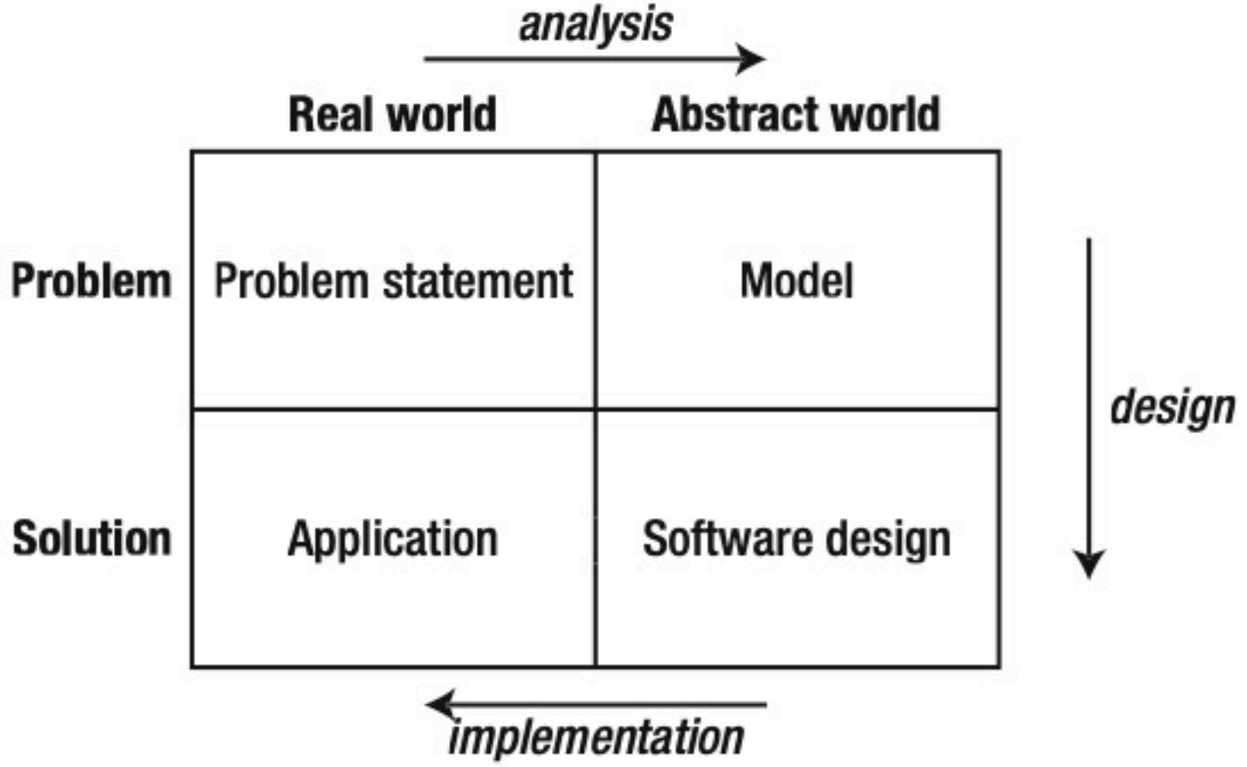


Figure 2-1. *The software process (based on Zelkowitz et al., 1979[1])*

Using Figure 2-1 as a way of thinking about software processes, we will now look at how the various steps relate to setting up a database project by applying those steps to Example 1-1, "The Plant Database."

Initial Problem Statement

We start with some initial description of the problem. One way to represent a description is with *use cases*, which are part of the *Unified Modeling Language* (UML),[2] a set of diagramming techniques used to depict various aspects of the software process. Use cases are descriptions of how different types of users (more formally known as *actors*) might interact with the system. Most texts on systems analysis include discussions about use cases. (Alistair Cockburn's book *Writing Effective Use Cases*[3] is a particularly readable and pragmatic account.) Use cases can be at many different levels, from high–level corporate goals down to descriptions of small program modules. We will concentrate on the tasks someone sitting in front of a desktop computer would be trying to carry out. For a database project, these tasks are most likely to be entering or updating data, and extracting information based on that data.

The UML notation for use cases involves stick figures representing, in our case, types of users, and ovals representing each of the tasks that the user needs to be able to carry out. For example, Figure 2-2 illustrates a use case in which a user performs three as yet unknown tasks. However, those stick figures and ovals aren't really enough to describe a given interaction with a system. When writing a use case, along with a diagram you should create a text document describing in more detail what the use case entails.

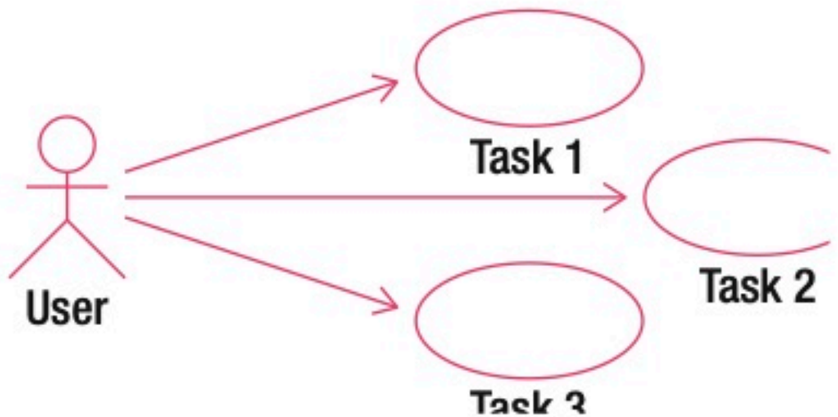


Figure 2-2. *UML notation for use cases*[4]

Let's see how use cases can be applied to the problem from Example 1-1 in the last chapter. Figure 2-3 recaps where we started with an initial database table recording plants and their uses.

plantID	genus	species	common_name	use1	use2	use3
1	Dodonaea	viscosa	Akeake	shelter	hedging	soil stability
2	Cedrus	atlantica	Atlas cedar	shelter		
3	Alnus	glutinosa	Black alder	soil stability	shelter	firewood
4	Eucalyptus	nichollii	Black peppermint gum	shelter	coppicing	bird food
5	Juglans	nigra	Black walnut	timber		
6	Acacia	mearnsii	Black wattle	firewood	shelter	soil stability

Figure 2-3. *Original data of plants and uses*

If we consider what typical people might want to do with the data shown in Figure 2-3, the use cases suggested in Example 2-1 would be a start.

EXAMPLE 2-1. INITIAL USE CASES FOR THE PLANT DATABASE

Figure 2-4 shows some initial use cases for the plant database. The text following the figure describes each use case.

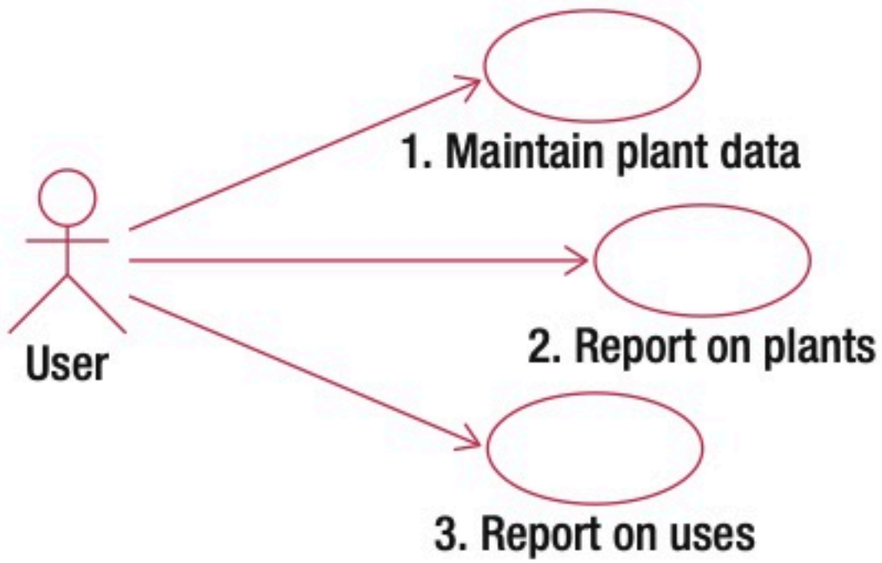


Figure 2-4. *First attempt at use cases for the plant database*

Use case 1: Enter (or edit) all the data we have about each plant; that is, plant ID, genus, species, common name, and uses.

Use case 2: Find or report information about a plant (or every plant) and see what it is useful for.

Use case 3: Specify a use and find the appropriate plants (or report for all uses).

As explained in the previous chapter, if the data is stored as in Figure 2-3, we cannot conveniently satisfy the requirements of all the use cases in Example 2-1. It is easy to get information about each plant (use case 2) by looking at each row in the table. However, finding all the plants that satisfy a particular use is extremely awkward. Have a go at finding all the plants suitable for firewood. You have to look in each of the use columns for every row.

Classes and Objects

Each *class* can be considered a template for storing data about a set of similar things (places, events, or people). Let's consider Example 2-1 about plants and their uses. An obvious candidate for our first class is the idea of a **`Plant`**. Each plant can be described in a similar way in that each has a **`genus`**, a **`species`**, a **`common_name`**, and perhaps a **`plantID`** number. These pieces of information, that we will keep about each plant, are referred to as the *attributes* (or *properties*) of the class. Figure 2-5 shows the UML notation for a class and its attributes. The name of the class appears in the top panel, and the middle panel contains the attributes. For some types of software systems, there may be processes that a class would be responsible for carrying out. For example, an **`Order`** class related to an online shopping cart might have a process for calculating a price including tax. These are known as *methods* and appear in the bottom panel. For predominantly information–based problems, methods are not usually a major consideration in the early stages of the design, and we will ignore them for now.

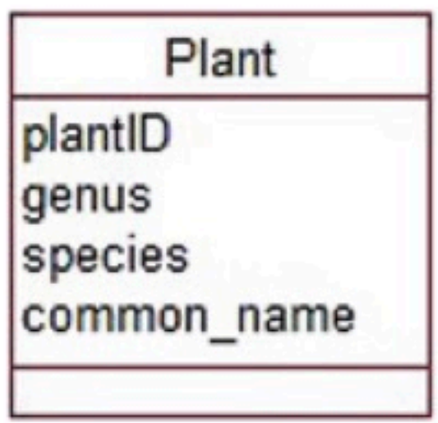


Figure 2-5. *UML notation for a class*

Each plant about which we want to keep data will conform to the template in Figure 2-5; that is, each will have (or could have) its own value for the attributes **`plantID`**, **`genus`**, **`species`**, and **`common_name`**. Each individual plant is referred to as an *object* of the **`Plant`** class. The **`Plant`** class and some objects are depicted in Figure 2-6.

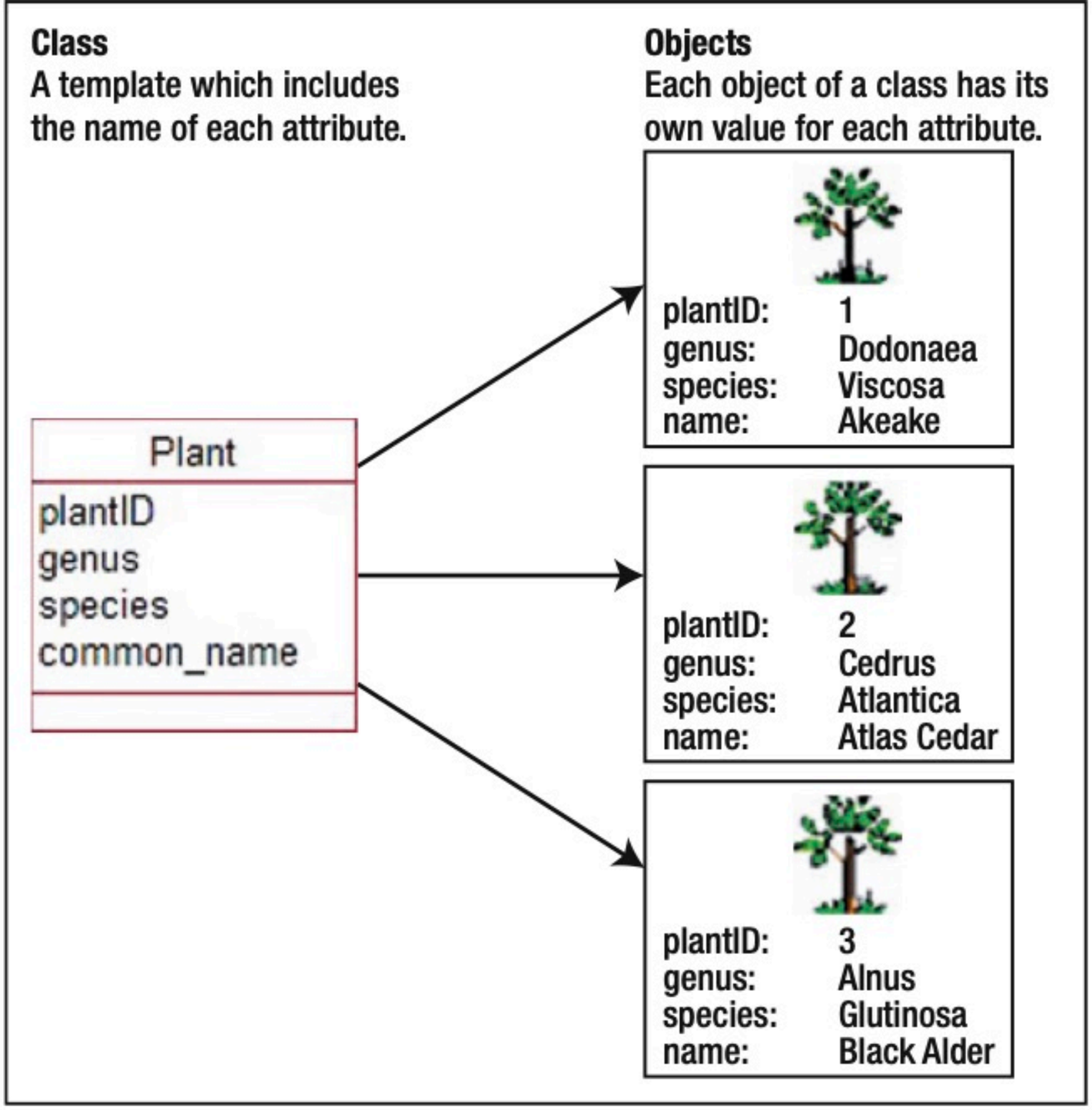


Figure 2-6. *A class and some of its objects*

The **`Plant`** class could include other attributes, such as typical height, lifespan, and so on. What about the uses to which a plant can be put? In the database table in Figure 2-3, these uses were included as several attributes (**`use1, use2`**, and so on) of a plant. In Example 1-1, we saw how having uses stored as several attributes caused a number of problems. What we have here is another candidate for a class: **`Use`**. In Chapter 5, we will discuss in more detail how we can figure out whether we need classes or attributes to hold information. Our new class, **`Use`**, will not have many attributes, possibly just **`name`**. Each object of the **`Use`** class will have a value for **`name`** such as "hedging," "shelter," or "bird food." What is particularly interesting for our example is the *relationship* between the **`Use`** and **`Plant`** classes.

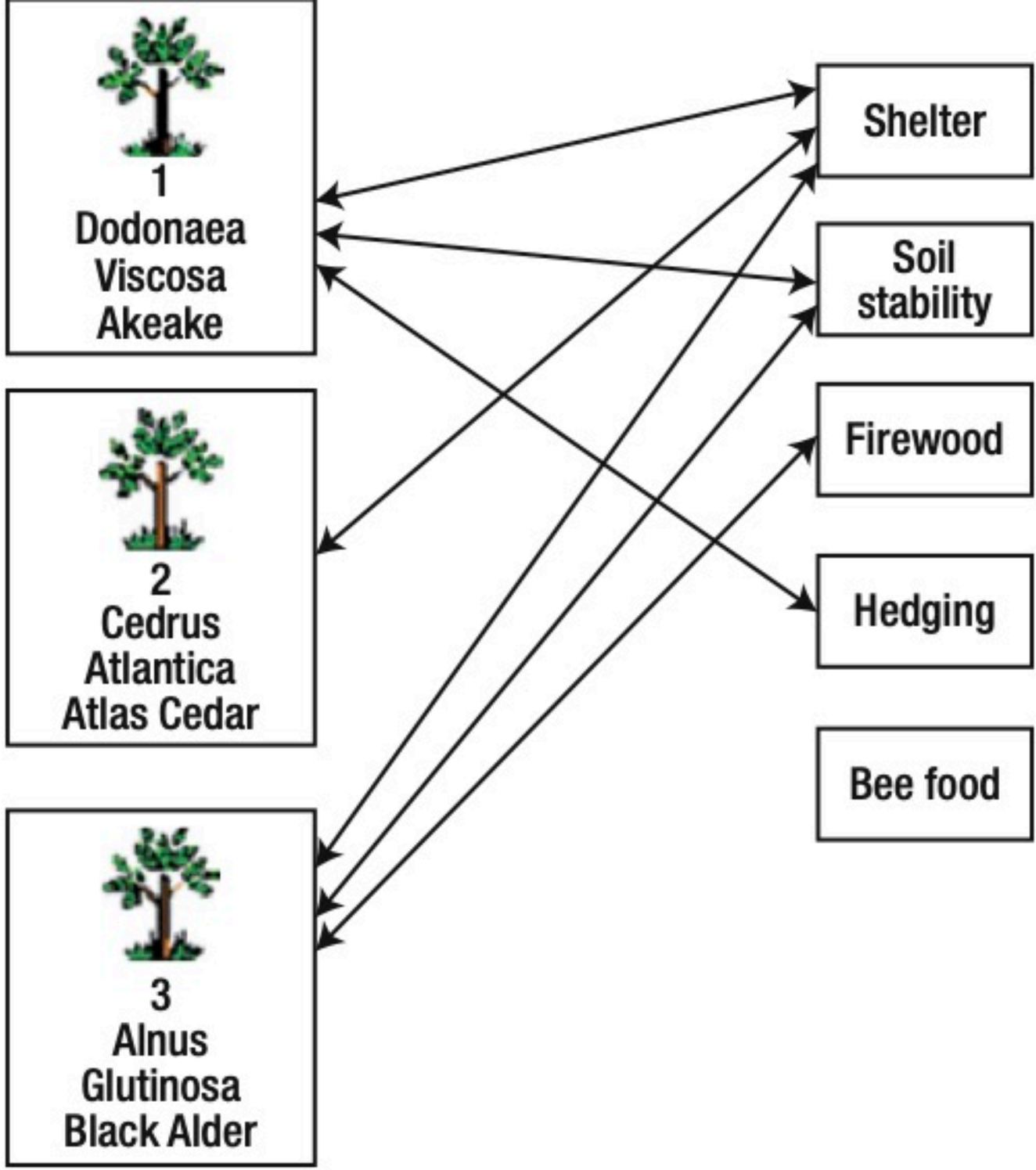


Figure 2-7. *Some instances of the relationship between Plant and Use*

In a database, we would usually create a table for each class, and the information about each object would be recorded as a row in that table as shown in Figure 2-8. The information about the specific relationship instances would also be recorded in a table. For a relational database, you would expect to find tables such as those in Figure 2-8 to represent the plants and relationship instances shown in Figure 2-7. We will look further at how and why we design tables like these in Chapter 7. For now, just convince yourself that it contains the appropriate information.

plantID	genus	species	common_name
1	Dodonaea	viscosa	Akeake
2	Cedrus	atlantica	Atlas cedar
3	Alnus	glutinosa	Black alder
4	Eucalyptus	nichollii	Black peppermint gum
5	Juglans	nigra	Black walnut
6	Acacia	mearnsii	Black wattle

Table Plant

Plant	Use
1	soil stability
1	hedging
1	shelter
2	shelter
3	firewood
3	soil stability
3	shelter

Table Plant Uses

Figure 2-8. *Plant objects and instances of the relationship between Plants and Uses expressed in database tables*

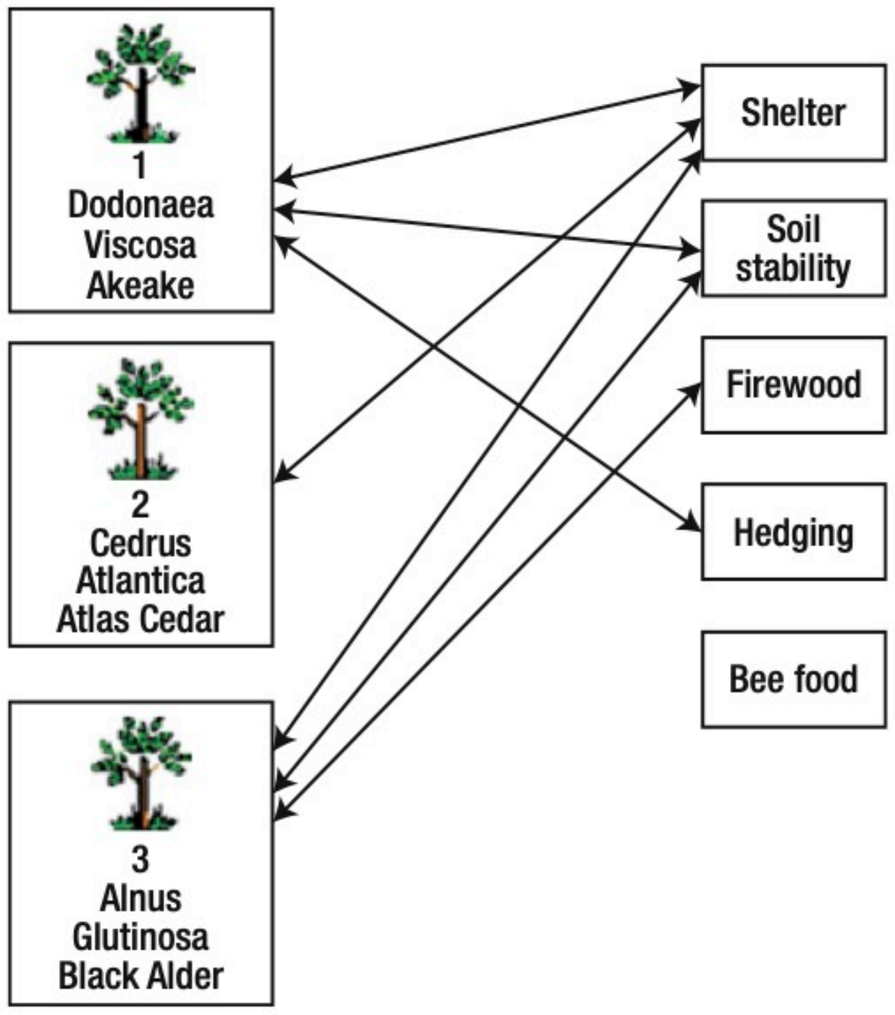


Figure 2-7. *Some instances of the relationship between Plant and Use*

In a database, we would usually create a table for each class, and the information about each object would be recorded as a row in that table as shown in Figure 2-8. The information about the specific relationship instances would also be recorded in a table. For a relational database, you would expect to find tables such as those in Figure 2-8 to represent the plants and relationship instances shown in Figure 2-7. We will look further at how and why we design tables like these in Chapter 7. For now, just convince yourself that it contains the appropriate information.

plantID	genus	species	common_name
1	Dodonaea	viscosa	Akeake
2	Cedrus	atlantica	Atlas cedar
3	Alnus	glutinosa	Black alder
4	Eucalyptus	nichollii	Black peppermint gum
5	Juglans	nigra	Black walnut
6	Acacia	mearnsii	Black wattle

Table Plant

Plant	Use
1	soil stability
1	hedging
1	shelter
2	shelter
3	firewood
3	soil stability
3	shelter

Table Plant Uses

Figure 2-8. *Plant objects and instances of the relationship between Plants and Uses expressed in database tables*

In UML, a relationship is represented by a line between two class rectangles, as shown in Figure 2-9. The line can be named to make it clear what the relationship is (e.g., "can be used for"), but it doesn't need to have a name if the context is obvious. The pair of numbers at each end of the line indicates how many objects of one class can be associated with a particular object of the other class. The first number is the minimum number. This is usually 0 or 1 and is therefore sometimes known as the *optionality* (i.e., it indicates whether there must be a related object). The second number is the greatest number of related objects. It is usually 1 or many (denoted n), although other numbers are possible. Collectively, these numbers can be referred to as the *cardinality* or the *multiplicity* of the relationship.

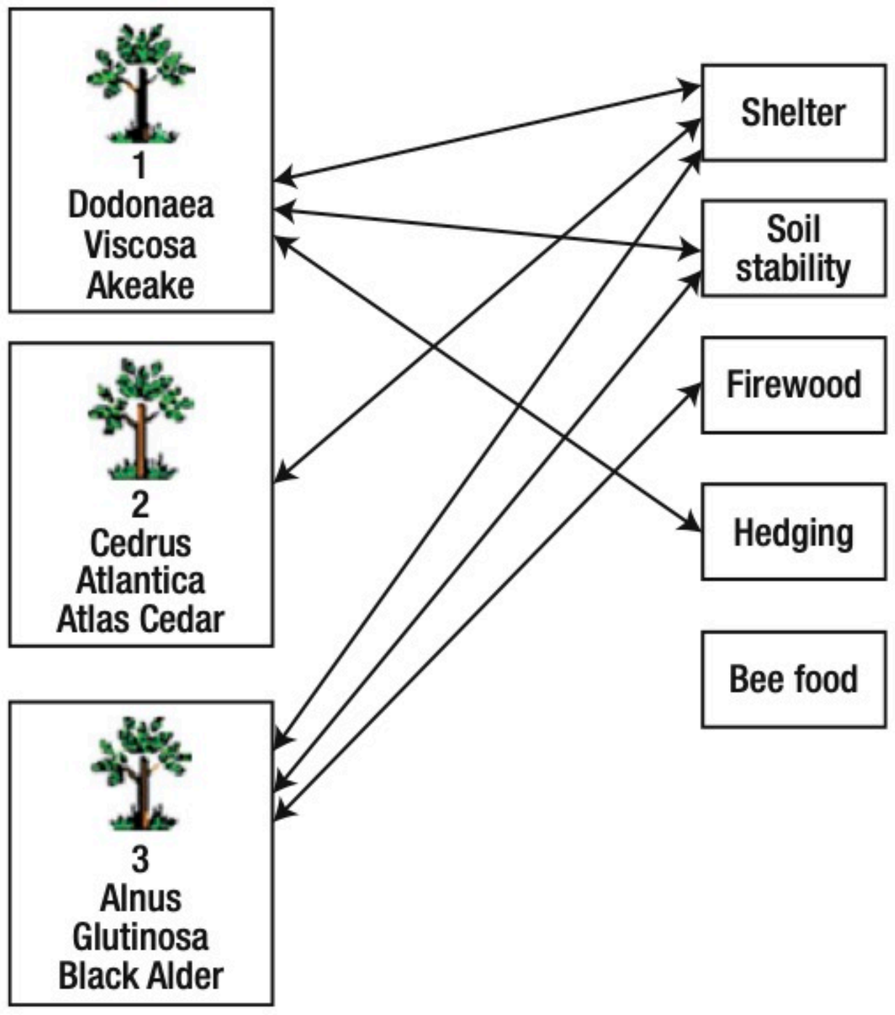


Figure 2-7. *Some instances of the relationship between Plant and Use*

In a database, we would usually create a table for each class, and the information about each object would be recorded as a row in that table as shown in Figure 2-8. The information about the specific relationship instances would also be recorded in a table. For a relational database, you would expect to find tables such as those in Figure 2-8 to represent the plants and relationship instances shown in Figure 2-7. We will look further at how and why we design tables like these in Chapter 7. For now, just convince yourself that it contains the appropriate information.

plantID	genus	species	common_name
1	Dodonaea	viscosa	Akeake
2	Cedrus	atlantica	Atlas cedar
3	Alnus	glutinosa	Black alder
4	Eucalyptus	nichollii	Black peppermint gum
5	Juglans	nigra	Black walnut
6	Acacia	mearnsii	Black wattle

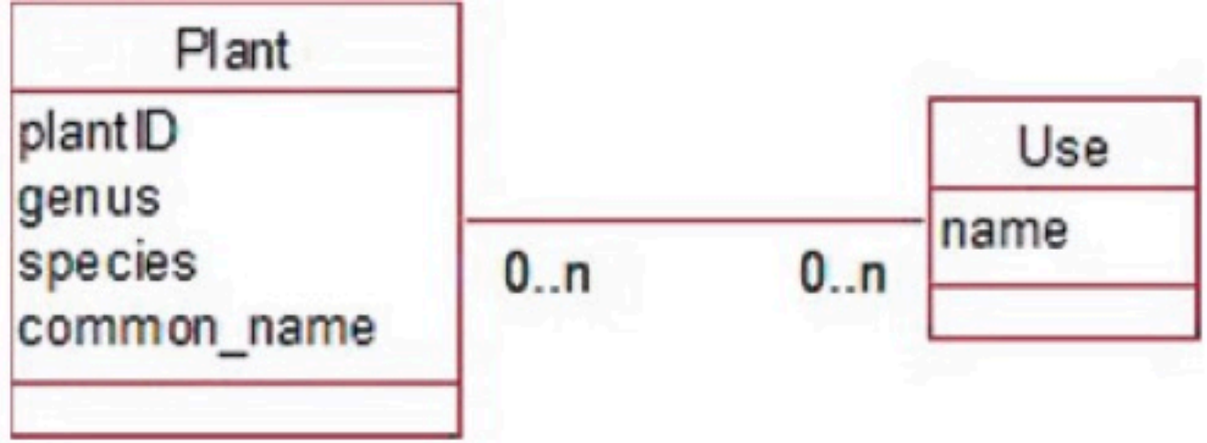
Table Plant

Plant	Use
1	soil stability
1	hedging
1	shelter
2	shelter
3	firewood
3	soil stability
3	shelter

Table Plant Uses

Figure 2-8. *Plant objects and instances of the relationship between Plants and Uses expressed in database tables*

In UML, a relationship is represented by a line between two class rectangles, as shown in Figure 2-9. The line can be named to make it clear what the relationship is (e.g., "can be used for"), but it doesn't need to have a name if the context is obvious. The pair of numbers at each end of the line indicates how many objects of one class can be associated with a particular object of the other class. The first number is the minimum number. This is usually 0 or 1 and is therefore sometimes known as the *optionality* (i.e., it indicates whether there must be a related object). The second number is the greatest number of related objects. It is usually 1 or many (denoted n), although other numbers are possible. Collectively, these numbers can be referred to as the *cardinality* or the *multiplicity* of the relationship.



re 2-11. *First attempt at a data model for plants example*

We now need to check whether this model is able to satisfy the requirements of the three use cases in re 2-4:

Use case 1: Maintain plant information. We can create objects for each plant and record the attributes we might require now or in the future. We can create use objects, and we can specify relationship instances between particular plant and use objects.

Use case 2: Report on plants. We can take a particular plant object (or each one in turn) and find the values of its attributes. We can then find all the use objects related to that plant object.

Use case 3: Report on uses. We can take a particular use object and find all the plant objects that are related to it.

We now realize that we have a new class, **Genus**, to add to our data model. Why is it important to include this new class? Well, if genus remains as simply an attribute of our original **Plant** class, we can enter pretty much any value for each object. Two objects with genus *Eucalyptus* might end up with different spellings (almost certainly if I were doing the data entry). This would cause problems every time we wanted to find or count or report on all *Eucalyptus* plants. The fact that our user has mentioned that grouping by genus would be useful means that it is important to get the genus data stored appropriately. Our revised data model in Figure 2-12 shows how genus can be represented so that the data is kept accurately.

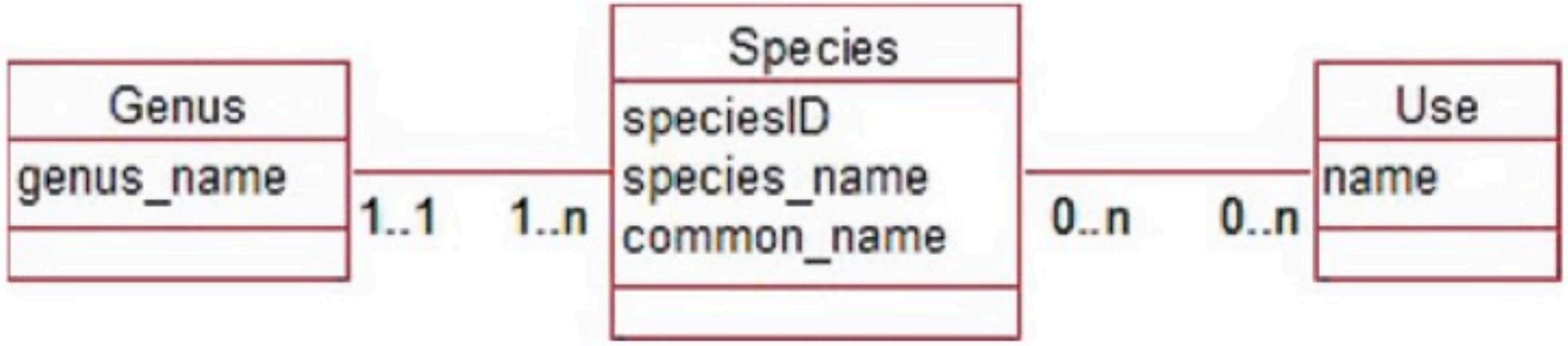
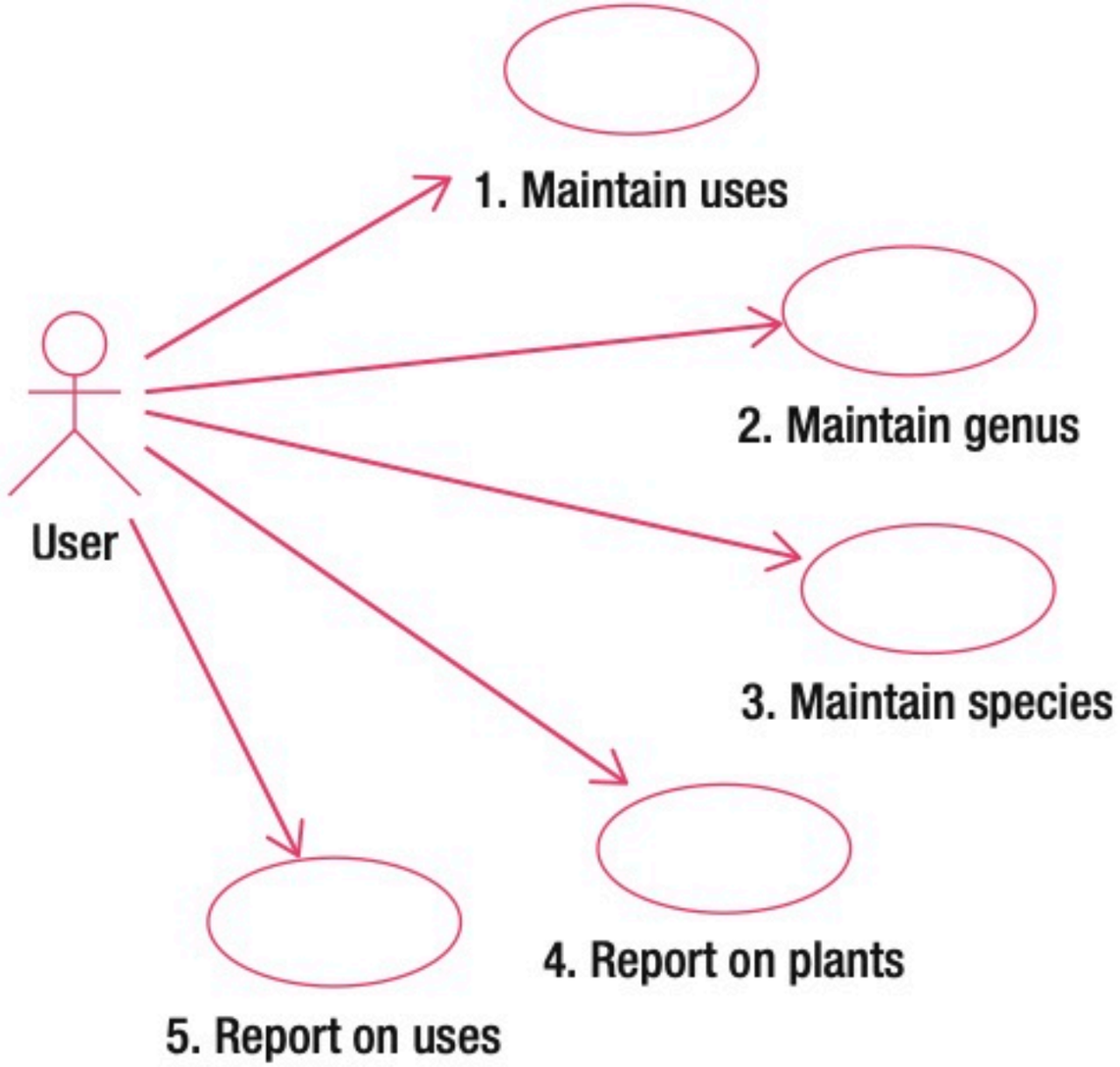


Figure 2-12. *Revised data model for our plant problem*



ure 2-13. *Revised use cases for the plant problem*

Use case 1: Maintain uses. Create or update a use object. Enter (or update) the name.

Use case 2: Maintain genus. Create or update a genus object. Enter the name.

Use case 3: Maintain species. Create a species object. Generate a unique ID, and enter the species and common name. Associate the new species object with one of the existing genus objects and optionally associate it with any number of the existing uses.

Use case 4: Report plant information. For each genus object, write out the name and find all the associated species objects. For each species object, write out the species and common name. Find all the associated uses and write out their names.

Use case 5: Report use information. For each use object, write out the name. Find all the associated species objects, and write out for each the associated genus name and the species and common names.

In very broad terms, each class will be represented by a database table. Because each species can have many uses and vice versa, we need an additional table for that relationship. This is generally the case for relationships having a cardinality greater than 1 at both ends (known as Many–Many relationships). (There will be more about these additional tables in Chapter 7.) The tables are shown in Figure 2-14 as they would look in Microsoft Access. Three tables correspond to the classes in Figure 2-12 and the extra table, **`PlantUse`**, gives us somewhere to keep the relationships between plant species and uses (Figures 2-7 and 2-8). The other relationships between the classes can be represented within the database by setting referential integrity between the four tables (more about this in Chapter 7).

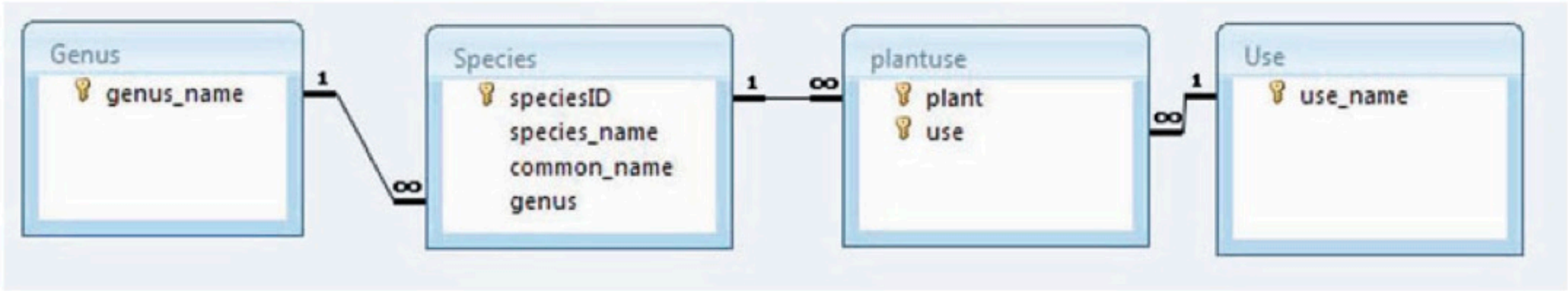


Figure 2-14. *Representing classes and relationships in Microsoft Access*

genus_name
Acacia
Aesculus
Alnus
Aristotelia
Betula
Boronia
Brachyglottis
Cedrus
Chaenomeles
Chionochloa
Clianthus

speciesID	species_name	common_name	genus
1	viscosa	ake-ake	Dodonaea
2	atlantica	atlas cedar	Cedrus
3	nigra	black walnut	Juglans
4	melanoxylon	Tasmanian blackwood	Acacia
5	hippocastanum	Horse Chestnut	Aesculus
6	glutinosa	Black alder	Alnus
7	incana	grey alder	Alnus
8	cordata	Italian alder	Alnus
9	serrata	Wineberry ; Mako Mako	Aristotelia
10	pendula	Silver birch	Betula

Table Genus

Table Species

(The value of genus must be one of the values in the Genus table)

use
bee food
bird food
coppicing
firewood
hedging
shelter
soil stability
timber

plant	use
1	hedging
1	shelter
1	soil stability
2	shelter
3	firewood
3	shelter
3	soil stability
4	bird food

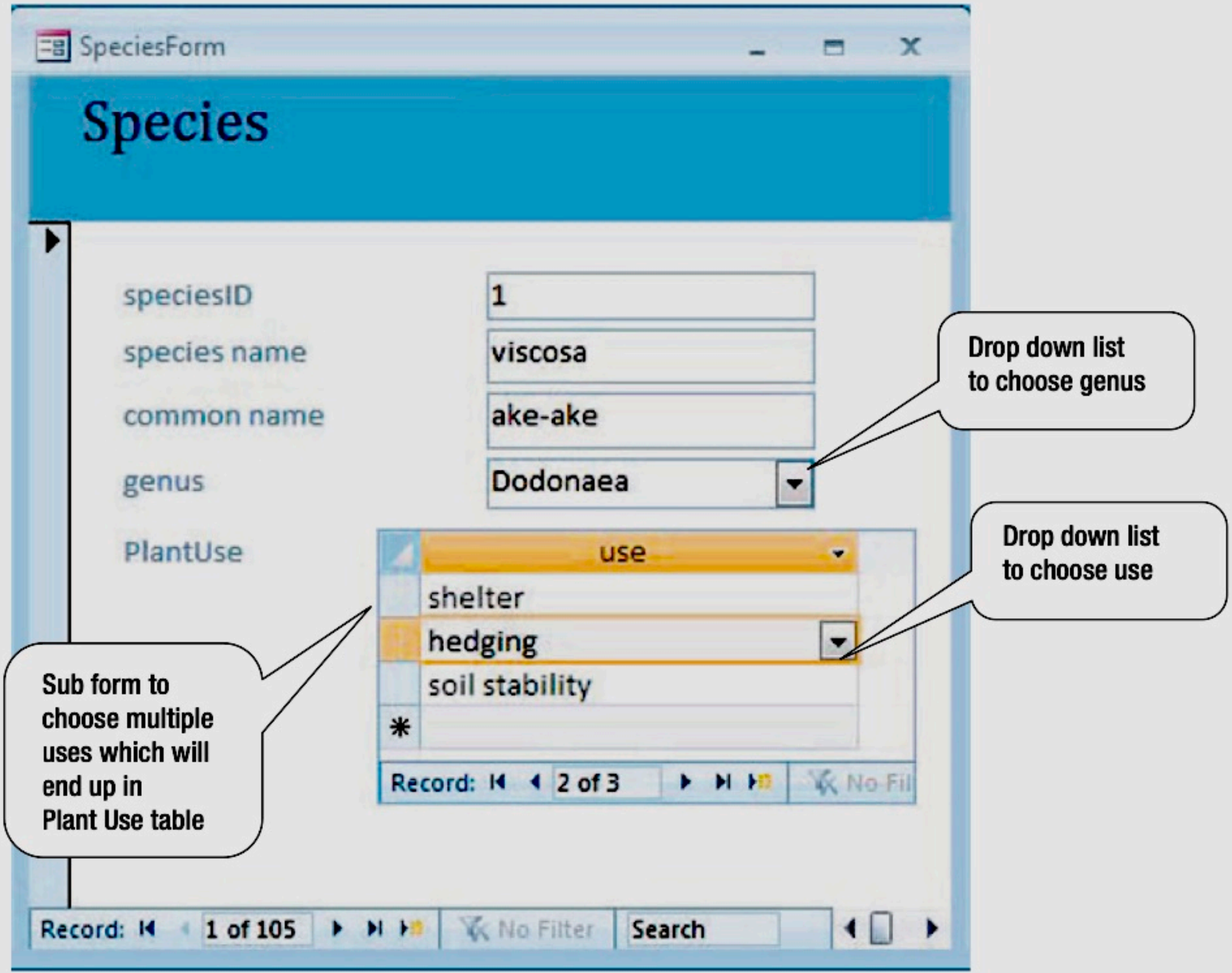
Table Use

Table PlantUse

(The value of plant must be one of the values in the Species table. The value of use must be one of the values in the Use table)

2-15. *Example data in tables for the plant database*

Figure 2-16 shows a very basic form for entering data about a particular species, It was created using the Form Wizard in Microsoft Access. This form allows us to enter data that will end up as one row in the **Species** table and several rows in the **PlantUse** table (one for each use for this particular species). The form also provides convenient ways to establish the relationships between a species and its genus and uses by providing drop-down lists that will contain each of the possible genus or use objects. This is one possible solution to satisfy the requirements of use case 3 (maintaining species data) in an accurate and convenient way.

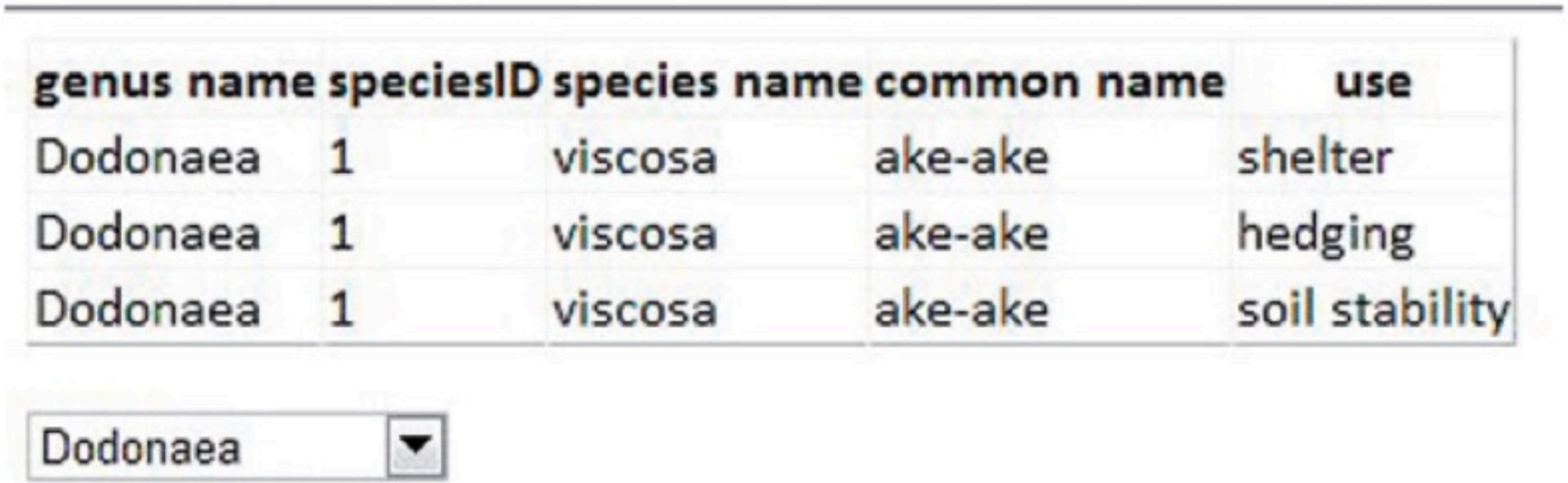


PlantUse

Use	ID	Genus	Species Name	Common Name
bird food				
	4	Acacia	melanoxylon	Tasmanian blackwood
	7	Alnus	incana	grey alder
	28	Eucalyptus	nichollii	Black peppermint gum
coppicing				
	30	Eucalyptus	gunnii	cider gum
	4	Acacia	melanoxylon	Tasmanian blackwood
	28	Eucalyptus	nichollii	Black peppermint gum
firewood				
	6	Alnus	glutinosa	Black alder
	3	Juglans	nigra	black walnut
hedging				
	1	Dodonaea	viscosa	ake-ake

2-17. *A simple report satisfying the use case for providing information on plants suitable for a specific use*

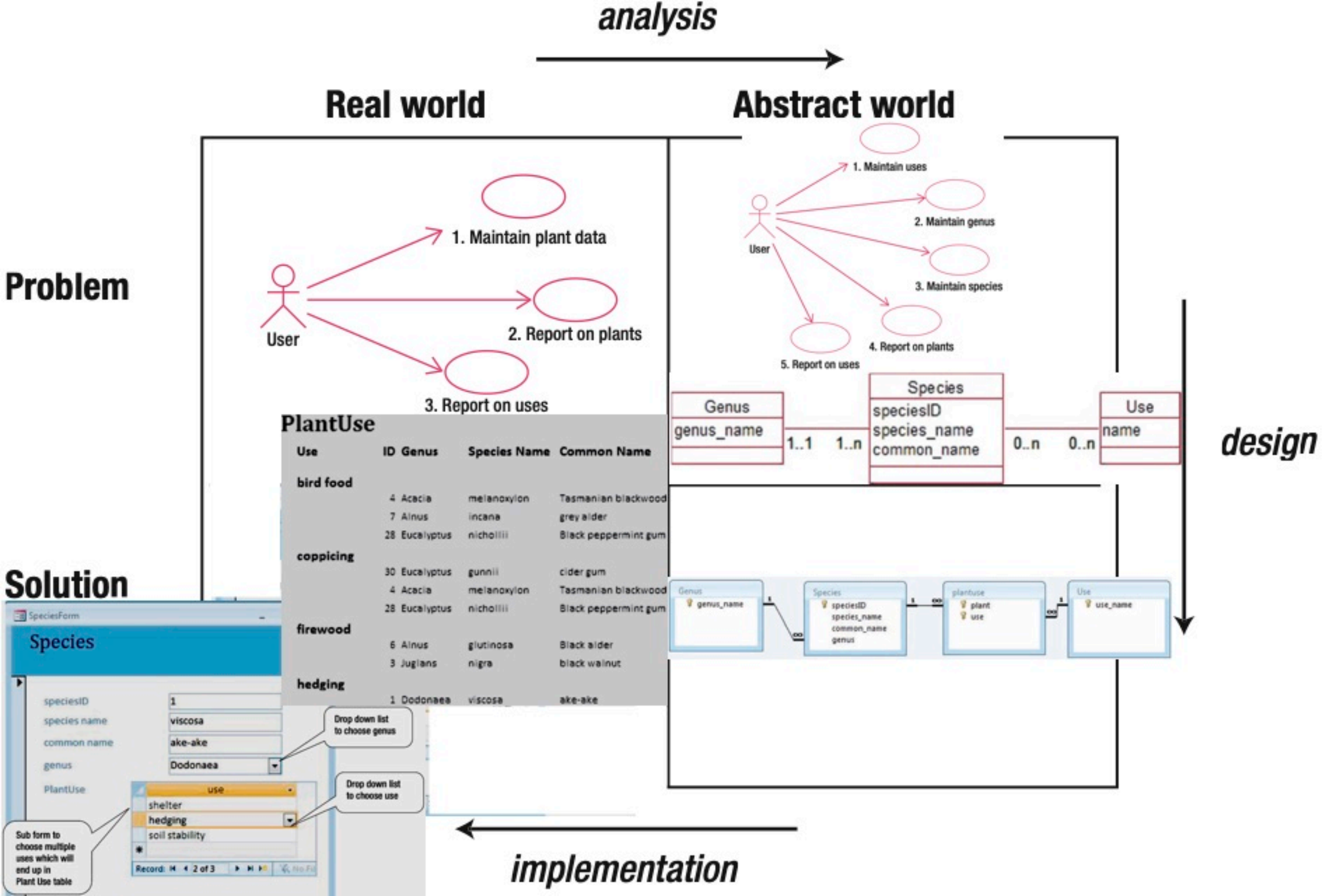
We could create a similar report to Figure 2-17, by grouping our data by genus instead of use. However, there are many different ways to access information from the database. Figure 2-18 shows a very simple web page view of our Access database. It allows users to select a genus and to see the associated species and uses (the web page was developed with Microsoft Expression Web).



genus name	speciesID	species name	common name	use
Dodonaea	1	viscosa	ake-ake	shelter
Dodonaea	1	viscosa	ake-ake	hedging
Dodonaea	1	viscosa	ake-ake	soil stability

Dodonaea

Figure 2-18. *A simple web page front end satisfying the use case for returning plant information grouped by genus*


analysis
Real world
Abstract world
Problem
1. Maintain plant data
User
2. Report on plants
3. Report on uses
1. Maintain uses
2. Maintain genus
User
3. Maintain species
4. Report on plants
5. Report on uses
Genus
genus_name
1..1
1..n
Species
speciesID
species_name
common_name
0..n
0..n
Use
name
design
PlantUse
Use ID Genus Species Name Common Name
bird food
4 Acacia melanoxylon Tasmanian blackwood
7 Alnus incana grey alder
28 Eucalyptus nichollii Black peppermint gum
coppicing
30 Eucalyptus gunnii cider gum
4 Acacia melanoxylon Tasmanian blackwood
28 Eucalyptus nichollii Black peppermint gum
firewood
6 Alnus glutinosa Black alder
3 Juglans nigra black walnut
hedging
1 Dodonaea viscosa ake-ake
Solution
SpeciesForm
Species
speciesID 1
species name viscosa
common name ake-ake
genus Dodonaea
PlantUse
use
shelter
hedging
soil stability
Record: 2 of 3
Drop down list to choose genus
Drop down list to choose use
Sub form to choose multiple uses which will end up in Plant Use table
implementation

A small sports club keeps information about its members and the fees they pay. The secretary wants to be able to record when members pay and print a report similar to that in Figure 2-20.

last_name	first_name	phone	type	gender	fee	date_paid
Smith	Jane	563201	Full	F	220	21/09/2011
Wilson	Harry	375967	Full	M	220	19/09/2011
Green	Bert	439871	MidWeek	M	150	
Jones	Bert	295784	Social	F	80	
Smith	Sharon	387648	MidWeek	F	150	16/08/2011

ure 2-20. *Membership data for a small club*

a) Think about when the different pieces of data might be entered. Sketch an initial use case diagram for data entry.

b) Consider what different things you are keeping information about and sketch a simple class diagram.

c) What options could you suggest to the club for different ways a report could be presented? Does your class diagram have the information readily available?