinalities at the outside ends of the two relationships. For example, if we have information that Joe Smith is required for Saturday's concert and that Beethoven's violin sonata is to be performed at Saturday's concert, we could incorrectly make the deduction that Joe performs in the violin sonata.

If we want to know which players are involved in particular works in a concert we need a ternary relationship that involves the three classes, `Player`, `Concert`, and `Work`, simultaneously Figure A-24 shows the ternary relationship represented by a new class, `Appearance`.



A-24. *A ternary relationship represented by an additional class*

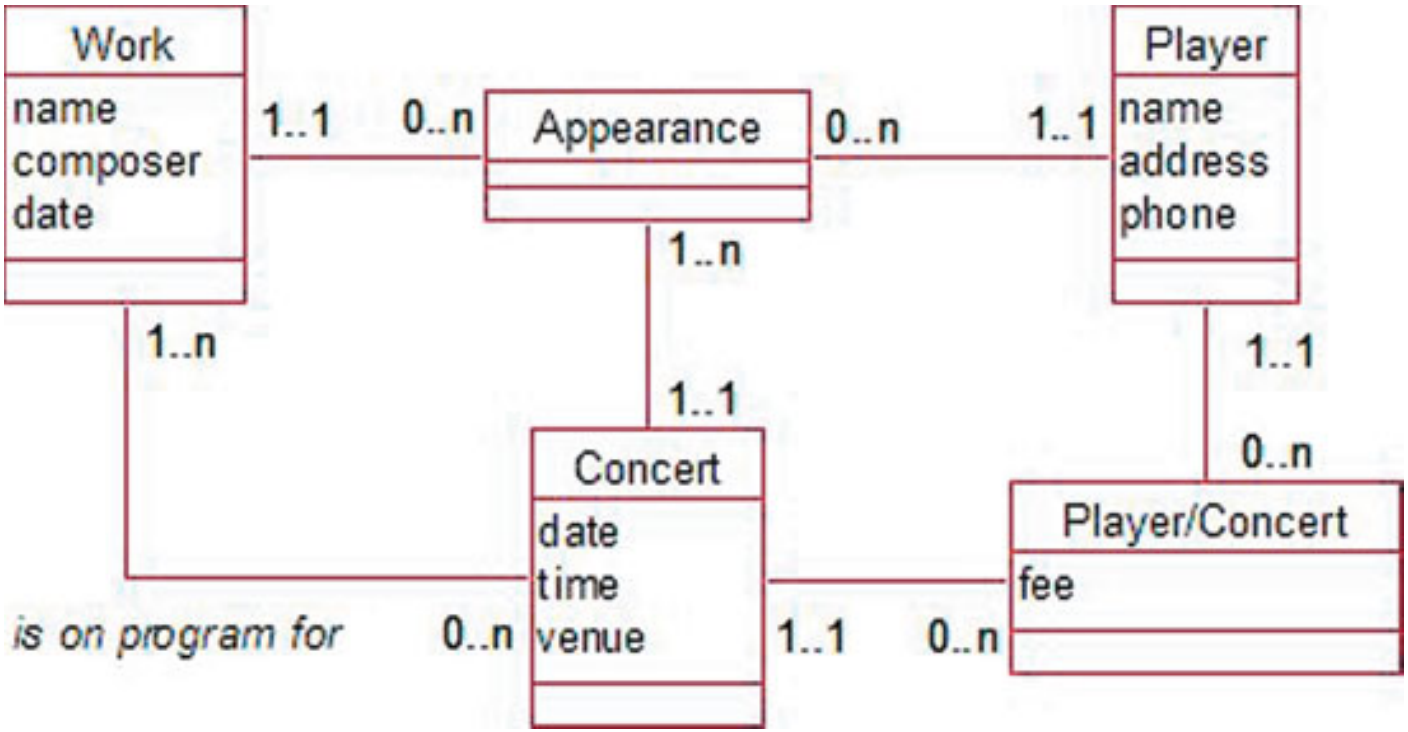
Each object of the `Appearance` class is related to one player, one work, and one concert. It allows us to keep data as in the table in Figure A-25. We can see that Joe played in the Sonata and the Symphony on Saturday, while Linda played only in the Sonata.

player	work	concert
Joe	Sonata	Saturday
Linda	Sonata	Saturday
Joe	Symphony	Saturday
Linda	Sonata	Monday
Mary	Duet	Monday

A-25. *Example data represented by the class Appearance*

Do we need a relationship between `Work` and `Concert` so that we can know what is on the program? From Figure A-25 we can see that Saturday's concert features the Sonata and the Symphony. However, the program will probably be decided long before any players are involved, so we need the information about works in a particular concert, independent of the players. The binary relationship between `Concert` and `Work` in Figure A-26 can record that information; a concert has one or more works on the program and a work might be performed at many concerts.

What about the fee a player is paid for performing in a concert? If the fees go with each work, then they could be included as attributes in `Appearance`. Joe is paid $30 for playing the Sonata on Saturday and $50 for playing the Symphony, for example. If a fee is for the entire concert (e.g., a travel allowance), then it is independent of the works and a new binary relationship between `Player` and `Concert` will be required. The relationship will be Many–Many (a player may be involved in many concerts and a concert will have many players). The new intermediate class `Player/Concert` is required so that we have somewhere to include the fee for particular pairings of player and concert.



A-26. *A more comprehensive model for concert information*