Sentences Tutorial

Sentences™ Version 3.5

A publication of:

Lazy Software Ltd.
Gemini House, Mercury Park
Wycombe Lane
Wooburn Town
Bucks HP10 0TT
UK

http://www.lazysoft.com
Email: info@lazysoft.com
Phone: 01628 642300
Fax: 01628 642301

Document Information
Ref: SNT/TUT/3.5/01
Group: User Documentation
Edition: 01
Date: April 2003

Table of Contents

The Sentences Tutorial

Introduction

Sentences is a multi-user, web-enabled database management system written in Java which includes its own development and deployment tools. It is the first commercial application to be based on the Associative Model of Data.

To find out more about the concepts that lie behind Sentences, read Simon Williams' book *The Associative Model of Data*, available from Lazy Software's Web site (http://www.lazysoft.com).

The commercial version of Sentences, called the Enterprise Edition, runs on a Web server as a Java servlet with client access through a conventional Web browser running a Java applet. The evaluation version of Sentences, called the Personal Edition, runs as a Java application. You can use either the Enterprise Edition or the Personal Edition for this tutorial.

This tutorial is a step-by-step introduction to using Sentences. Follow the instructions in this tutorial to create an example Skills register application for IT staff. This application deals with information about the programmers who work for a company, the programming languages that they each know, their levels of proficiency, their current project assignment, and more. Detailed information about all the features introduced in this tutorial is available in the *Sentences User's Guide*.

Note *If you are using the Enterprise Edition of Sentences a more sophisticated example of a Skills register application is available as part of the Application Suite.*

This tutorial assumes that you have a working knowledge of your computer's operating system, and are familiar with Web browsers. You should also understand how to use menus and commands, and how to use a mouse.

Document conventions

Please note that in this tutorial:

- *Type in* means to enter data with your keyboard
- *Press* means press a key on your keyboard

- *Click* means move your mouse pointer over a button or other item on the screen and press once on the primary button on your mouse (usually the left button if you are right-handed).
- *Double-click* means move your mouse pointer over a button or other item on the screen and press twice rapidly on the primary button on your mouse (usually the left button if you are right-handed).
- *Right-click* means move your mouse pointer over a button or other item on the screen and press once on the secondary button on your mouse (usually the right button if you are right-handed).
- *Shortcut menu* means the pop-up context menu displayed when you right-click on an object.

This document uses the following typographical conventions:

- Names of menus, commands, dialogs, and fixed elements in Sentences are shown in **boldface** font. For example:
  the **All types** folder
  the **Expand node** command on the **View** menu
  the **Properties** toolbar button.
- Names of actual files and directories are shown in `monospaced font`. For example:
  the `Profiles.chap` file.
- Example data items are shown in a sans serif font. When documenting association types and associations in Sentences, the verb is shown in *sans serif italics*, and nested associations are enclosed in parentheses.
  For example:
  Person
  Project
  Person, *assigned to*, project
  (Louise East, *assigned to*, Pisces), *starting date*, 25th March 2000

Introducing the associative model of data

Like other databases, the Sentences database is designed to record information about things in the real world. Sentences is based on the associative model of data, described in Simon Williams' book *The Associative Model of Data*, which is available from Lazy Software.

This section of the tutorial introduces some of the key concepts in the associative model of data.

Entities and associations

The associative model divides real-world things into two sorts: entities and associations.

- **Entities** are things that have discrete, independent existence. An entity's existence does not depend on any other thing. Some types of things that would be represented by entities are houses, people, bicycles, and buildings.
- **Associations** represent links or interaction between other things. Their existence depends on one or more other things which may themselves be entities or other associations. If any of those things ceases to exist, then the thing represented by the association ceases to exist or becomes meaningless. Some types of things that would be represented by associations are customers, marriages, or corporate headquarters.

For example:

- A customer is an association between two people, or between a person and a legal entity, or between two legal entities.
- A marriage is an association between two people.
- A corporate headquarters is an association between a corporation and a building or a location.

An association may depend upon another association: for example, a sales order may depend on a customer, which is itself an association. Similarly each line of a sales order depends on the sales order itself.

The distinction in the associative model of data between entities and associations allows you to distinguish between things that have discrete, independent existence, and the various different ways in which such things interact with other things. Each such interaction is a thing in its own right, about which you may want to record information.

For example:

- A person is an entity, whilst a person's roles as a customer, an employee, a spouse, or a team member are associations.

- An enterprise is an entity, whilst an enterprise's roles as a customer, a supplier, or a tenant are associations.

In Sentences, an association has three components, the source, the verb and the target. Sources and targets may be either entities or associations. The verb describes the nature of the link between the source and the target.

Entities and values

Entities and associations are the two kinds of real world things that you want to record information about in an application. There are other things in the real world that you do not need to record additional information about.

Numbers, dates, and times in the real world do not have any qualifying attributes of their own, and so in the associative model they are known as values. In Sentences, values are special kinds of entities.

Values by definition do not need to be defined or described by associations, and therefore values can only be used as the targets of associations and never as the sources of associations. Apart from this restriction the behaviour of values is very much the same as the behaviour of entities. They are reusable, and the same value can be the target for more than one association.

Creating the example application

In this tutorial you take on the imaginary role of a manager of a software development group (you may already be one). You are going to create a Sentences application that allows you to track the skills and current assignments of your programmers.

Starting the database schema

The first step in creating an application is to outline the data items to be recorded, and the relationships between those data items. This is called the database schema. This first part of this tutorial shows you how to build the schema for your application and then add some data to it.

Your application deals with such things as programmers, programming languages, and project assignments. Examples of the data items that you can record in the application are: "Jim is skilled in Java", "Jim is working on project Capricorn", and "Project Capricorn runs from 1st February until 30th June".

Using the terms of the associative model, programmers, programming languages, and projects can be represented in Sentences by entities. The statements "Jim is

skilled in Java", "Jim is working on project Capricorn", and "Project Capricorn runs from 1st February until 30th June" can be represented in Sentences by associations.

The first step in building a Sentences application is to define patterns of data called types. You can define certain attributes for a type that are shared by all the data items that are instances of a that type. Sentences uses both entity types and association types which together make up the database schema for an application.

You do not have to construct the whole of your database schema in Sentences before you can start to add data. You can define some entity and association types to build part of the schema, add some entity and association data, and then test, develop and refine the schema.

The schema for the example Skills register application includes definitions of Person and Programming language as entity types, and a definition of Person, *is skilled in*, Programming language as an association type.

Association types always have three parts which are known as the source (Person in this example), the verb (*is skilled in*) and the target (Programming Language). The source and the target may be entity types or association types. Targets may also be value types.

More details about the use of types are given later in this book, and full details can be found in the *Sentences User's Guide*.

Running Sentences

If you are using the Sentences Enterprise Edition you can access the Sentences server by entering the location and name of your Sentences start page in your web browser, or by clicking a link on a Sentences start page. You may need to contact your local system administrator or network manager for details of the correct way to access Sentences on your system.

If you are using the Sentences Personal Edition, simply start the application.

Sentences displays the Sentences Explorer showing the Human Resources example application. In the Enterprise Edition the Sentences Explorer may run inside a Web browser window.

The Sentences Explorer

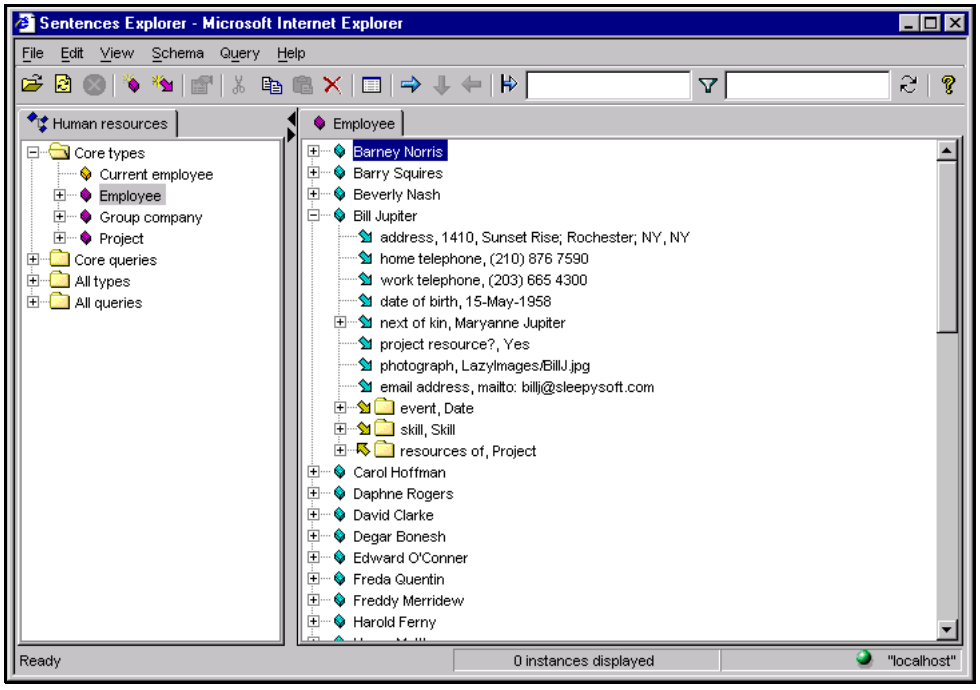

Figure 1 **The Sentences Explorer**

The Sentences Explorer has its own toolbar, menu bar and status bar, and is divided into two panes, with the schema pane on the left and data pane on the right. The information in each of the panes is shown as a tree display.

The Explorer Menu bar

The Sentences Explorer menu bar provides access to a number of pull-down menus. The menu names and the groups of commands on each menu are listed below.

menu name...	has commands about...
File	Create a new profile, or open, edit, delete, and refresh existing profiles; reload data based on Positioner and Filter settings
Edit	Edit data and view Properties

menu name…	has commands about…
View	Change the current Explorer display; display additional database information; open the Diagram Editor
Schema	Edit and build your schema
Query	Use queries, and work with query results
Help	Help topics, Tutorial, and support information

Some menu commands are only available when certain items are selected in the Explorer. Some menu commands are not available in certain Sentences editions, or may be disabled for certain users.

A full list of menu commands is given in Appendix A, ‘Menus and Commands summary’ in the *Sentences User’s Guide.*

The Explorer Toolbar

The Sentences Explorer toolbar is located beneath the menu bar. The toolbar buttons give rapid access to some frequently used commands.

Button	Description	Action
	open profile	use the Open Profile dialog to open a profile
	refresh profile	refresh the data in the current profile
	stop	stop retrieving data from the server
	create entity type	create a new entity type
	create association type	create a new association type

Button	Description	Action
	properties	open the properties dialog for the selected entity or association; open the Query Editor for the selected query
	cut	cut the selected item
	copy	copy the selected item
	paste	paste the selected item
	delete	delete the selected item; displays the delete impact dialog
	Dataform	open the Dataform for the selected entity type or entity instance or association instance
	new tab	opens a new Sentences Explorer tab on the selected item
	new tab on target	opens a new Sentences Explorer tab on the target of the selected association
	close tab	closes the current tab
	Positioner value field	data entry field for the Positioner tool
	Filter value field	data entry field for the Filter tool
	reload data	reloads data using current Positioner and Filter conditions; displays the results of a selected query
	help	displays help

Creating chapters and profiles

Sentences stores schema information and application data in files known as *chapter files*. Access to data is managed through a *Profile*, which is a named collection of schema and data chapter files.

To start work on the example application you must create a new profile and new chapter files. Although you could store schema information and application data in a single chapter file, using separate schema and data chapter files is recommended.

If several users may be working through this tutorial on the same Sentences installation you may wish to give unique names to the profile and chapter files you use in the tutorial, for example, by adding your initials to each name.

Note *The Sentences server process must have permission to create files in the chapters location.*

Follow these steps to create a new profile and new chapter files:

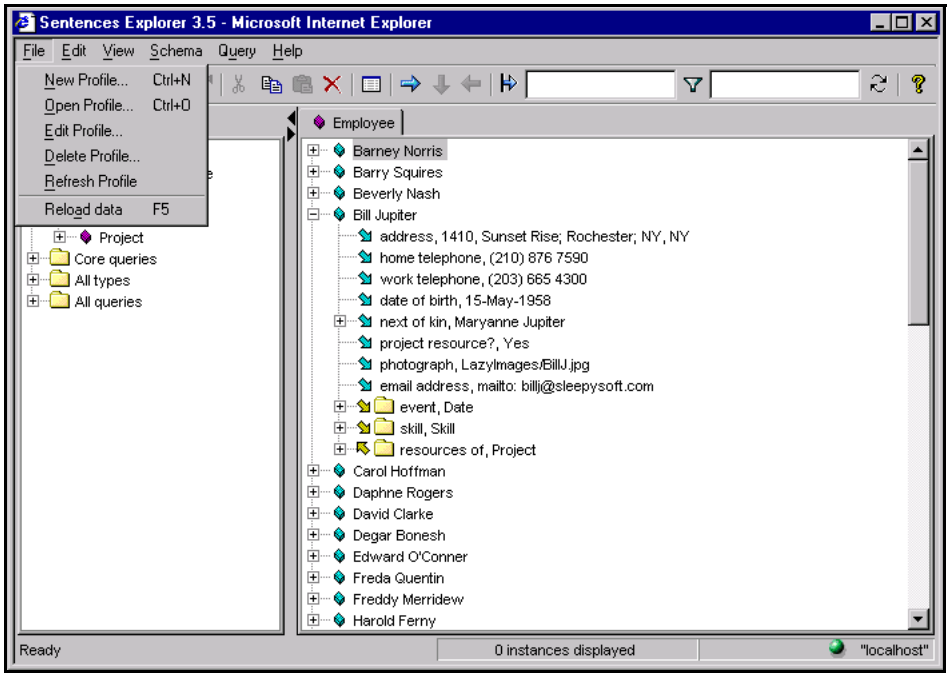

Figure 2 **The Sentences Explorer File menu**

1. Select **New Profile** from the **File** menu. (Remember that all the menu commands in this tutorial refer to the menus in the Sentences applet, and not in the Web browser window which may also be displayed.) Sentences displays a blank **Edit Profile** dialog.

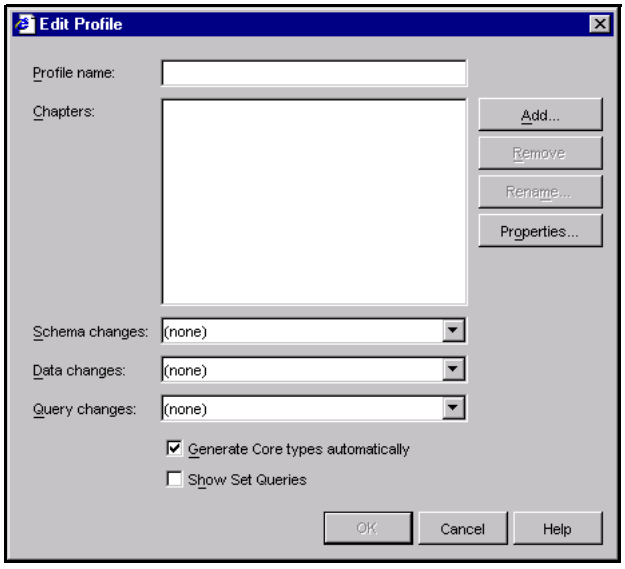

Figure 3 **The Edit Profile dialog**

Use the **Edit Profile** dialog to define the name of the profile, the list of chapters in the profile, and the chapters used for saving changes to your application's schema data, and queries.

2. In the **Profile Name** field type in the name Skills register.
3. Click **Add** to add one or more chapters to your profile. Sentences displays the **Add Chapter** dialog as shown in Figure 4.

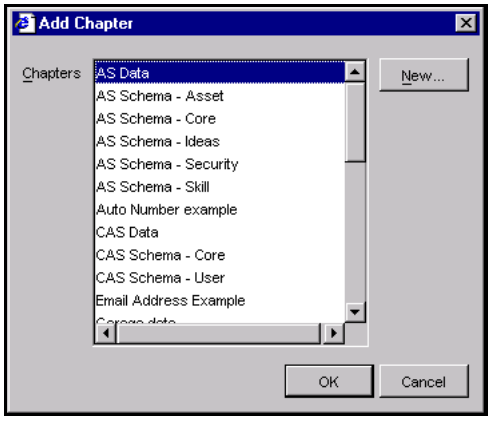

Figure 4 **The Add Chapter dialog**

4. The **Add Chapter** dialog shows the names of all the chapter files on your system. The actual list shown depends on whether you are using the Enterprise Edition or the Personal Edition of Sentences, and on the applications that exist in your Sentences installation.

   Click **New** to create a new chapter file. Sentences displays the **New Chapter** dialog as shown in the illustration Figure 5.

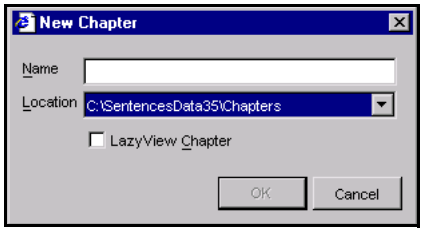

Figure 5 **Creating a new chapter**

   You can store Sentences chapters in more than one location, but these locations must first be defined by your Sentences administrator. Click the down arrow next to the **Location** field to see if alternative locations have been defined for your Sentences installation. If necessary, consult your Sentences administrator about the best location for your tutorial files.

5. In the **New Chapter** dialog, type in the name for your new chapter and click **OK**. Create a chapter named `Skills register data`. Click **OK** to return to the **Add Chapter** dialog. Repeat the last two actions to create a chapter named

`Skills register schema`, and then a chapter named `Skills register queries`.

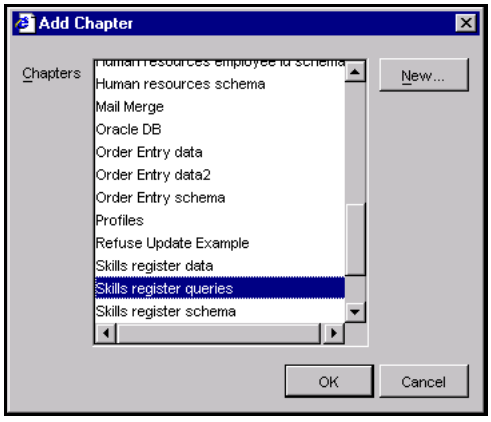

Figure 6 **New chapters in the Add Chapter dialog**

6. In the **Add Chapter** dialog, highlight one of the chapters you want to add to your profile (`Skills register data`, `Skills register schema`, or `Skills register queries`). To select another chapter, hold the **Control** button and click again. Click **OK** to return to the **Edit Profile** dialog.

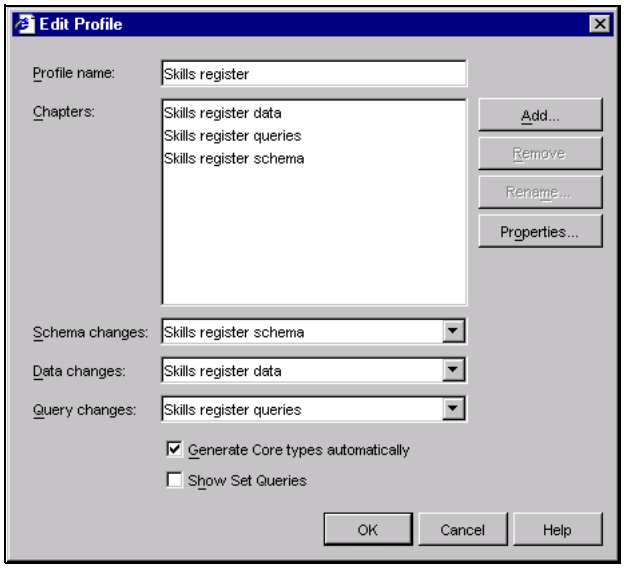

Figure 7 **The Edit profile dialog showing the new chapters**

7. Select `Skills register schema` as the **Schema changes** chapter, `Skills register data` as the **Data changes** chapter, and `Skills register queries` as the **Query changes** chapter.

8. Place a check mark in the **Generate Core types automatically** checkbox.

   The Sentences Explorer has a special folder which contains copies of some of the entity types in your schema, called the **Core types** folder. When you are working with a large schema, the **Core types** folder gives you quick access to frequently used types. You can either choose the types for the **Core types** folder yourself, or ask Sentences to do this automatically, as you have done here. The way Sentences makes this selection is explained later.

9. The Edit Profile dialog should now look like the illustration in Figure 7. Click **OK** to complete creating your profile.

Figure 8 **Change Profile confirmation dialog**

Sentences displays a confirmation dialog which prompts you to save the changes to your profile. Click **Yes** to confirm and to return to the Sentences Explorer interface.

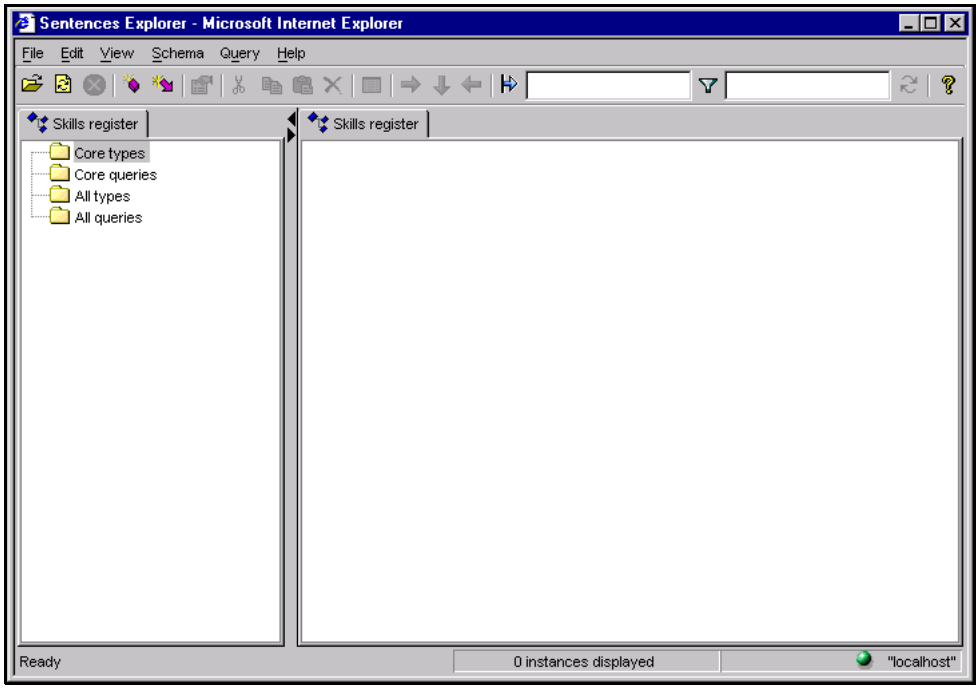

Figure 9 **New profile name shown as Sentences root**

The profile name appears on the tab above the schema pane (the left-hand pane) in the Explorer.

If you wish, you can stop work on the tutorial at this point and take a break. Any changes you make using a Sentences Explorer dialog or a Sentences Dataform are automatically saved in the Sentences database whenever you click a **Save** button.

Creating your first entity type

You are now ready to create some entity types. To start building the schema for the Skills register application you need to create entity types for Person and Programming language.

To add a new entity type you can perform one of the following actions:

- Select **Create Entity Type** from the **Schema** menu
- Click the **Create Entity Type** button (the magenta cube) on the Sentences Explorer toolbar
- Right-click the **All types** folder in the Sentences Explorer, and select **Create Entity Type** from the shortcut menu.

The following example illustrates using the shortcut menu. All the subsequent steps in this procedure are the same, whichever way you start.

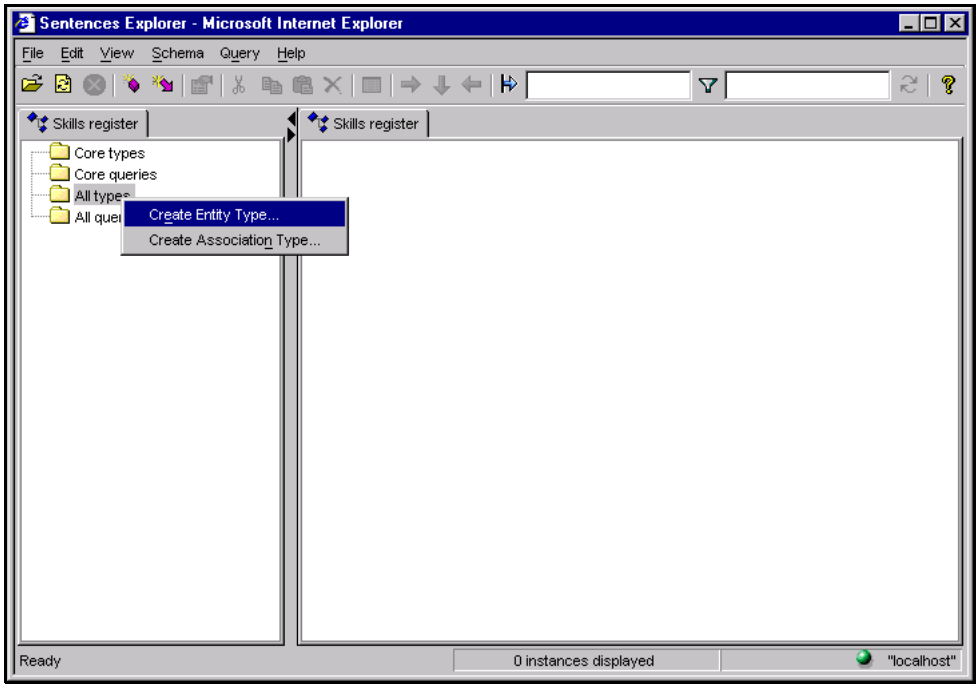

Figure 10 **Entity Type shortcut menu**

1. Right-click the **All types** folder and select **Create Entity Type** from the shortcut menu.

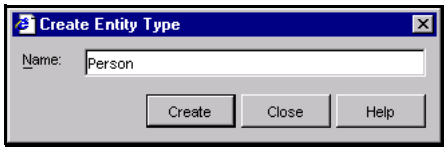

Figure 11 **Create Entity Type dialog**

2. Type in Person as the name of your first new entity type in the **Create Entity Type** dialog, and click **Create**.

You can create more entity types by typing in their names and clicking **Create** after each one. Create another entity type named Programming language.

3. After creating the Person and Programming language entity types, click **Close**.

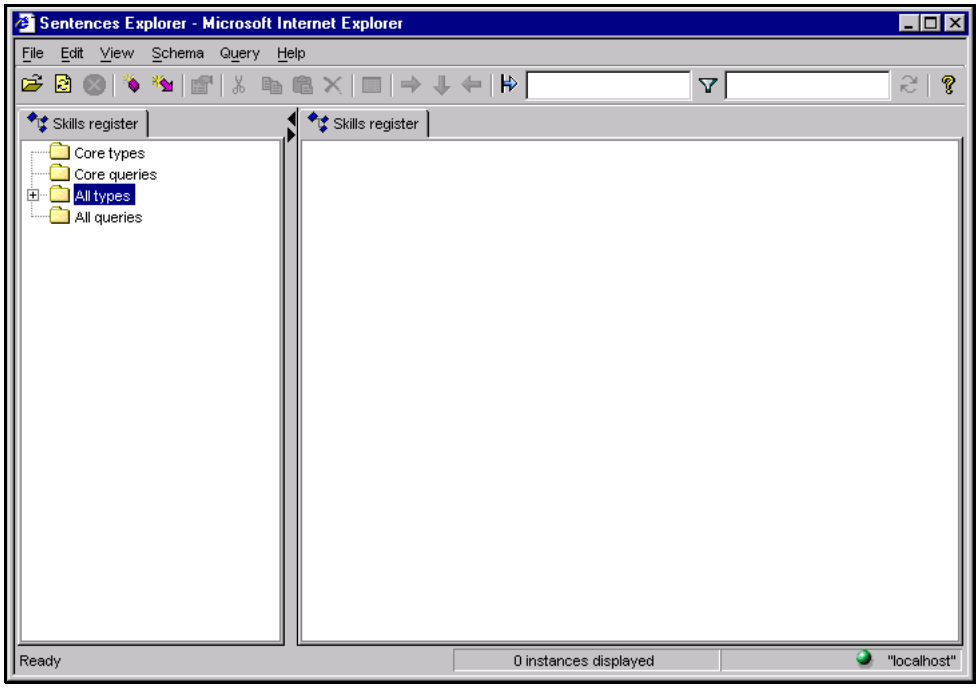

Figure 12 **Expand sign on All types folder**

The **All types** folder now has a expand sign (+) next to it, as shown in Figure 12. Click the expand sign to expand the entity type tree and to view your new entity types.

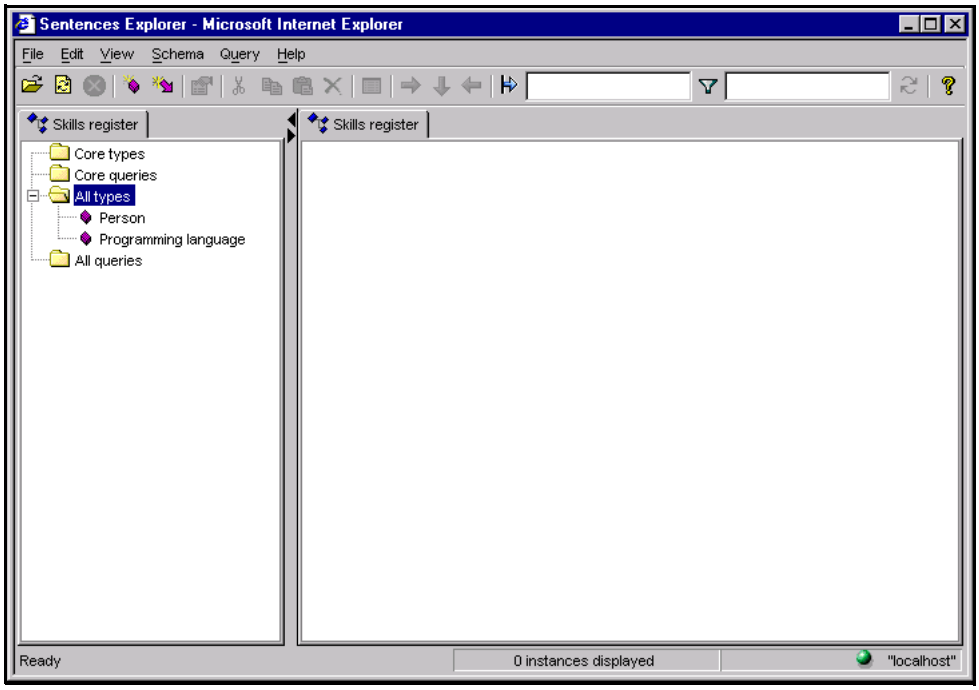

Figure 13 **Expanded All types folder**

Figure 13 shows the entity types you created displayed under the **All types** folder.

Adding entity instance data

When you are creating an application in Sentences you can create some parts of your schema, add some data, and then continue to build the schema.

All entity and association instance data in Sentences is added using the Dataform. The Dataform is Sentences' dynamic interface for adding and reviewing entity and association data. It is dynamic because the content of the Dataform display changes as your application schema grows.

You can create custom Dataforms in Sentences, but that is beyond the scope of this tutorial. The *Sentences User's Guide* explains how to create custom Dataforms.

You can now start adding entity data for the Programming language entity type. To begin, right-click the entity type name Programming language and select **Default Dataform** from the shortcut menu. You could also highlight the entity type name and press **Ctrl+D** on your keyboard, or select **Default Dataform** from the **View**

menu, or click the **Dataform** button  on the toolbar. Sentences displays the Dataform shown in Figure 14.

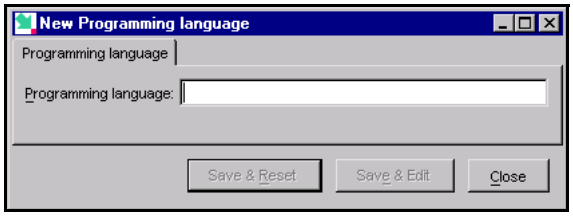

Figure 14 **An empty Dataform for Programming language**

This Dataform only contains one data field, for the name of the new entity instance. When you add association types that have this entity type as their source, data fields for those associations are also be shown on the Dataform.

Type in the name of the first programming language you are going to use, which is C++. Notice that after you start typing the **Save & Reset** and **Save & Edit** buttons become active, and the **Close** button has been replaced by the **Cancel** button.

Figure 15 **Creating a programming language entity**

Click **Save & Reset**, which saves the data you have entered and displays an empty Dataform. Now create some more programming language entities: XML, Java, and Visual Basic. Click **Save & Reset** after each one. After the last language name, click **Close**.

The Programming languages are displayed in the Sentences Explorer data pane, on the right-hand side of your screen.

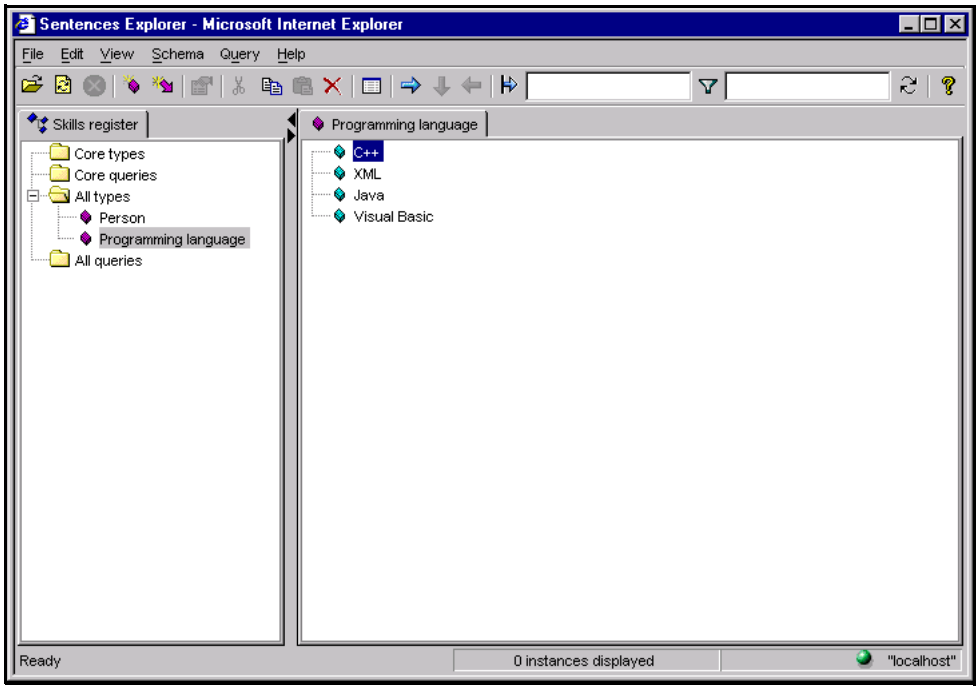

Figure 16 **Programming language entities displayed in the data pane**

Creating an association type

Before you add more data it is time to create an association type. An association type has three parts: a source, a verb and a target. Sources and targets for association types can be either entity types or other association types. The verb of an association type describes the link between the source and target.

In the first association type you are going to create, the source and target are both entity types. This association type is Person, *skilled in*, Programming language.

There are three ways to add a new association type:

- Select **Create Association Type** from the **Schema menu**
- Click the **Create Association Type** button (the magenta arrow) on the Sentences Explorer toolbar
- Select **Create Association Type** from the shortcut menu. To display the shortcut menu either right-click the **All types folder** in the Sentences Explorer, or right-

click the specific entity type that is going to be the source for the association type.

The following example illustrates using the shortcut menu from the source entity type. All the subsequent steps in this procedure are the same, whichever way you start.

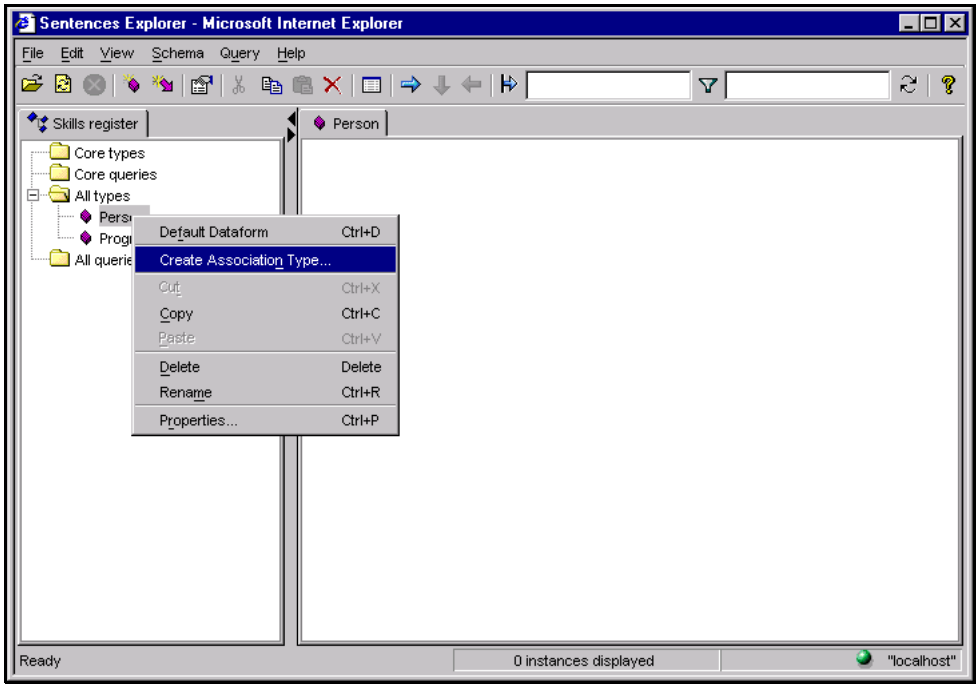

Figure 17 **Entity type shortcut menu**

1. Right-click the source entity type, Person, and select **Create Association Type** from the shortcut menu. Sentences displays the **Create Association Type** dialog.

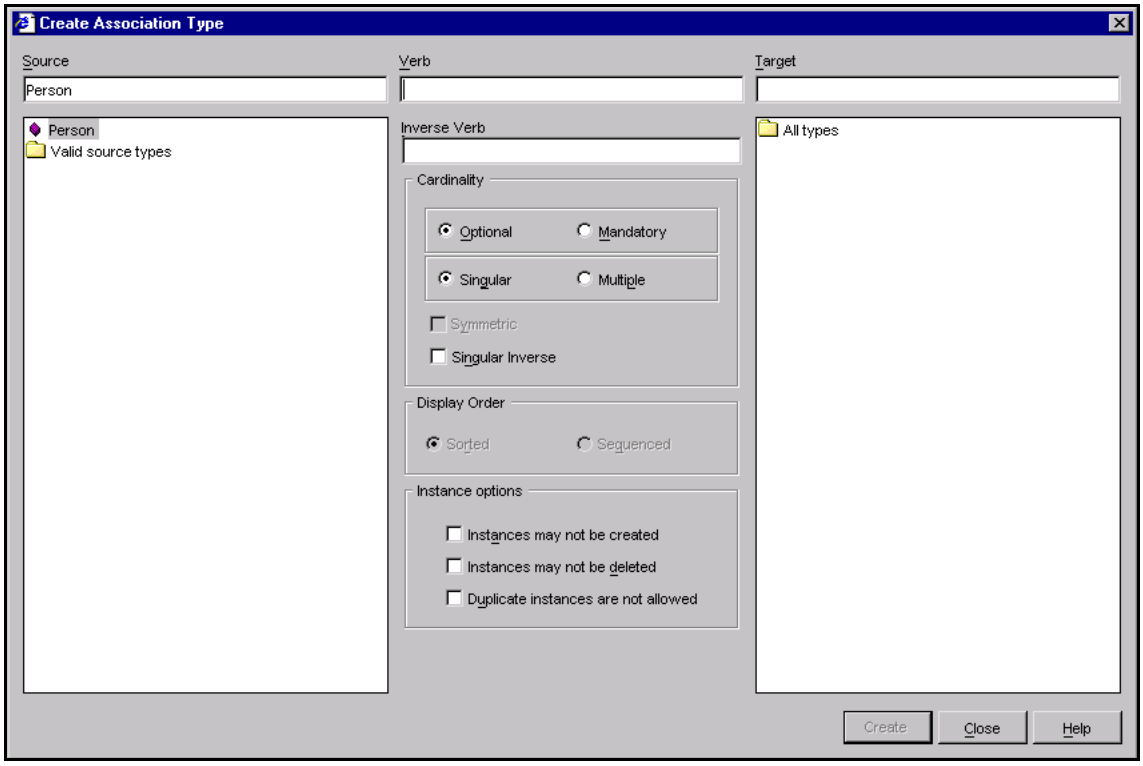

Figure 18 **The Create Association Type dialog**

If you right-click on a specific entity type to launch the **Create Association Type** dialog, then that entity type appears in the **Source** text box when the dialog box is opened. Otherwise, you can double-click on the **Valid source types** folder in the left-hand display pane to select the source entity type. The entity type appears in the **Source** text box.

2. Double-click on the **All types** folder in the right-hand display pane and select the target entity type. You should select Programming language. The entity type appears in the **Target** text box.

Typing a new name in either the **Source** text box or the **Target** text box of the **Create Association Type** dialog activates the **Create** button on the dialog which allows you to create a new entity type. When you click the **Create** button Sentences prompts you to confirm that you want to create a new entity type with that name.

The verb in an association type describes the association between the source and the target. Remember that the verb can be any linking phrase you choose. A longer

phrase is often more useful than a short phrase, especially when your application is going to contain many different association types. For example, to create an association type between the entity type car and the entity type colour, the verb phrase *is colour* is more useful then the verb phrase *is*, even though that is all that the English language needs to express the association.

The associative principle underlying the Sentences application is bi-directional. This means that Sentences recognises the association between the target and the source, as well as between the source and the target. Therefore you need to specify an inverse verb as well as a verb.

The inverse verb is the same link as the verb, but expressed from the target to the source. For example, if the verb of an association type is is colour then the inverse verb might be colour of. Sentences always suggests a default inverse verb based on the verb you entered, usually by adding the word "of". You can always type in your own inverse verb instead of the one suggested by Sentences.

3. Define the verb for the association type in the **Verb** text box. In this example, type the verb phrase skilled in.
4. Define the inverse verb in the **Inverse Verb** text box. In this example type the inverse verb phrase skill of.

The **Create Association Type** dialog allows you to define a number of attributes for the association type.

Note *This tutorial does not describe all the attributes available on the* **Create Association Type** *dialog. For information on* **Instance options**, **Singular inverse**, **Symmetric**, **Display Order (Sorted** *and* **Sequenced)**, *and the* **Metatypes** *folder, please refer to the Sentences User's Guide.*

Use the radio buttons in the **Cardinality** section of the dialog to define how many association instances may exist for each association source. If an association type is **Optional** it does not have to exist, so there can be zero or more association instances of that type. If an association type is **Singular**, then there may only be one, and not more than one association instance of that type.

You can use a combination of the **Cardinality** settings to determine whether an association type may have zero, one, or more than one association instance. In our example, a member of staff may not have any programming language skills, or may know only one programming language, or may know two or more programming languages. This means that this association type must be **Optional** and **Multiple**.

5. Select the **Optional** button (the default) to allow for a zero entry, and select the **Multiple** button to allow for more than one entry.

The completed **Association Type** dialog is shown in Figure 19.

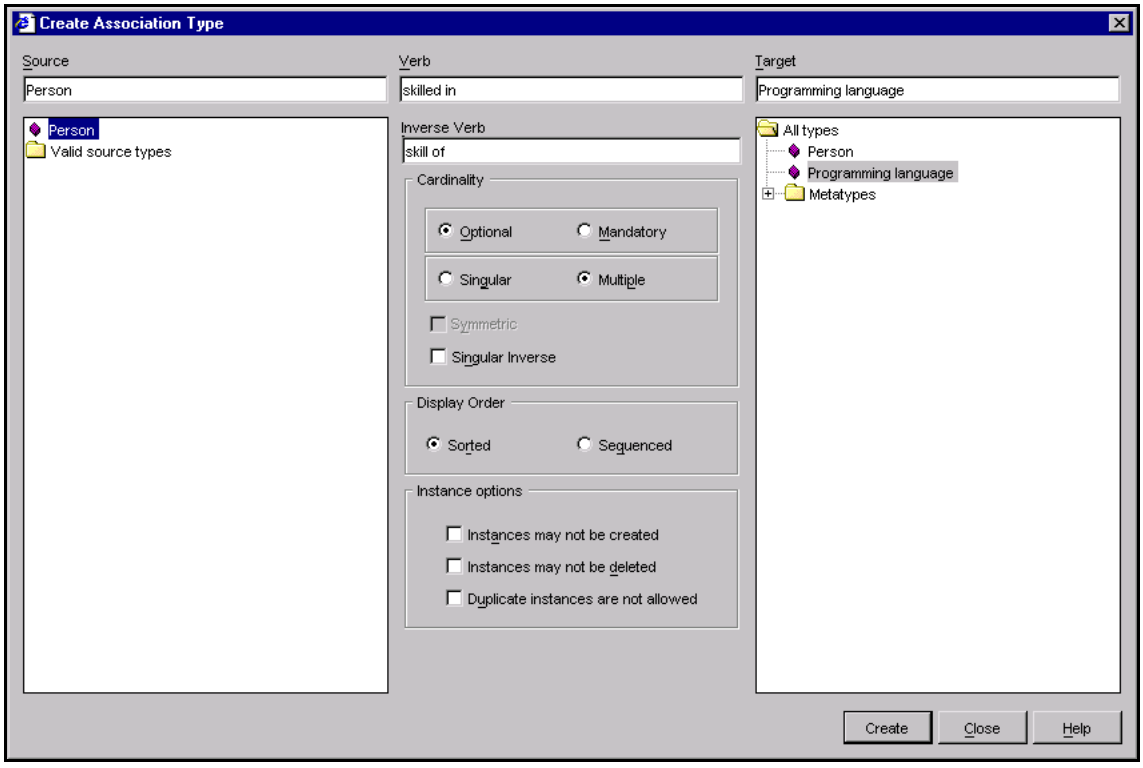

Figure 19 **Completed Create Association Type dialog**

Click **Create** to complete this association type, and click **Close** to return to the Sentences Explorer.

In the Sentences Explorer the new association type you have created is now nested below the entity type Person. Click the expand button next to the `Person` entity type to display the association, as shown in Figure 20. The association type is indicated by a magenta arrow that points downwards.

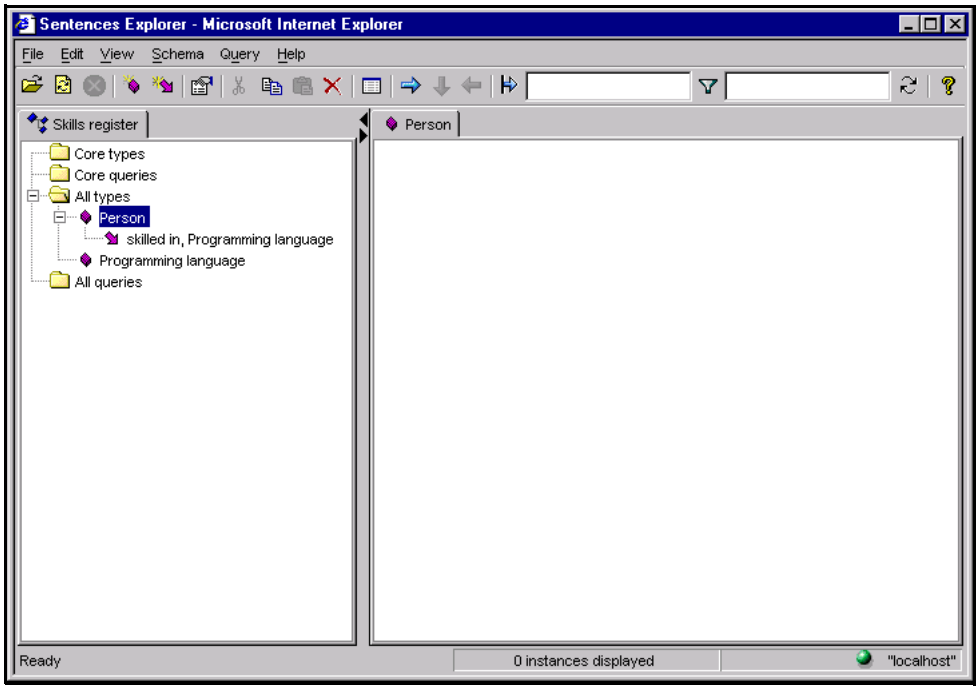

Figure 20 **Association type with source Person**

The expand button in the Sentences Explorer data pane indicates that an entity type is the source of an association type. The button is not displayed next to an entity type that is only the target of an association type. However you can double-click an entity type name and display any related inverse association types, that is association types for which it is the target, as shown in Figure 21. The inverse association type is indicated by a magenta arrow that points upwards.

Figure 21 also shows the list of programming languages you created earlier. They are displayed in the right-hand data pane because you have selected Programming language in the left-hand schema pane.

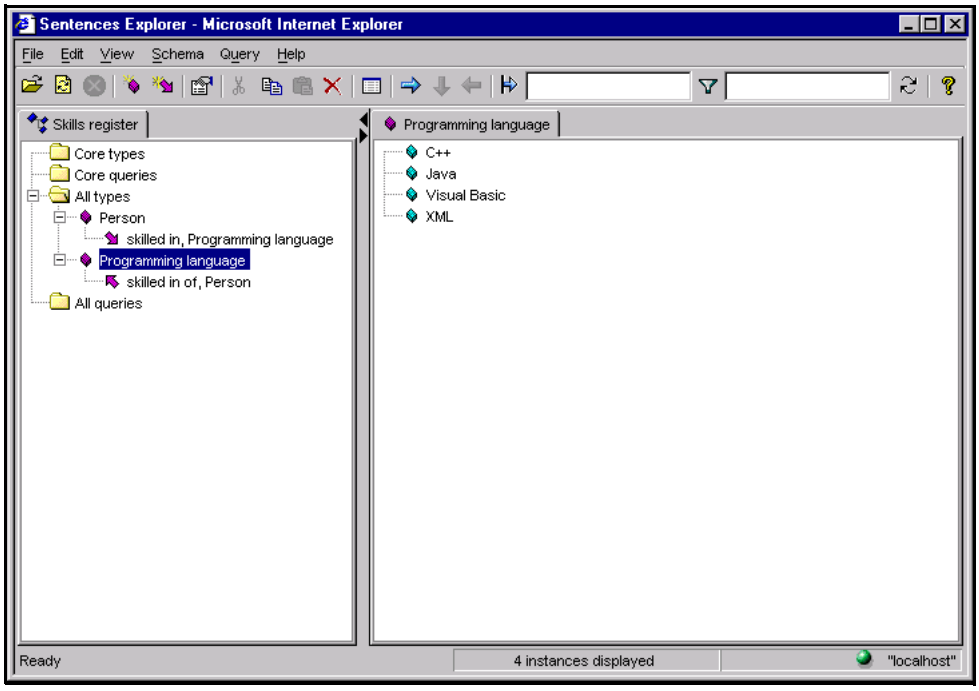

Figure 21 **Inverse association type with target Person**

Adding more entity data

The example application already contains names of programming languages, and an association linking Programming language to Person. Now you can add information about the programmers and their skills, and you can do this in one step.

Right-click the entity type name Person and select **Default Dataform** from the shortcut menu. You could also highlight the entity type name and either select **Dataform** from the **View** menu, or click the **Dataform** button on the toolbar. Sentences displays the Dataform shown in Figure 22.

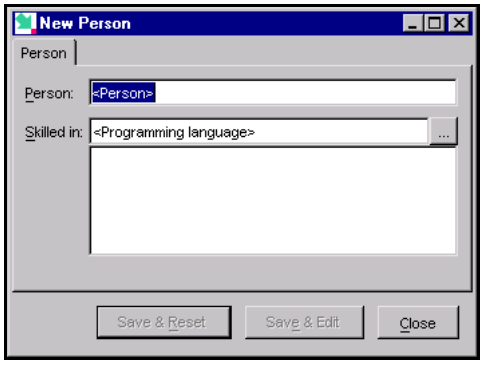

Figure 22 **Dataform for creating a new Person entity**

This Dataform has two fields. The first field represent the source entity, and the second field represents the target for the association you have just created.

The first field of a Dataform always has the name of the source type as its label, and here the label is **Person**. Data fields for association targets are always labelled with the association verb, and here the label is **Skilled in**. The association field shows space for more than one entry because you defined the Person, *skilled in*, Programming language association type as **Multiple.** The following steps show you how to enter data about the first of our programmers and his programming skills.

1. Enter the name of our first programmer, Bill North, in the **Person** field.

2. Next to the **Skilled in** field there is an ellipsis button  . Click the ellipsis button to display a picker list.

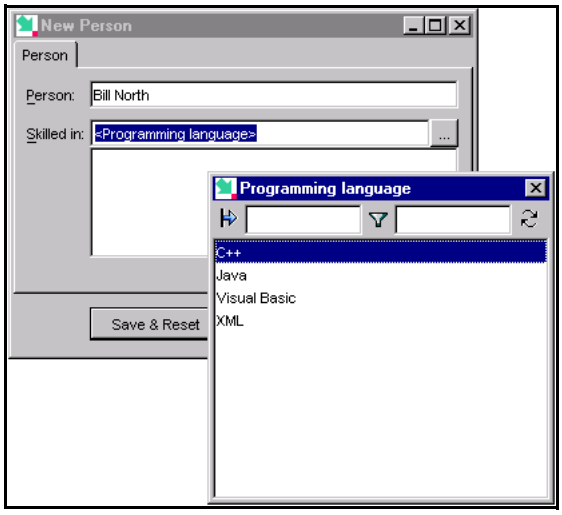

Figure 23 **Dataform with picker list**

3. Click C++ in the picker list to select this programming language as one of the skills of this programmer.

4. Click the ellipsis button again, and select Java. Repeat this action and select XML. Click **Save &Edit** (or **Save & Reset**) to store all the data you have entered and then click **Close** to return to the Sentences Explorer.

In the Sentences Explorer data pane there is now an expand button next to Bill North. Click this expand button to reveal the folder of associations *skilled in*, Programming language, which also has an expand button. Click this button to display the list of associations you have just created.

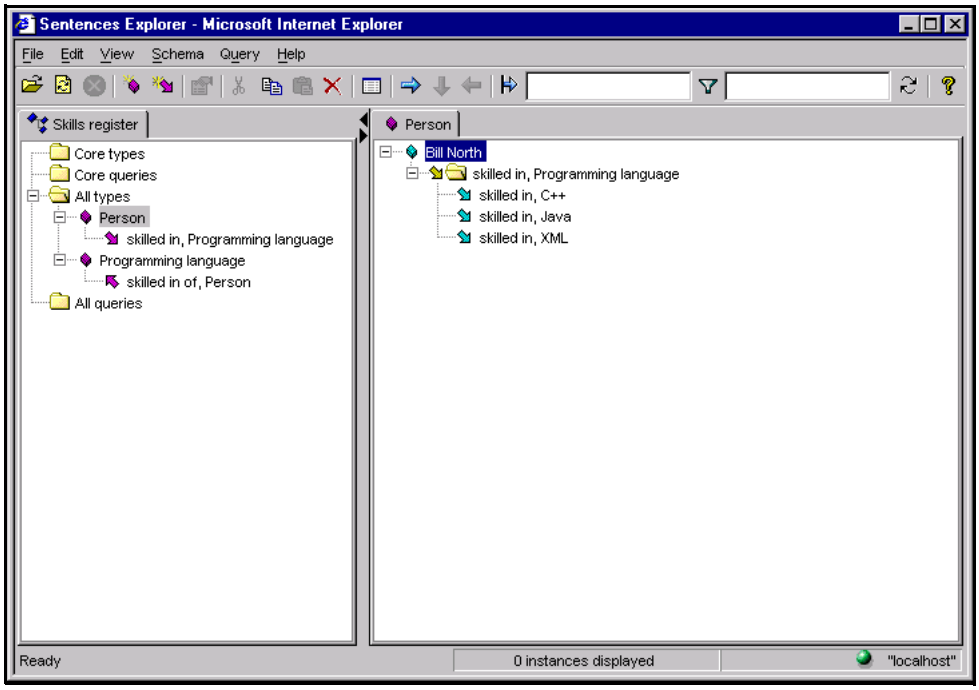

Figure 24 **Bill North's programming languages**

Now it's your turn to enter information about three more programmers and the programming languages they know. You need to repeat the steps 1 to 4 above for programmers named James South, Louise East, and Sue West.

Their programming skills are listed below.

Person	Programming languages
Bill North	C++, Java, XML
James South	C++, XML
Louise East	Java, Visual Basic
Sue West	Visual Basic, XML

By allowing you to select the programming languages from the picker list Sentences is making sure you don't enter data more times than is necessary. In the associative model a data item can be used many times even though it is stored only once. In our

example the data instance XML is stored once, even though you use three associations to indicate that three programmers know this language. All three associations make use of the same target.

After you have entered the names and skills of all four programmers, highlight the Person entity type in the schema pane. The list of programmers is displayed in the data pane, as shown in Figure 25. Each name has an expand button, as you have just defined associations for each of these entity instances.

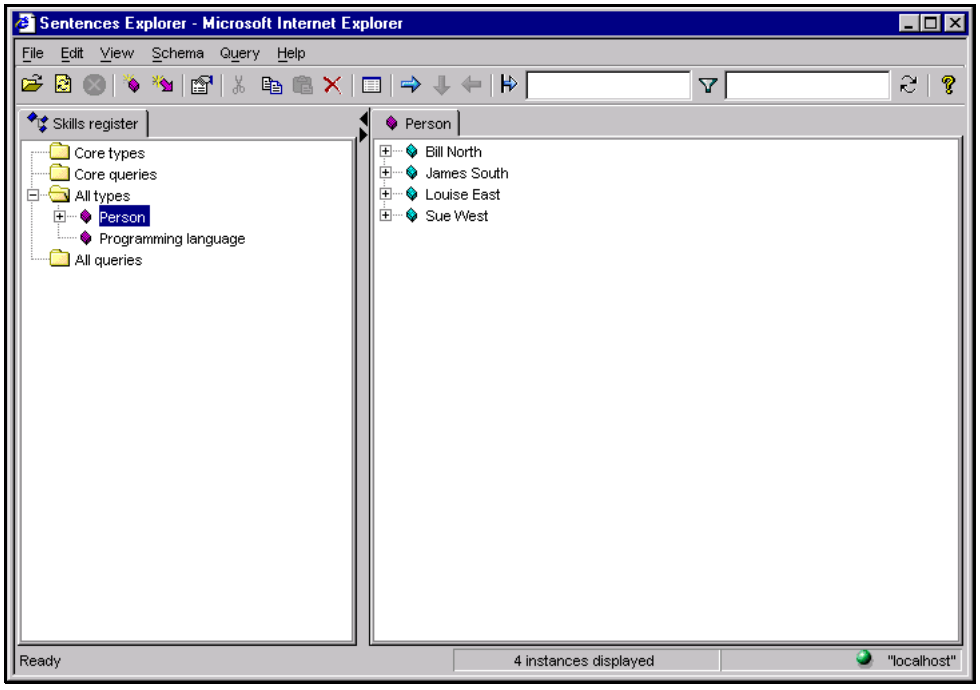

Figure 25 **All defined programmers**

Highlight the association type *skilled in*, Programming language in the schema pane. The list of all the associations of this type are displayed in the data pane, as shown in Figure 26.

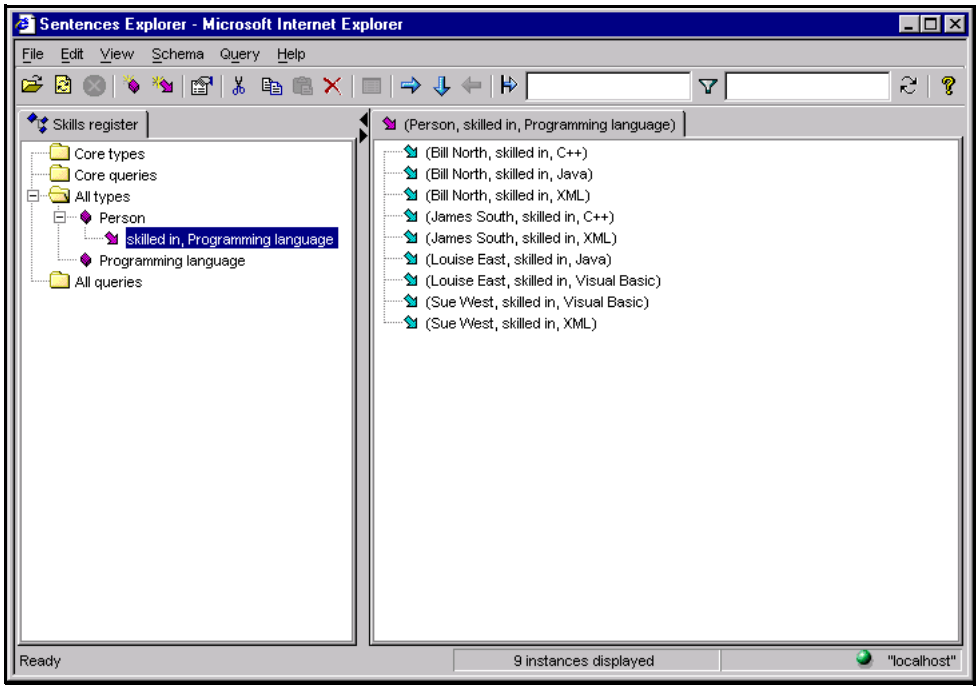

Figure 26 **All the skills of all the programmers**

In the next section of the tutorial, you use the *skilled in,* Programming language association type as the source of another association type (see “Using an association type as a source” on page 36). Before you do that you can look at the way that Sentences allows you to use the associations you have just created as a starting point for investigating your application.

You can move from an association to its source or its target, and then move on from the source or target as you wish. The example application is of course still very small and straightforward, but the method shown here can be used at any level even in complex Sentences applications.

To see more information about the source and target of an association, do the following:

1. Highlight an association in the data pane (for example James South, *skilled in*, XML)
2. Select **Show source & target** from the shortcut menu (or from the **View** menu)

Sentences displays the source and target for the association, as shown in Figure 27. These are cube-shaped like entity icons and the source icon has a bar projecting to the right, and the target has a bar projecting to the left.

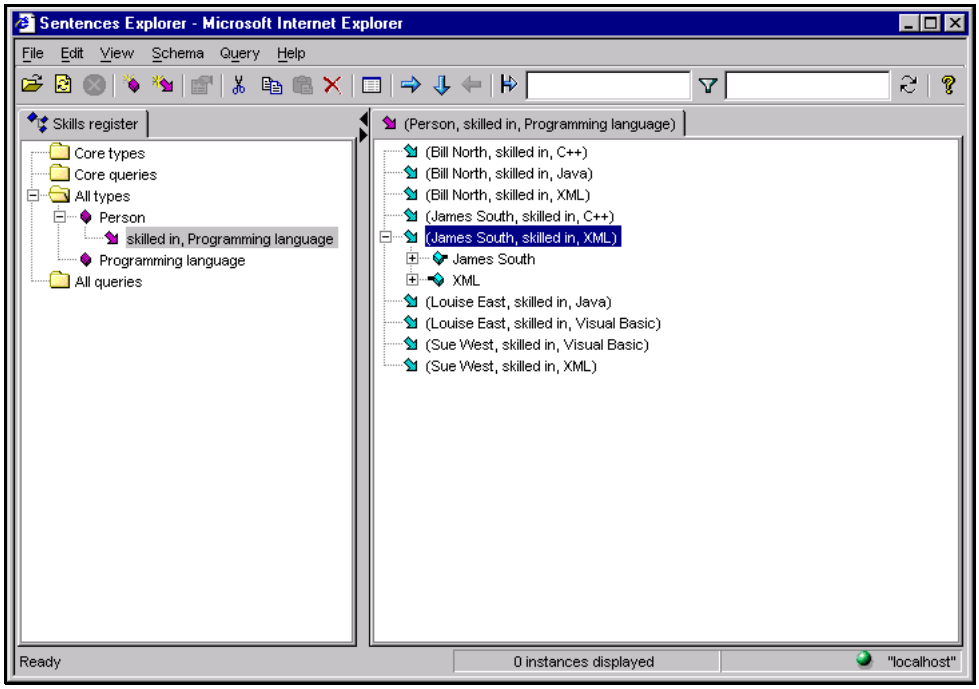

Figure 27 **Source and target of an association**

3. Click the expand button next to the source, and then click the expand button next to the multiple associations folder to display all the programming skills of James South.

4. Click the expand button next to the target, and then click the expand button next to the multiple associations folder to display a list of all the programmers skilled in XML. This is a list of *inverse* associations, showing the relationship between a target (a programming language) and a source (a person) and so this list shows the *inverse* form of the association verb (skill of).

The result of these expand actions is shown in Figure 28.

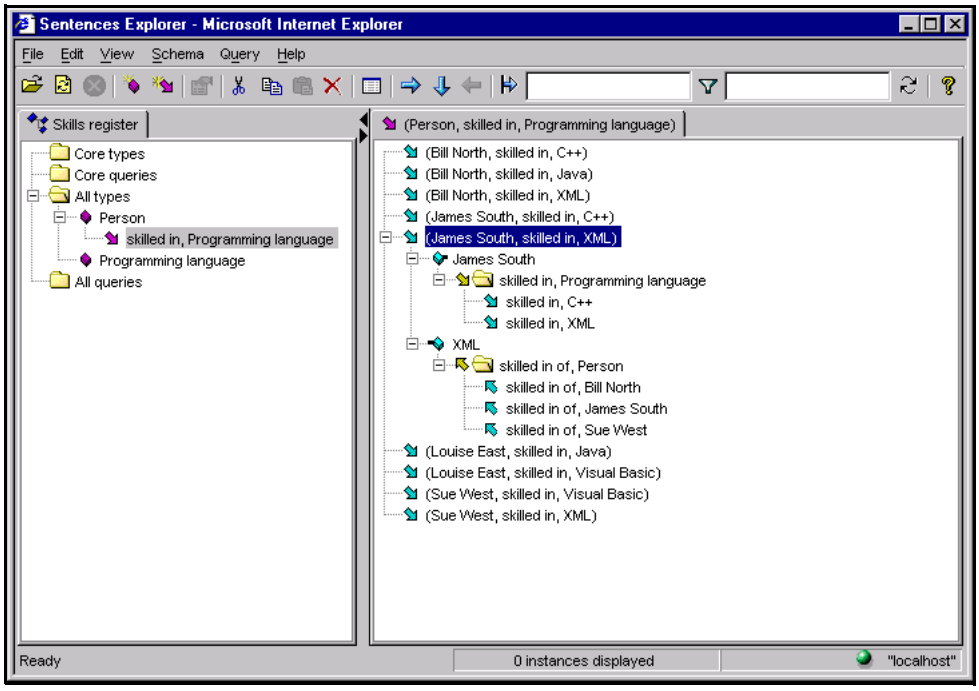

Figure 28 **Expanded associations from source and target**

Using an association type as a source

In the associative model of data, an association type may have another association type as its source, or as its target, or indeed both its source and its target may be association types.

The next step of this example uses an existing association type as the source of a new association type.

Note *This tutorial does not illustrate using an association type as the target of an association type. For more information see the Sentences User's Guide.*

In the Sentences Explorer schema pane (the left-hand pane), highlight the association type Person, *skilled in*, Programming language. The data pane (the right-hand pane) displays a list of all the skills of all the programmers, as shown above in Figure 26.

You can now enhance this information by adding data about the skill level of each programmer in each language, so that you can distinguish between a novice programmer and an expert programmer.

This section of the tutorial shows you how to create a new entity type called Level, and populate it with three instances, Novice, Competent and Expert, and then create a new association type which links this new information to data you already have. The existing association type Person, *skilled in*, Programming language is the source, and Level is the target of this new association.

When it is necessary to write out Sentences association types you should use commas to separate the source, verb, and target elements, and use parentheses around all three elements to group associations. Nested parentheses are used to show nested associations. The previous association type example was (Person, *skilled in*, Programming Language). The new association type uses this as its source element, and can be written as:
((Person, *skilled in*, Programming Language), *at level*, Level).

Now follow these steps:

1. Right-click the **All types** folder and select **Create Entity Type** from the shortcut menu.
2. Create an entity type called Level.
3. Right-click the new entity type, and select **Default Dataform** from the shortcut menu.
4. Create three entities named Novice, Competent, and Expert. You must click **Save & Reset** after typing in the name for each entity and click **Close** after the last one.

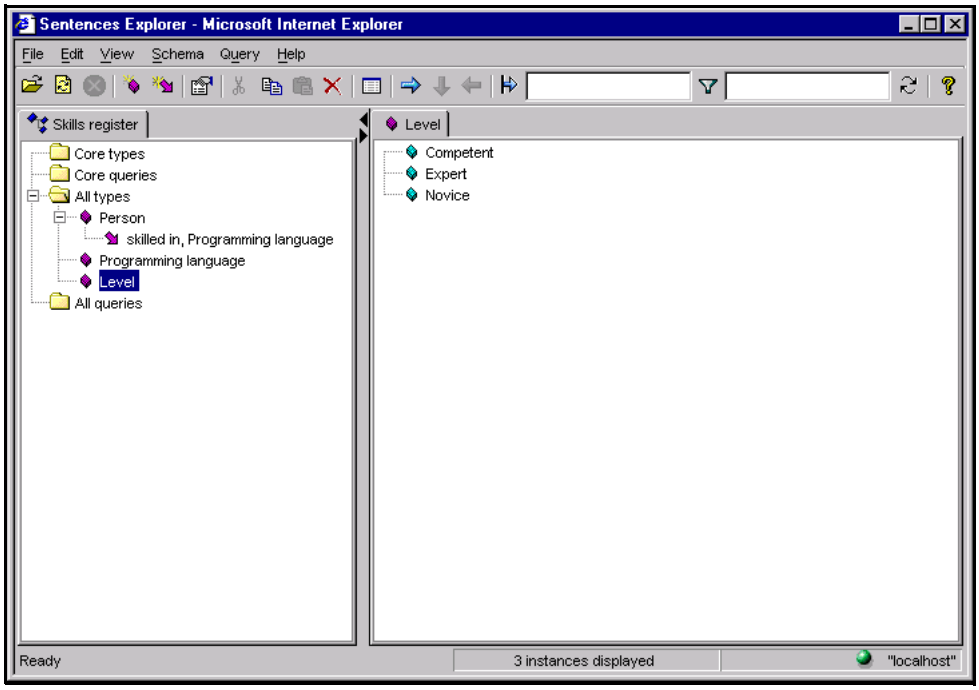

Figure 29 **Instances for the Levels Entity Type**

Your Sentences Explorer should now look like Figure 29.

5. Right click the association type (Person, *skilled in*, Programming language) and select **Create Association Type** from the shortcut menu.
6. In the **Association Type** dialog, enter the verb at level, and the inverse verb level of.
7. Select the entity type Level as the target.
8. Select the **Optional** radio button (the default) and the **Singular** radio button (also the default).

Selecting **Optional** means that you can choose whether or not to specify a level for each skill of each programmer. Selecting **Singular** means that if you do choose to specify a level you can only select one level value for each association.

The **Create Association Type** dialog should look like Figure 30.

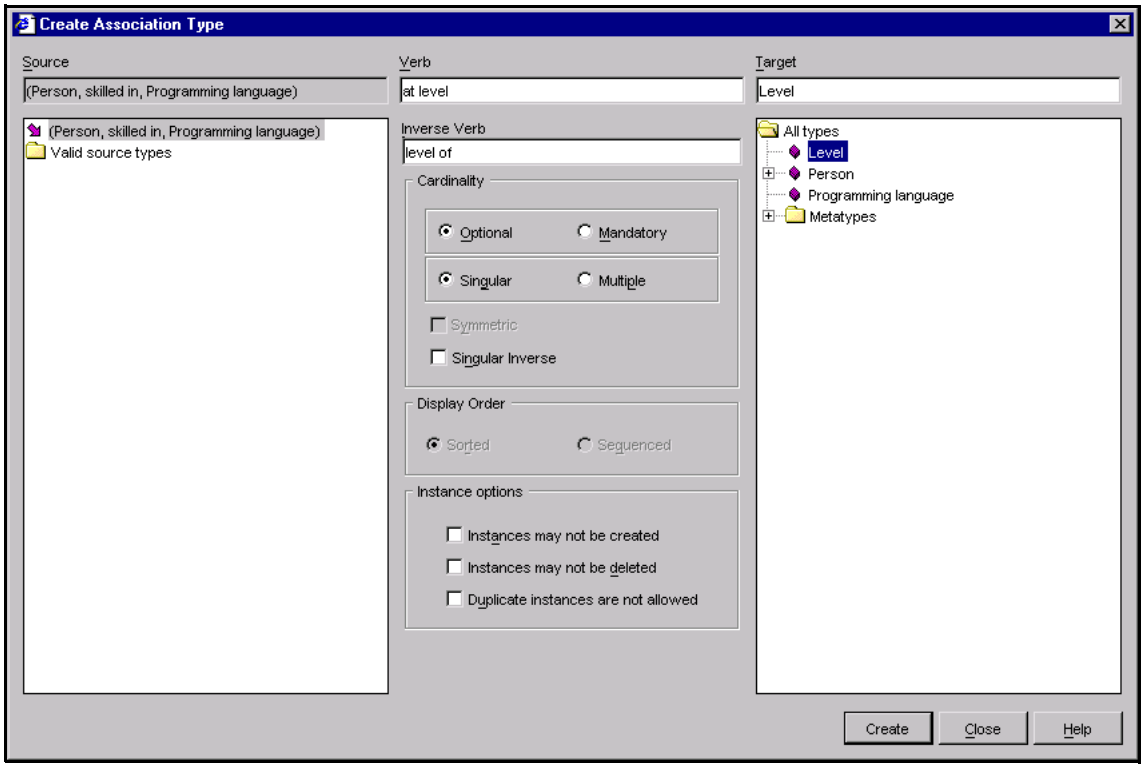

Figure 30 **The Create Association Type dialog with an association type as source]**

9. Click **Create** and then click **Close** to return to the Sentences Explorer
10. In the schema pane, select the association type Person, *skilled in*, Programming Language. (If you click the expand button (expand sign) you can see the new association type *at level*, Level.)
11. In the data pane right-click the first existing association (Bill North, *skilled in*, C++) and select **View association** from the shortcut menu.

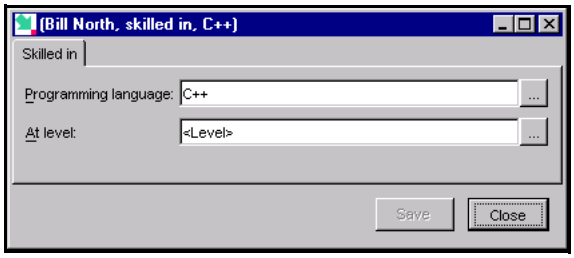

Figure 31 **Dataform display on an existing association**

12. In the Dataform, click the ellipsis button next to the *At level* field to display a picker list and select Competent, and click **Save**, then **Close**. Double-click the association in the data pane or click the expand button next to it. Sentences displays a data entry for the *at level*, Level association type.

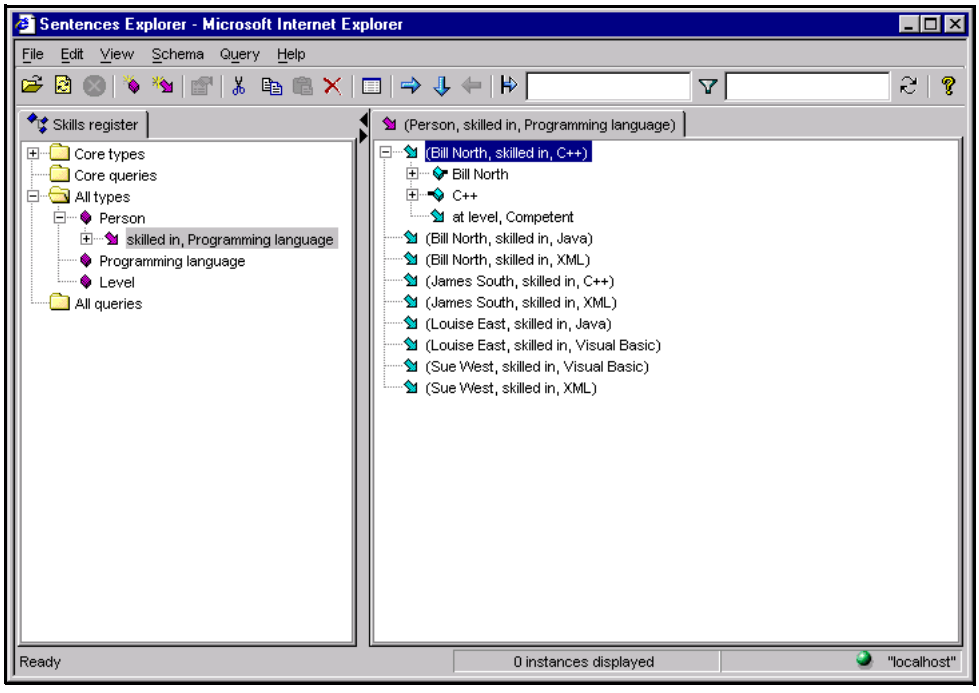

Figure 32 **Sentences Explorer data pane showing a nested association**

Repeat the steps above to add skill levels for all the existing instances of the Person, *skilled in*, Programming language association. Highlight an association of this type

and select Dataform from the shortcut menu. Use the list below to set the values of the *at Level* associations.

Person	Programming language	Level
Bill North	C++	Competent
Bill North	Java	Novice
Bill North	XML	Expert
James South	C++	Competent
James South	XML	Expert
Louise East	Java	Competent
Louise East	Visual Basic	Novice
Sue West	Visual Basic	Expert
Sue West	XML	Novice

When you highlight the *at level*, Level association type in the schema pane, the completed list of skills and levels associations should look like Figure 33.

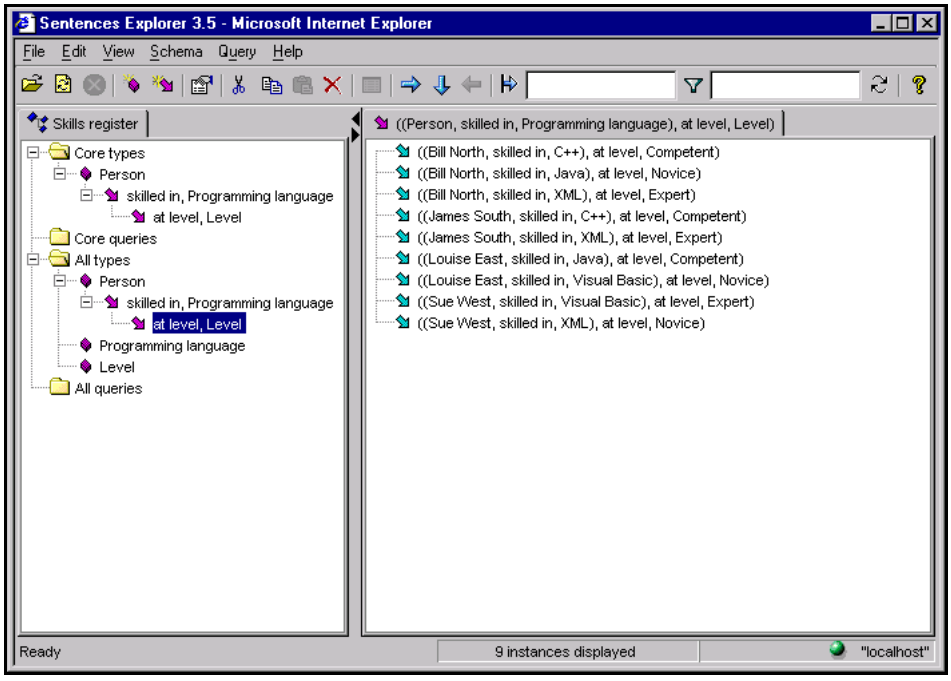

Figure 33 **Sentences Explorer data pane showing list of nested associations**

The view of the Sentences Explorer shown in Figure 33 is interesting for two reasons:

- The label at the top of the current tab in the data pane which shows the source, verb, and target of the current association type, shows a nested association type.
- The **Core types** folder is open and contains an exact copy of the Person entity type and all the association types linked to it.

  Earlier in this tutorial you selected the **Generate Core types automatically** option in the **Edit profile** dialog (see Figure 7 on page 17). When Sentences selects entity types for the **Core types** folder automatically, the folder displays all the entity types that are the source of an association type that is itself the source of other association types. In other words, an entity type that is the original source of a nested association type is displayed in the **Core types** folder as well as in the **All types** folder. The Person entity type now meets the criteria for automatic inclusion in the **Core types** folder, and so it is displayed there.

If you had not selected the **Generate Core types automatically** option you could choose which entity types are displayed in the **Core types** folder. For more information about this option, see the *Sentences User's Guide*.

Creating a Mandatory and Singular association type

If you create an association type with the **Cardinality** attributes **Mandatory** and **Singular**, it means that an association of this type must exist for each entity of this type, and that there can be no more than one association of this type.

Carry out the following steps to add an example of a **Mandatory** and **Singular** association type to the application:

1. Right-click the **All types** folder and select **Create Entity Type** from the shortcut menu.
2. Create an entity type called Phone extension.
3. Right-click the new entity type, and select **Default Dataform** from the shortcut menu.
4. Create four instances named 402, 404, 406, and 408. You must click **Save & Reset** after each entity and click **Close** after the last one.
5. Highlight the Person entity type, and select **Create Association Type** from the shortcut menu.

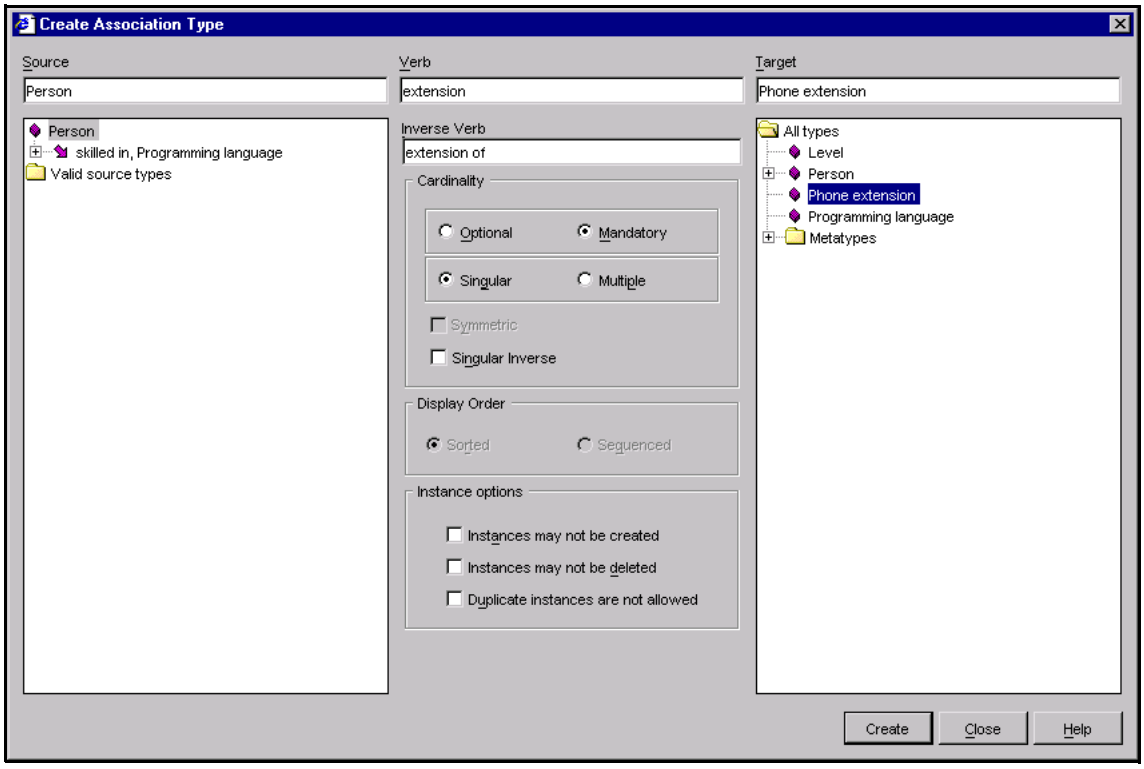

Figure 34 **The Create Association dialog for Phone extension**

6. Create a new association type (Person, *extension,* Phone extension). Use the phrase *extension of* as the inverse verb. Make sure you select the **Mandatory** and the **Singular** radio buttons in the dialog.

7. Click the Person entity type in the schema pane, and then right-click the name Louise East the data pane, and select **Default Dataform** from the shortcut menu.

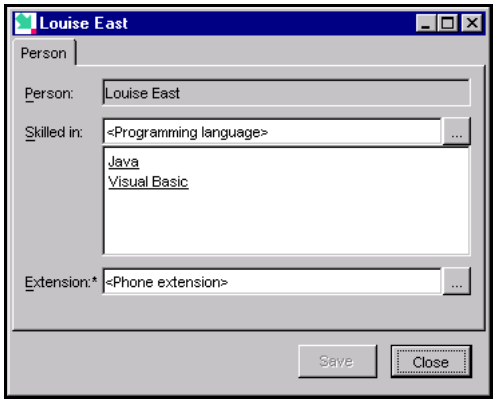

Figure 35 **Dataform with Phone extension association**

The Dataform now shows the new Phone extension association, as shown in Figure 35. You can add a target for this association in the same way as you added targets for Programming Language or Level. This association is different from the previous ones because you defined it as **Mandatory**. You can see that is marked with an asterisk in the Dataform.

If you open a Dataform to create new instances and their associations, and one or more of those associations is **Mandatory**, Sentences does not allow you to save the Dataform unless you specify a target for each of these associations. For more information on this topic, please see the description of Dataforms and their use in the *Sentences User's Guide*.

You must now enter the phone extension associations for each of your programmers. Select each of the programmers in turn and open the Dataform to add associations to a phone extension for each of them. Create the associations shown in the list below.

Person	Phone extension
Bill North	402
James South	404
Louise East	408
Sue West	406

If you take another look at the Dataform shown in Figure 35, you can see that each of the programming languages has been underlined. This is the way that Sentences

indicates that the target of an association is also the source of another association. In this case, each of the associations of the type *skilled in*, Programming language is the source of an association of the type *at level*, Level.

Defining entity Properties and using datatypes

You are now going to add information about ongoing projects to the Skills register application. Each project must have a name, a start date, and an end date.

Try and create three projects with the names Capricorn, Pisces, and Taurus, by yourself, before you read the instructions below. Then read the instructions to see if you were right:

1. Right-click the **All types** folder and select **Create Entity Type** from the shortcut menu.
2. Create an entity type called Project.
3. Right-click the new entity type, and select **Default Dataform** from the shortcut menu.
4. Create three instances of the type Project named Taurus, Capricorn, and Pisces. Remember to click **Save & Reset** after each entity and click **Close** after the last one.

The next step involves dealing with date entries. Sentences uses datatypes to handle a range of known data formats, including dates, numbers, percentages and hyperlinks. By assigning a datatype to an entity type you make sure that Sentences recognises its entity instances as, for example, numbers or dates or hyperlinks and deals with them correctly.

Note *The way dates and times in Sentences are displayed on your system depends on the country and language settings (the locale settings) of your operating system. This tutorial uses the day-month-year format used in the UK locale rather than the month-day-year format used in the US locale.*
In Windows you can check the current date style settings on your computer by selecting **Settings/Control Panel** *from the* **Start** *menu, and looking at the* **Date** *tab on the* **Regional Settings** *dialog. (The exact wording for checking the date settings varies between different versions of Microsoft Windows.)*

Assigning a datatype automatically turns an ordinary entity type into a value type. The behaviour of value types is slightly different from that of entity types. For example, a value type cannot be used as the source of an association type. Value

types and instances are shown in the Sentences Explorer with a disc-shaped icon. For more information about value types please see the *Sentences User's Guide*.

You can use the **Properties** dialog to set the datatype for an entity type. In addition, if you create an entity type which has the same name as one of Sentences' predefined datatypes, the entity type is defined with that datatype automatically.

The example application needs to hold information about the start and end dates of projects. To do this you can create an entity type named Date and use it as the target of association types defining start and end dates. Each individual date, for example 14th March 1999 or 23rd October 2003, is an instance of this entity type.

The name Date is the same as the name of one the predefined datatypes and so Sentences automatically makes it into a value type with the **Date** datatype.

More information about all of the options available in the **Properties** dialog, and about datatypes, is given in the *Sentences User's Guide*. It is also possible for programmers to create their own custom datatypes.

To continue building the example application carry out the following steps:

1. Create an entity type named Date.
2. Right-click the Date entity type, and select **Properties** from the shortcut menu.

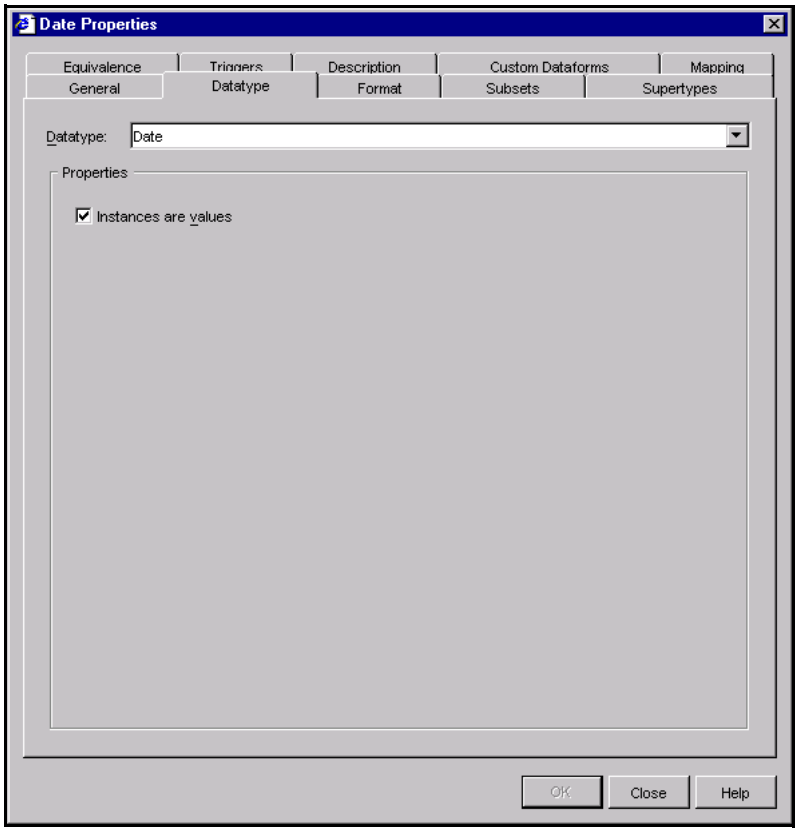

Figure 36 **The Datatype tab in the Properties dialog**

3. In the **Properties** dialog select the **Datatype** tab. You can see that the Date datatype has already been selected. You can click the down arrow to review the list of predefined datatypes.

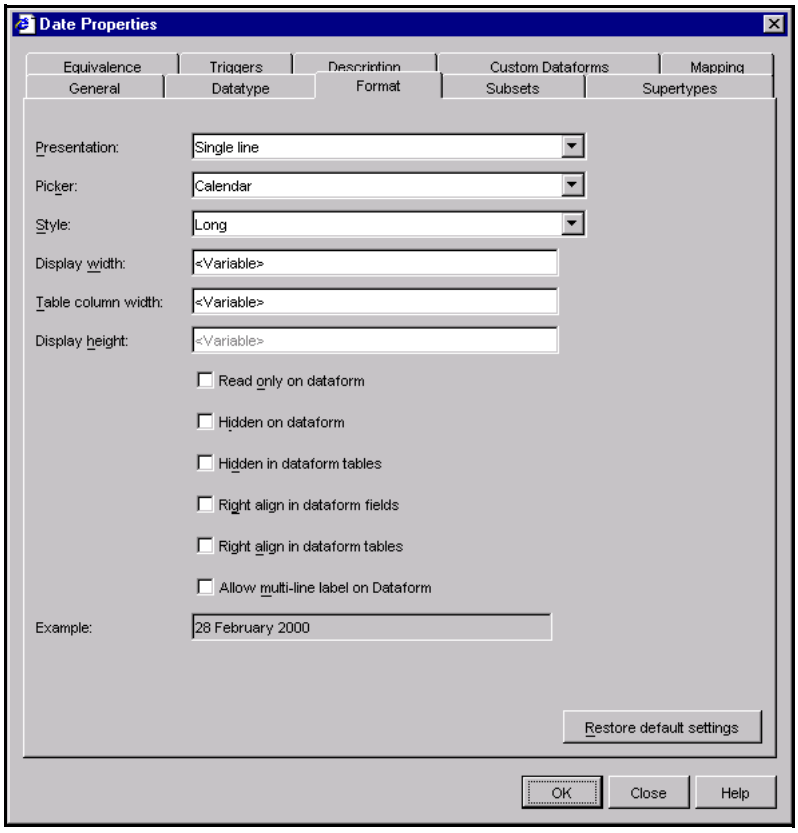

Figure 37 **The Format tab in the Properties dialog**

4. Select the **Format** tab. In the picker field, select the **Calendar** picker. This means that when you want to choose a date you can do so from a convenient calendar style picker.

5. In the **Style** field, select the **Long** date format. Now click **OK**, followed by **Close**.

The next step is to create two association types for the entity type Project which showing the start and end dates for each project. The verb of an association type always appears as the field label in the Dataform. You can take advantage of this property by using the verbs *start date*, and *end date* for these two new association types. Their full source, verb, target descriptions are (Project, *start date*, Date) and (Project, *end date*, Date). You can accept the default suggestions for the inverse verbs of these association types. You should define both of these association types

as **Mandatory** and **Singular**, as you did earlier for the Phone extension association type.

Select the first Project instance, Capricorn, and right-click. Select **Default Dataform** from the shortcut menu. You can now enter the start and end dates for your project directly. The instances you create now can be used later by other associations, and are available on a picker list of Date instances.

Even though you have selected the **Long** style to display dates, you can still use the short date format defined on your computer to enter the dates, for example **dd/MM/yy**. This means you can type in 31/12/99 or select the date from the Calendar picker, and Sentences displays 31 December 1999.

If you enter the year using two digits rather than four digits, Sentences assumes a century starting eighty years before the current date and ending 20 years after the current date.

For the project Capricorn, you should use the start date 15 August 2002, and the end date 31 December 2002.

The dates are displayed by Sentences in the **Style** you selected in the **Properties** dialog, as shown in Figure 38.

Figure 38 **Dataform with start and end dates**

Repeat this procedure for the two other projects. The list below shows the dates for all the three projects.

Project	start date	end date
Capricorn	15 August 2003	31 December 2003

Project	start date	end date
Pisces	25 June 2003	15 March 2004
Taurus	01 October 2003	10 May 2004

You can see all the dates for these projects in the Sentences Explorer view shown in Figure 39.

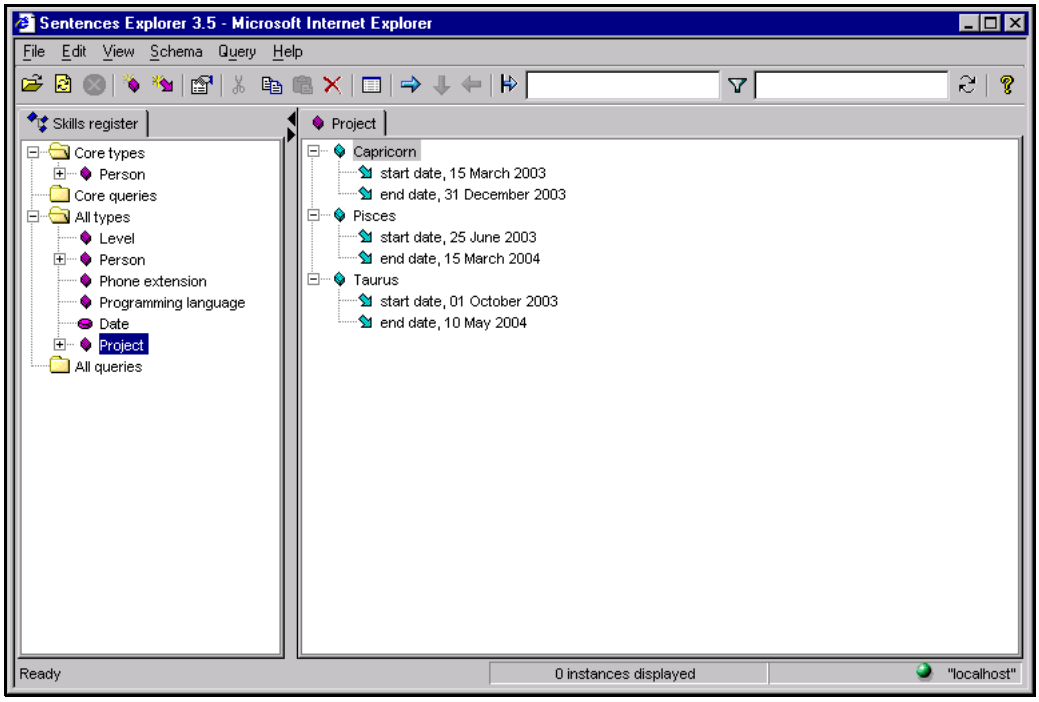

Figure 39 **Sentences Explorer data pane showing Project Start and End dates**

Adding more application features

Your original plan was to match programmers to their project assignments. Even though you have not yet completed the schema that can do that, you already have a functioning database.

If you have followed all the steps in this tutorial so far, you should feel confident enough to work by yourself a little. If you are still a bit uncertain about working

with Sentences you can take a break here and review what you have covered before continuing.

When you are ready, try and work from the following brief instructions to complete the link between Person and Projects:

Create an association type in the source, *verb*, target format Person, *assigned to*, Project. Use the phrase assignment of as the inverse verb. Make the association type **Optional** and **Singular** (the default options).

Assign the programmers to projects as follows:

Person	Project assignment
Bill	Capricorn
James	Taurus
Louise	Pisces
Sue	Taurus

The Sentences application now knows a lot about the programmers in your team. Right-click on the name of one of the programmers, Sue West, and select **Default Dataform** from the shortcut menu. The Dataform is shown in Figure 40.

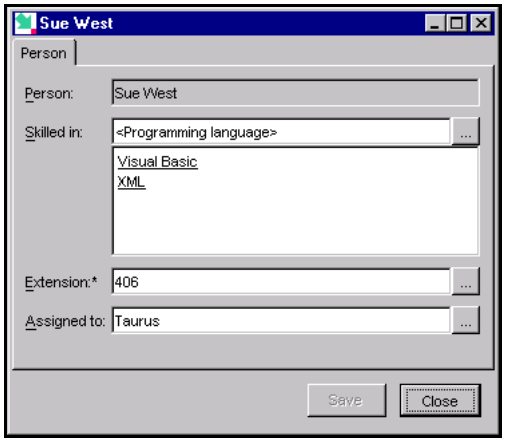

Figure 40 **Default Dataform showing the assigned to project association**

In the open Dataform right-click on Taurus, the name of the Project that Sue West is assigned to, and select **View "Taurus"…** from the shortcut menu. Sentences displays a second Dataform, showing the start and end dates of the project.

A Dataform opened from the Sentences Explorer, like the Dataform opened on Sue West, is known as a parent Dataform. Any Dataform opened from another Dataform, like the Dataform opened on Taurus, is known as a child Dataform and has an **Apply** button, rather than a **Save** button.

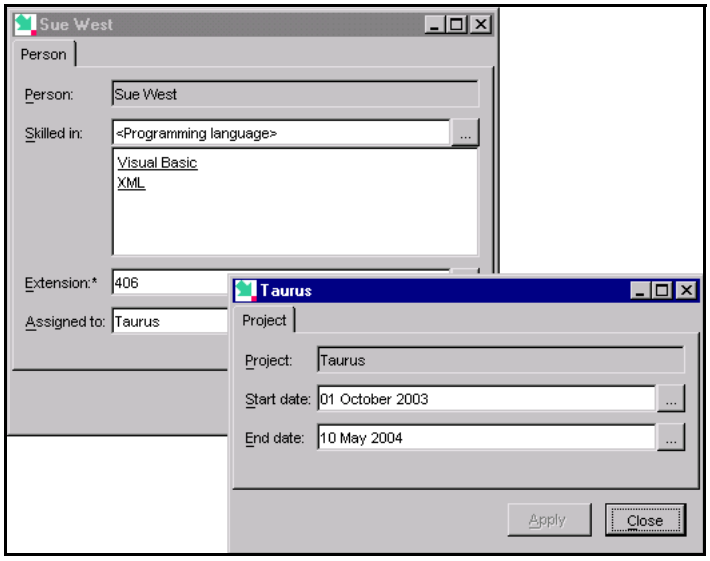

Figure 41 **Child Dataform on project Taurus**

Using images

You can add graphics and images to your Sentences application. Sentences supports the JPG and GIF image formats. You can now add an identity picture for each member of staff to our example application.

All the external files used by Sentences, including image files, are stored on a server, which is usually the Sentences server, and they must be accessed by referring to a URL (web address), which is different for every installation. The actual location of the image file directory is selected during Sentences installation, and is referenced by the web page used to start Sentences. The images used by this tutorial are located in that directory.

The Sentences administrator must also create a virtual directory on the web server for the images directory, and may also provide a base address for a relative URL for image file addresses, called the `ImageBaseURL`.

If there is a defined ImageBase URL on your system, you can refer to an image by typing in its file name only. If your system does not have an ImageBase URL defined you need to enter a full URL in the format:
`http://[web address]</SentencesData35/Images/><filename>`
where
`[web address]` is the address of your Sentences installation,
`</SentencesData35/Images/>` is the default directory structure, and
`<filename>` is an image file.

The default directory used for image files for this tutorial is named `LazyImages`, and the image themselves are named `Bill.jpg`, `James.jpg`, `Louise.jpg`, and `Sue.jpg`. If your system has been set up using the default installation for Sentences, then you can refer to an image with a relative address, such as:
`LazyImages\Bill.jpg`

You must check with your local system administrator or network manager for the correct Web address for your system.

Carry out the following steps to add images to the example application:

1. Create a new entity type called Photo.
2. Right-click Photo, and select **Properties** from the shortcut menu.
3. In the **Properties** dialog for Photo, select the **Datatype** tab and select **Image** from the dropdown list. Click **OK**, and then click **Close.** You can see that Sentences has defined Photo as a value type with a disc-shaped icon.

   If you do not define Photo with the **Image** datatype Sentences does not know how to handle the file reference.
4. Create a new association type, with the format (Person, *ID picture*, Photo). Select the **Optional** and **Singular** (default) radio buttons. Accept the default suggested for the inverse verb.
5. Right-click the name of Bill North, and select **Default Dataform** from the shortcut menu.

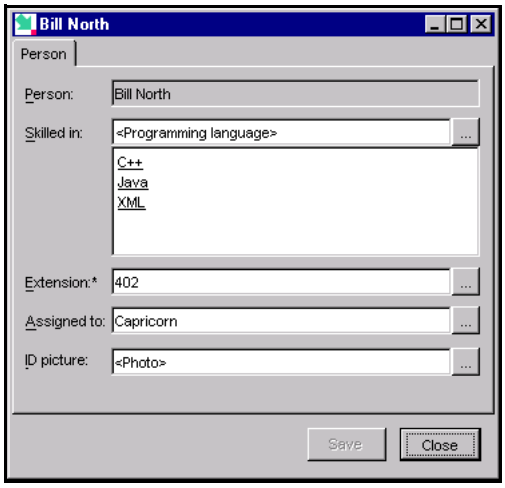

Figure 42 **Dataform showing the ID Picture association**

The Dataform now includes a control for the ID picture of each person, as shown in Figure 42.

6. In the **ID picture** field type in the web address for the image you want. If you are working with a default installation, type in:
   `LazyImages/Bill.jpg`

   Otherwise, use a Web address in the format:
   `http://<hostname:portname></SentencesData35/Images/>`
   `LazyImages/Bill.jpg`

Note *You may need to consult your local system administrator or network manager for details of the correct way to access Sentences images on your system*

Take care to enter the file name exactly. File names for images are case-sensitive because when you select an image file you are accessing a file on a Web server. If you have entered the file name correctly the image appears as soon as you click **Save** or move to another field. To change the image, click in the image area to display the **ID picture** field, and type in the details of a new image file.

The picture is now displayed on the Dataform, as shown in Figure 43.

Click **Save**, and then click **Close**.

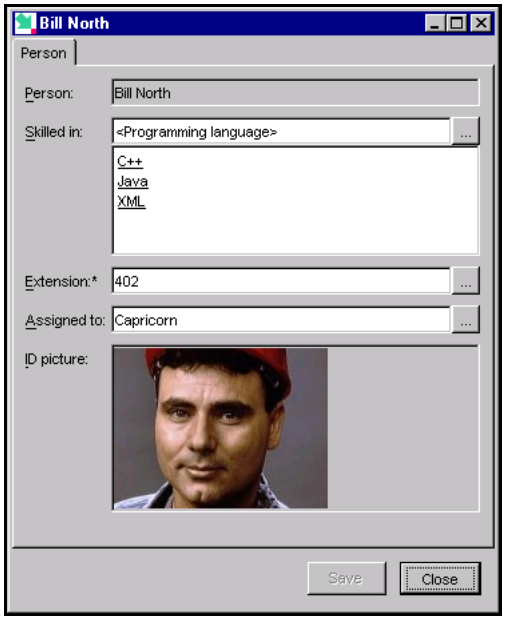

Figure 43 **Dataform showing the ID picture**

When you typed in the path for your image file, you created an instance of the entity type Photo, and you created an association between the Person instance Bill and the Photo instance `Bill.jpg`. This information is saved in your application, in the same way as any other association. You can now open the Dataform for each of the other three programmers and add their ID pictures in the same way.

The file names of all the ID pictures are:

Person	ID picture file name
Bill	`LazyImages/Bill.jpg`
James	`LazyImages/James.jpg`
Louise	`LazyImages/Louise.jpg`
Sue	`LazyImages/Sue.jpg`

Viewing your data in the Sentences Explorer

Everything in your Sentences database can be viewed using the Sentences Explorer. The Sentences Explorer allows you to expand or collapse each element to display or hide associations and their associated entities. This applies to both the schema pane and the data pane.

The phrase “drilling down” describes the process of clicking the expand sign adjacent to an element and then clicking another expand sign beneath. In the data pane you can drill down from an instance to an association, and from there to another associated instance without any limit. The data displayed may become repetitive at some point.

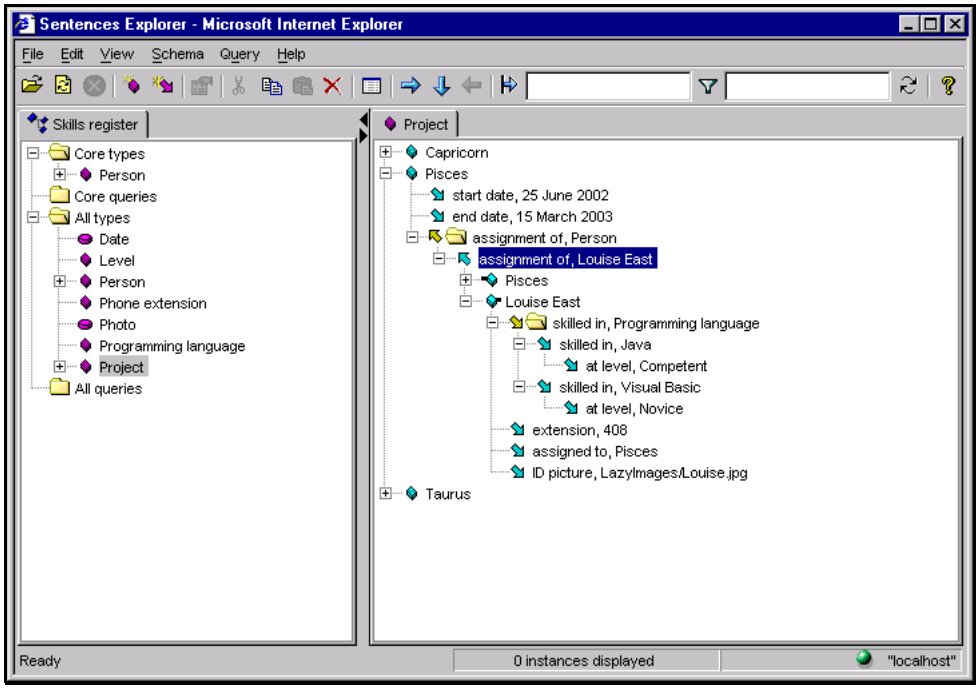

Figure 44 Drilling down through the Sentences Explorer data pane

Figure 44 shows as an example of a drilled down view of data. The sequence of actions that produced this view was as follows:

1. Select and expand the project instance Pisces
2. Expand the associations folder *assignment of*, Person

3. Double-click the association *assignment of,* Louise East
4. Expand the target instance Louise East
5. Expand the associations folder *skilled in*, programming language
6. Expand the association *skilled in,* Visual Basic

Using Explorer tabs

As your application grows the Sentences Explorer tree may become too large to manage. You can make the view of your data more manageable by adding additional tab cards to your Sentences Explorer.

You can choose any selected element in either the schema pane or the data pane to be the source of a new Sentences Explorer tree on a separate tab pane, by clicking the **New Tab** button  on the toolbar. If you select an association type or an association, you can also create a new tab on the target of the association, by clicking the **New Tab on Target** button  on the toolbar.

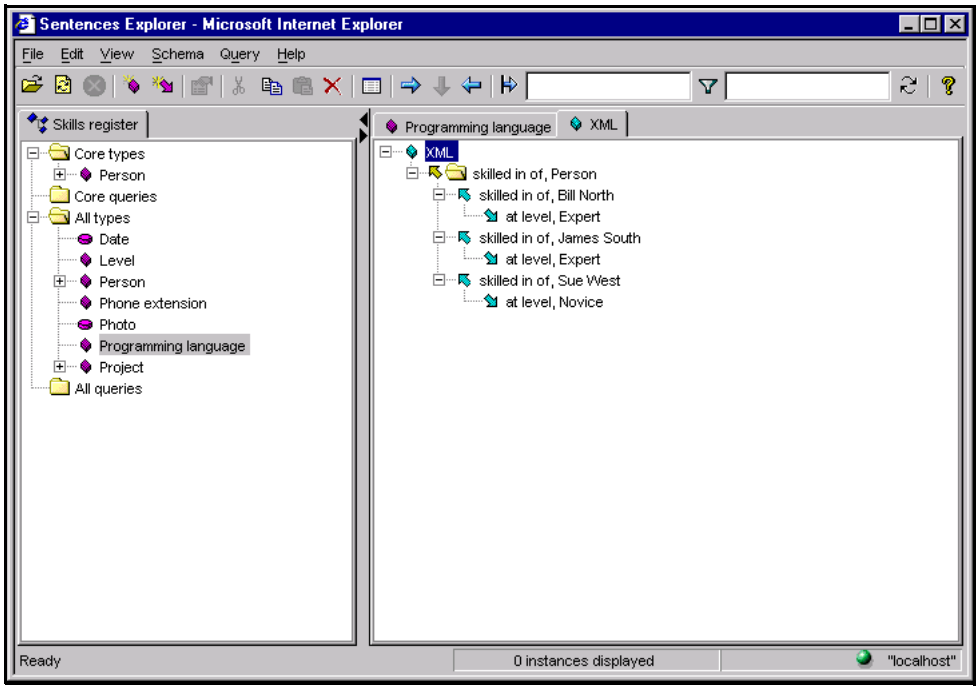

Figure 45 **Using a tab in the Sentences Explorer data pane**

Figure 45 shows a new display tab based on the instance XML.

The use of tabs does not affect your data in any way. It is designed to simplify your access to data. There is no limit to the number of tabs you can create during a session, but tabs are not saved when you end your Sentences session.

Using the Sentences Query Editor

Sentences includes its own query language that can be used to interrogate the database, and this section of the tutorial demonstrates some basic features of the Sentences Query Editor. The Query Editor has other uses which are beyond the scope of this tutorial, but which are described in the *Sentences User's Guide*. For example, the Query Editor is used for designing custom Dataforms.

The Query Editor is based on the associative model of data and uses trees and operations on trees. This means that a query is a tree of entity and association types that you select to bring you the data you need. The query results are also in the form of a tree of data.

In order to make the results of the example queries more meaningful you must first spend a couple of minutes adding some more data. The following table lists some data about four more programmers and their skills. These names appear in the Human resources example profile, but the details used in this tutorial are different.

You must enter a Phone extension for each person you add as you made this a **Mandatory** association. There are no ID pictures available for these additional people.

First add the new names, their phone extensions, and their project assignments:

Person	Phone extension	Project assignment
Barney Norris	413	Taurus
Freda Quentin	415	Capricorn
Harold Ferny	417	Pisces
Mary Venice	419	(not assigned)

Next add their programming languages and skill levels:

Person	Programming language	Level
Barney Norris	Java	Expert
Barney Norris	XML	Expert
Freda Quentin	C++	Expert
Freda Quentin	Visual Basic	Novice
Harold Ferny	C++	Novice
Harold Ferny	Java	Competent
Mary Venice	Java	Expert
Mary Venice	XML	Competent

You can now start creating your new query.

Creating a new query

To create a new query, follow these steps:

1. In the Sentences Explorer right-click the **All Queries** folder, and select **Create Query** from the shortcut menu.

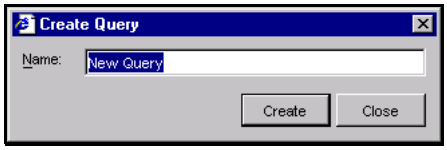

Figure 46 **Create query dialog**

2. In the **Create Query** dialog, give your query the name Availability and click **Create**. Sentences immediately opens the Query Editor. The Query Editor is shown in Figure 47.

   If you want to edit an existing query you can right-click on the query name in the **All Queries** folder and select **Edit Query** from the shortcut menu. Alternatively you can open the Query Editor by highlighting a query and clicking the **Properties** button on the Toolbar.

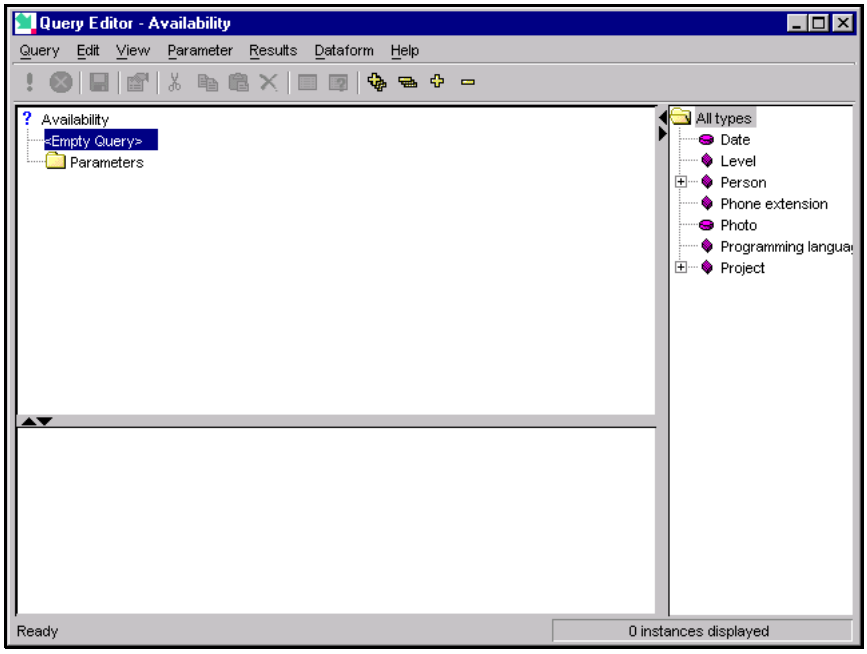

Figure 47 **The Query Editor**

The Query Editor has three panes. On the right is a schema pane showing schema types from the Sentences Explorer. The top left-hand pane is the Query Editor itself, and the bottom left-hand pane is used to display the query results. When you start a new query, the schema pane shows the **All types** folder from the Sentences Explorer and all of its contents. As you build your query this pane only shows the schema elements that may be added to the currently selected query node.

As a first step you should formulate your query as a simple question, as this helps identify the entity or association type from which your query type is developed. For example, a question like “Who is not assigned to any project after 15th March 2004?” indicates that the result you expect is a list of people, your query must start with the entity type in your application that represents people, which is Person.

The first element in your query is known as the data request node. You can add additional nodes based on the associations that have the entity type chosen as the data request node as their source. Building the query tree in this way can continue to whatever depth is needed. You can also add special sort, selection, and calculation nodes to your query.

Building a query using drag-and-drop

You can use drag and drop to add elements to your query.

1. Drag the Person entity type from the **All types** folder to the <Empty query> placeholder, as shown in Figure 48.

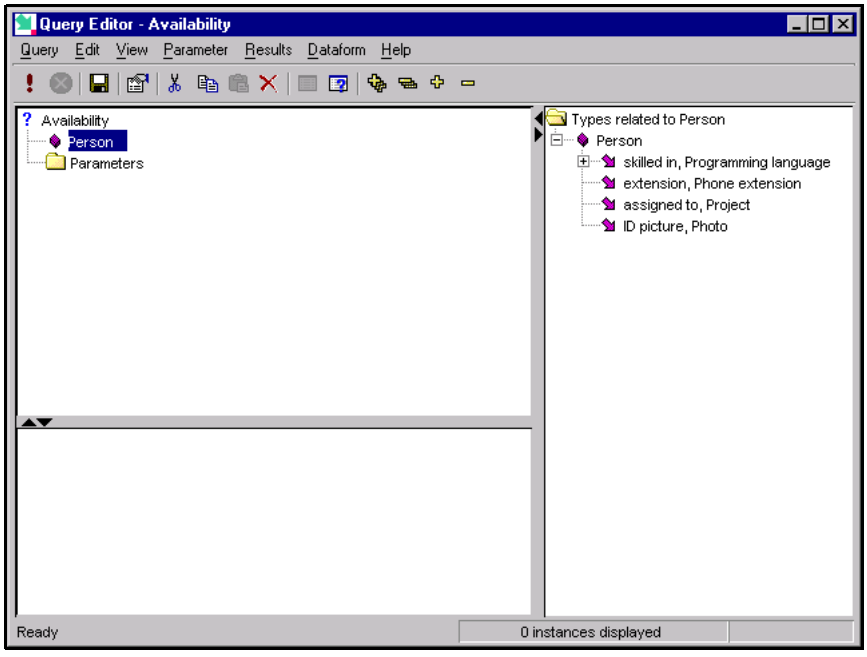

Figure 48 **The first node of the query (the data request node)**

2. In the Query Editor, click the expand button next to Person in the schema pane. Highlight the association *assigned to*, Project, and drag-and-drop this association onto the Person node. The appearance of your mouse cursor changes from an arrow to a “no entry” sign as you pass over positions where you cannot drop the item you are dragging, and changes into a rectangle as you pass over locations where you can drop the object.

Saving queries

You can make changes to your query in the Query Editor and run it (as explained in the section “Running queries and viewing results” on page 64) to test your query

structure as you develop it. However, unlike data changes in the Explorer, changes made in the Query Editor need to be saved explicitly.

You can save the query by clicking the **Save** button  on the toolbar or by selecting **Save as…** from the **Query** menu and giving your query a name. After you have saved the query once you can use the command **Save Query** on the **Query** menu to save it.

Only saved queries are displayed in the **All queries** folder in the Sentences Explorer, and these are available to other users who have access to the same Sentences server.

Running queries and viewing results

You can view your query results in the Query Editor as you build it.

1. Click the Execute button on the Query Editor toolbar  or select **Execute Query** from the **Results** menu.

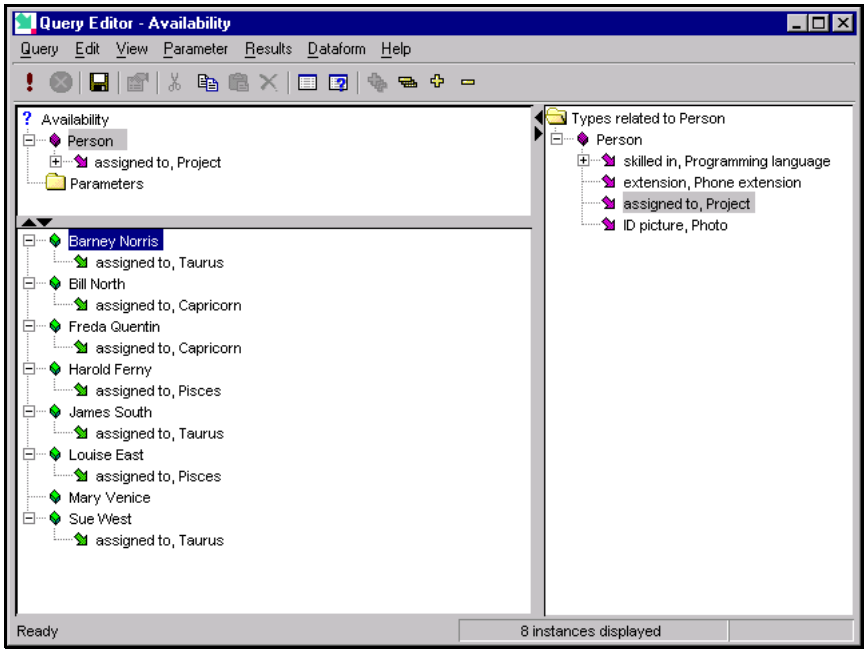

Figure 49 **Query results with assigned to Project node added**

The results are shown in Figure 49. You can see that for each person selected the results display the project to which they are assigned. You can also see that query result icons are always green.

Using a Required node

The query we are creating looks at the dates of assignments but the results shown in Figure 49 list all eight programmers in the database, even though one of them has not been assigned to a project. It is possible to exclude results that do not have the associations referred to by a query node by making that a **Required** node.

1. Highlight the node *assigned to*, Project.
2. Select **Set Required** from the **Edit** menu or from the shortcut menu. The node now shows an asterisk as a reminder that is **Required**.
3. Run the query. The results are shown in Figure 50, and now include only seven programmers.

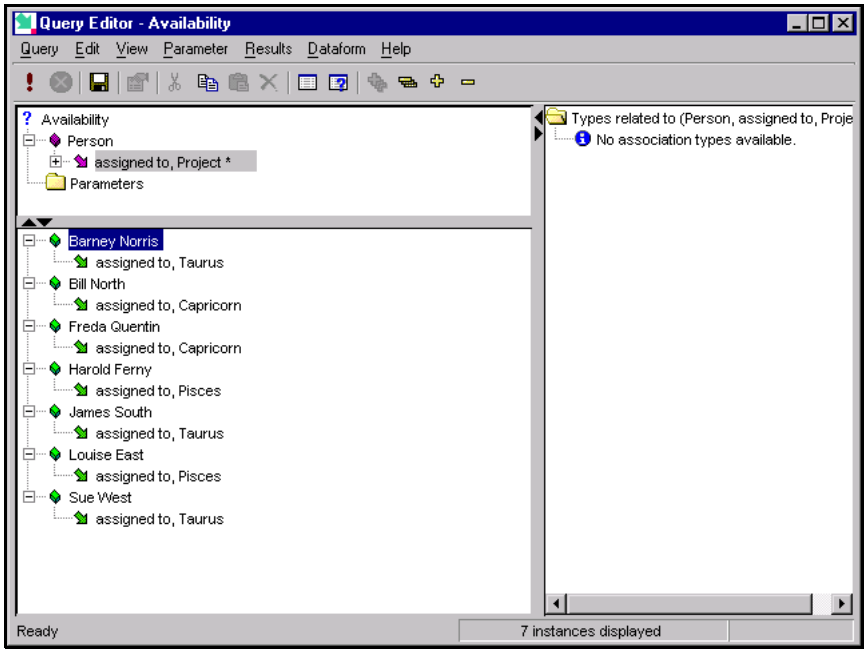

Figure 50 **Query results after using Required node**

On reflection, the restriction brought by the **Required** node is not actually useful in this query. If a programmer does not have any assignment listed at all, then they are available after a given date, so it is wrong to exclude them. Other ways of selecting results are shown in later steps. At this stage you must cancel the **Required** node.

1. To cancel a **Required** node, highlight the node and select **Set Optional** from the **Edit** menu or from the shortcut menu.

Adding nodes using menu options

As well as using drag-and-drop you can use menu options to add nodes to the query.

1. In the query pane, click the expand button next to *assigned to*, Project. Sentences displays the target, Project, as a child node.
2. Highlight the node Project and right-click to display the shortcut menu. Select **Add** to display the sub-menu, and then select **Associations** to display the **Add Associations** dialog. Select the associations Project, *end date,* Date and select the **Add** button to add it as a node in the query.

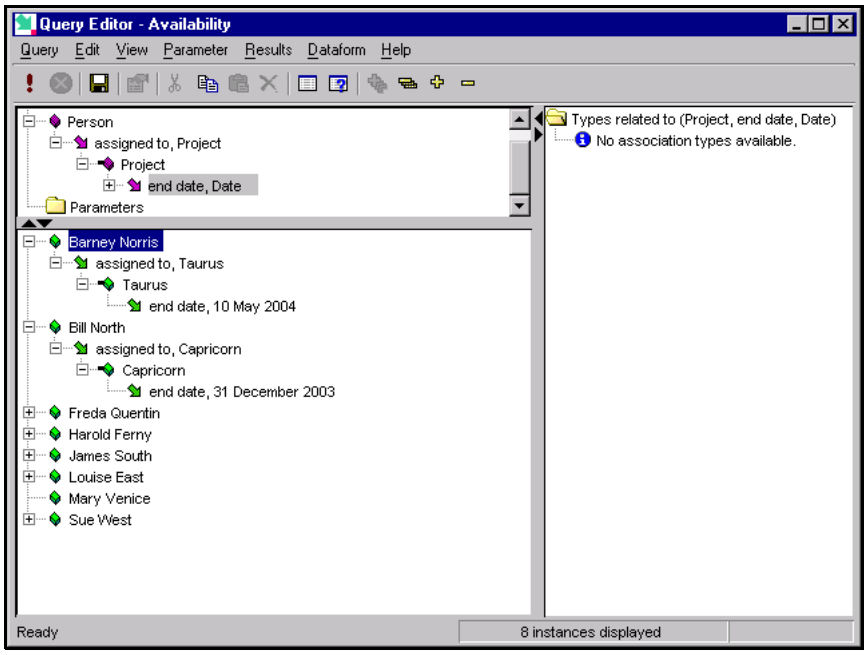

Figure 51 **Query results showing project end dates**

3. You can now execute the query. The query results now show the end date for each project, as shown in Figure 51. Remember to save your query again.

Using expressions in queries

The following sections of the tutorial demonstrate the use of sort nodes, selection nodes, and derived type nodes in queries. All of these features use expressions to calculate, filter or sort results.

The expressions used in the examples in this tutorial each use one particular method or one particular comparison operator that can be used in expressions. Details of all the available methods, constants, arithmetic and comparison operators, logical connectives, and so on, that can be used in query expressions are given in the *Sentences User's Guide*.

Using a selection node

You can now add a selection node with an expression to the query so that you can select those people who are available after the chosen date.

To add a selection node, follow these steps:

1. Highlight the node Person in the Query Editor and select **Add** then **Selection** from the **Edit** menu or from the shortcut menu. Selection nodes are always added to the parent node of the node on which you want the selection filter to operate.

   Sentences adds an empty selection node which can be edited immediately, as a child of Person, as shown in Figure 52.

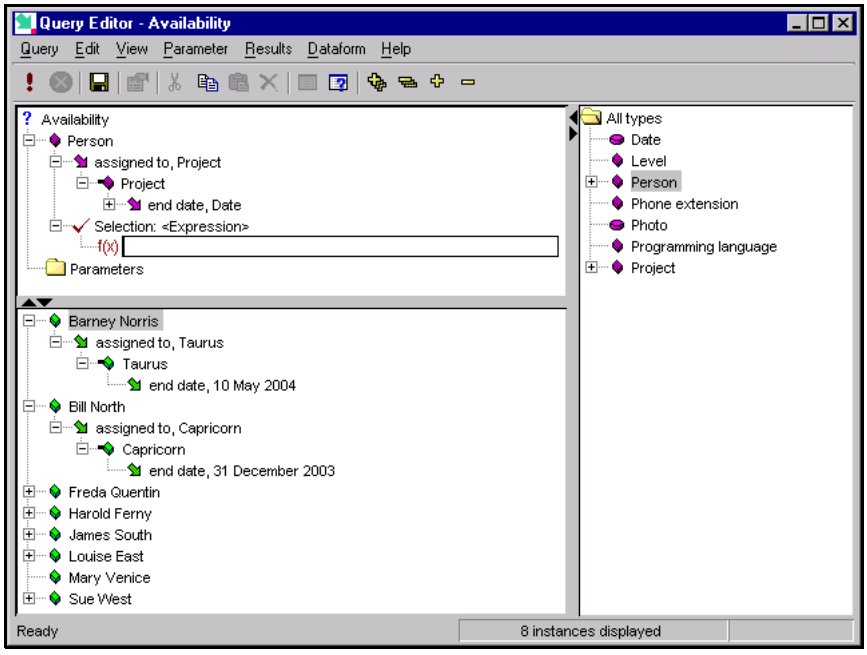

Figure 52 **Adding a selection node**

   If the selection node is not editable, right-click the expression node (<Expression>) and select **Edit expression** from the shortcut menu.

2. To find the programmers available after 15th March 2004, you want to select the programmers assigned to projects that have an end date that is on or before that date (that is, a date that is less than or equal to the specified date).

   The actual expression you must type in for this query is:
   Date<=Date("15/03/2004")
   which means "select those dates which are less than or equal to the specific date 15th March 2004". As there is only one Date node in this query, and that node is the target of the *end date* association, you do not need to take any extra steps to

specify which Date this selection expression refers to. If the query contained more than one Date node you would be able to specify a particular Date node by renaming it and referring to the renamed node in the query expression.

Note *Dates in expressions in Sentences queries must be expressed in the locale of the Sentences server. See the Note on page 46 for more details about Sentences and locales.*

As this selection node was added to Person, the results for this selection node only shows those Person entities for whom this date selection is true. The other Person entities are not shown.

After you type in the expression, press **Enter** or click outside the expression edit box.

3. Run the query. The result of the query is shown in Figure 53, showing the programmers who are assigned to projects that end before the date specified in the expression.

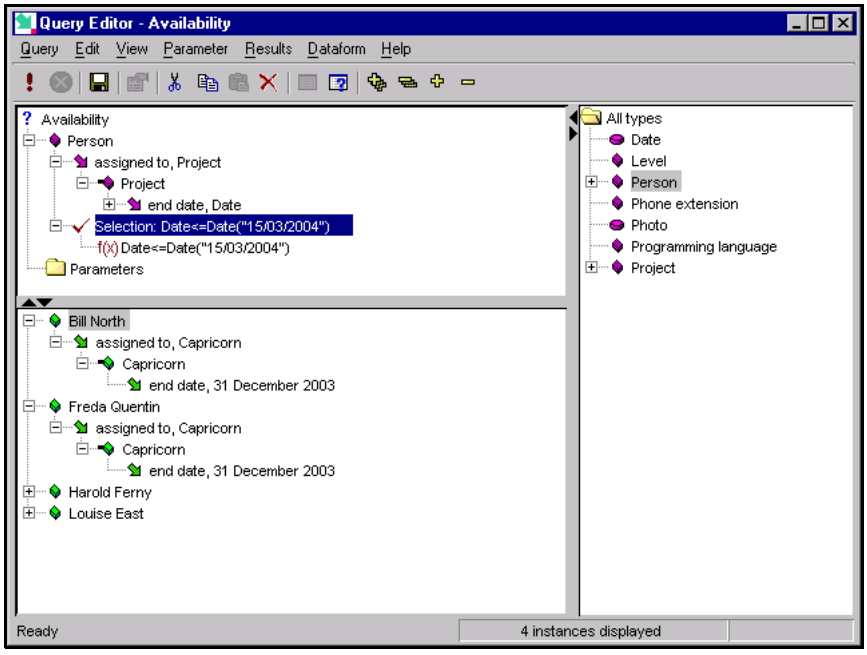

Figure 53 **Query results after selection by date**

However, this does not actually show all the programmers who are available, as some may not have been assigned to any project at all. In the Sentences database, the Project, end date, date association does not exist for these programmers.

One of the functions available in Sentences queries can be used to find associations that exist or do not exist. This is the `Exists()` conditional function. The negative version of this function is made by adding the ! mark before it to create a “does not exist” function `!(Exists())`.

In Sentences queries you can join two conditions in the same expression by using AND (both expressions must be true) OR (either expression must be true) or XOR (one but not both of the expressions must be true). In this case OR is appropriate.

4. Highlight the Selection expression and select **Edit expression** from the **Edit** menu or from the shortcut menu.
5. Edit the expression to read as follows:
   Date<=Date(“15/03/2004”) OR !(Exists(Date))
   and press **Enter**.
6. Execute the query. The results are shown in the illustration Figure 54.

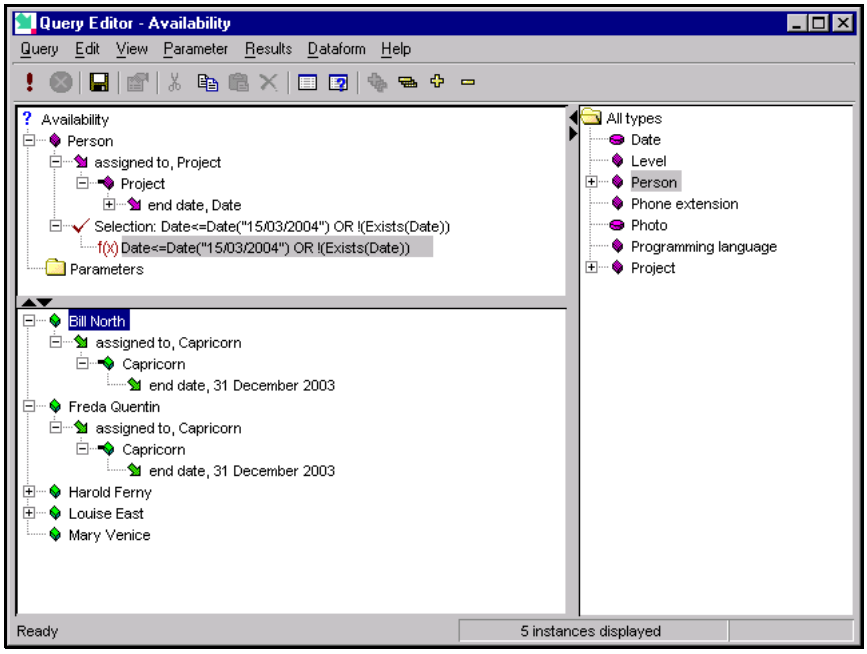

Figure 54 **Results including unassigned programmers**

Using a sort node

You may wish to view your results in a particular order. You can do this by adding a **Sort** node to the query.

1. Highlight the node Person in the Query Editor and select **Add** then **Sort** from the **Edit** menu or from the shortcut menu.

   Sentences adds an empty sort node which can be edited immediately, as a child of Person, as shown in Figure 55.

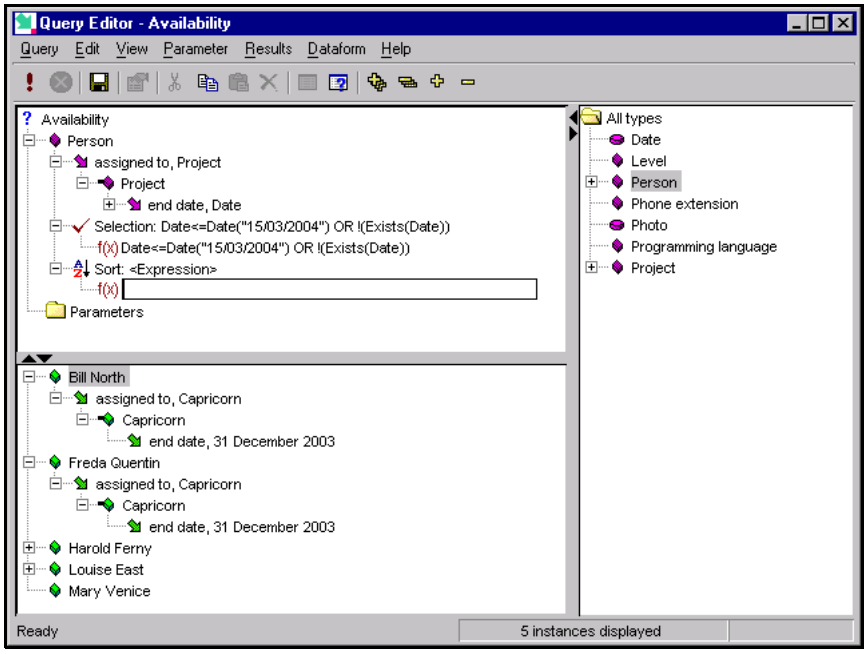

Figure 55 **Adding a Sort node**

If the sort node is not editable, right-click the expression node (<Expression>) and select **Edit expression** from the shortcut menu.

2. For the **Sort** node expression you need to enter the name of the node you wish to sort your results on, followed by either A for ascending order, or D. Assume you have a reason for viewing the programmers names in reverse alphabetical order. (In this application, each programmers name is their first name and last name, so this action sorts them by the first letter of their first name.) Type in Person, D as the expression and press **Enter** or click outside the expression edit box.

3. Run the query. The sorted results are shown in Figure 56.

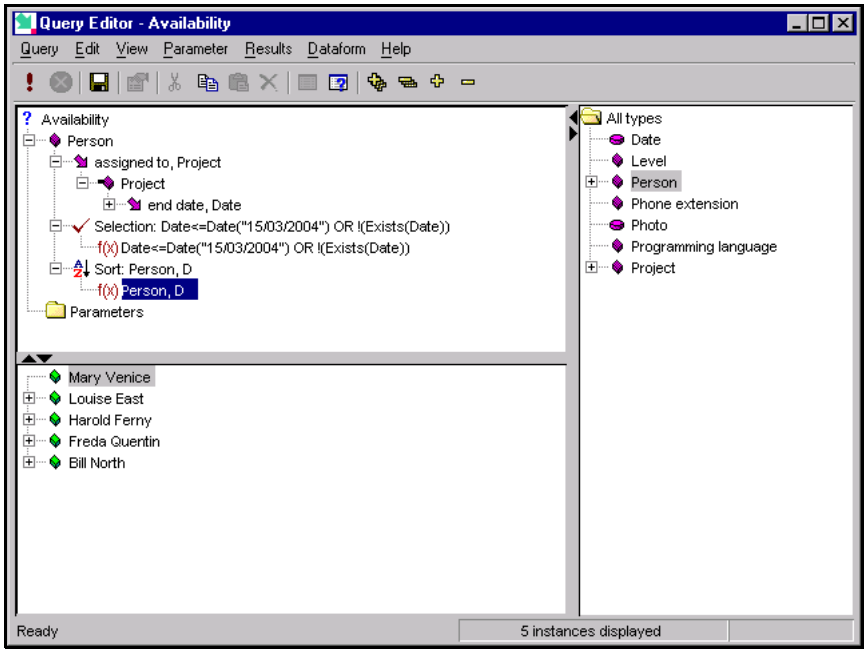

Figure 56 **Query results after Sort by Descending alphabetical order**

Binding a node to an instance

You can restrict your query results to those instances where a specific association exists by linking a query node to a specific instance. This is known as “binding a node”. To demonstrate this you can now create a new query.

1. In Sentences Explorer, select **Create Query** from the Query menu, or from the shortcut menu on the **All queries** folder.
2. Create a new query named Expertise and open the Query Editor.
3. Start the query with Person as the data request node, and add the association *skilled in*, Programming language.
4. Expand the *skilled in*, Programming language node and add the association *at level*, Level.
5. Run the query. Parts of the results are shown in Figure 57.

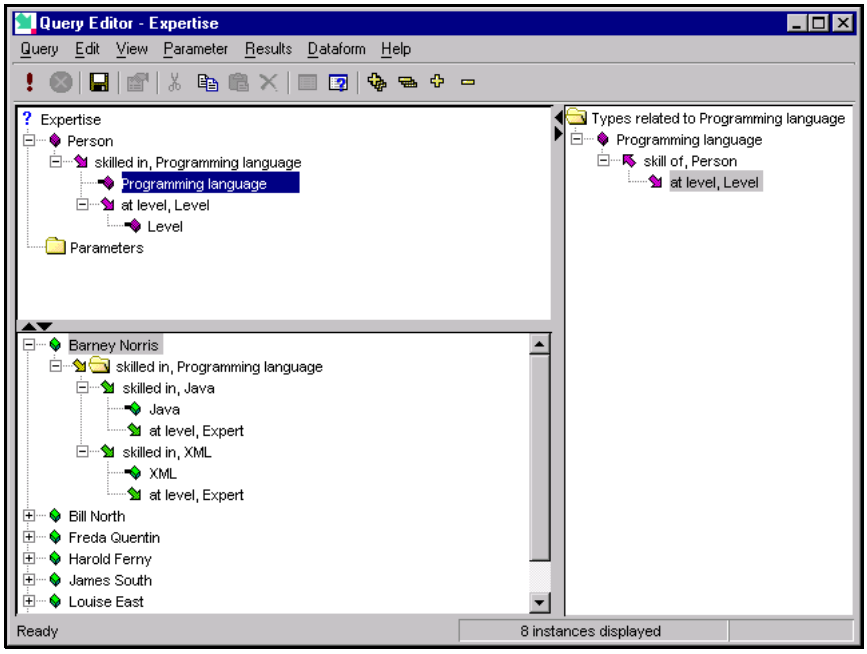

Figure 57 **Unrestricted results for the Expertise query**

6. Highlight the node Programming language and select **Bind to instance** from the **Edit** menu or the shortcut menu.
7. Select Java from the picker list.
8. Run the query. The results are shown in Figure 58. The query results include all eight programmers, but only those who have skill in Java are shown with expand buttons.

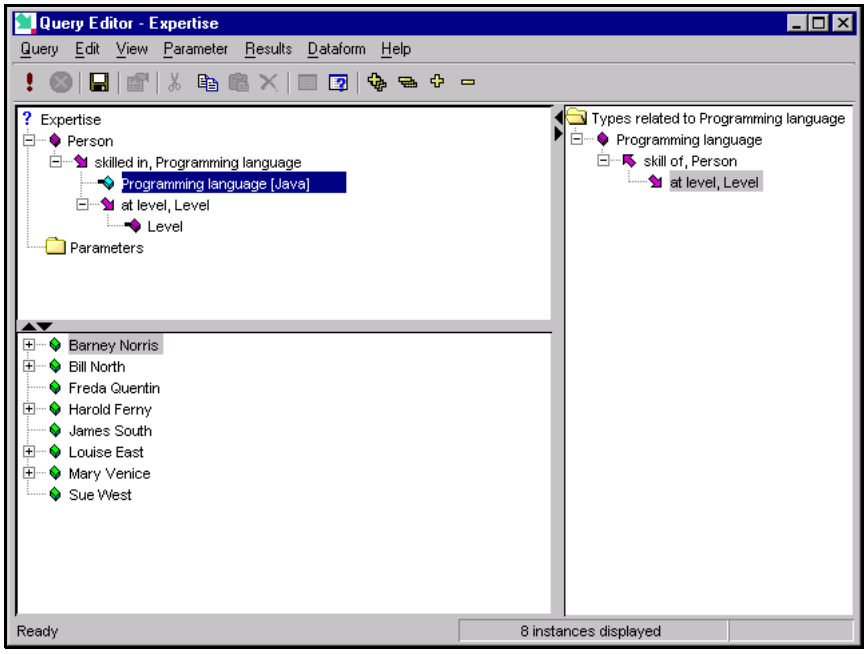

Figure 58 **Expand buttons shown for programmers with Java skills**

9. To remove those programmers who do not have Java skills from the query results, highlight the node *skilled in*, Programming language and make it **Required** by selecting **Set Required** from the **Edit** menu or from the shortcut menu.

10. Run the query. The results are shown in Figure 59 and now only include the five programmers with Java skills.

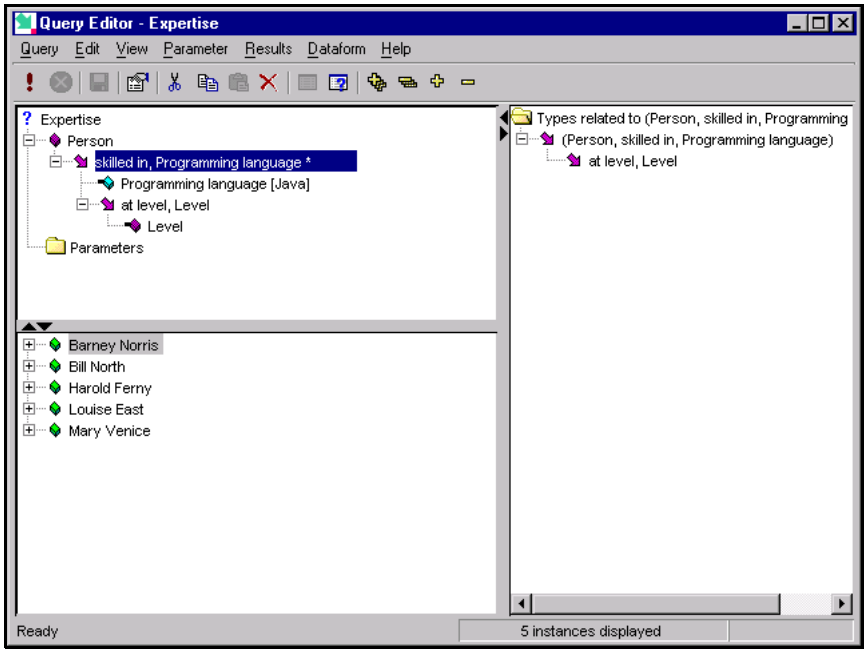

Figure 59 **Results showing only Java programmers**

11. To restrict the results to those programmers who are Expert in Java, highlight the Level node, and bind it to the instance Expert. Also make the node *at level*, Level into a **Required** node.

12. Run the query. Look at the results shown in Figure 60. You have now restricted your query results to show you only the two programmers who are Java experts. You should save your query before moving on to the next section.

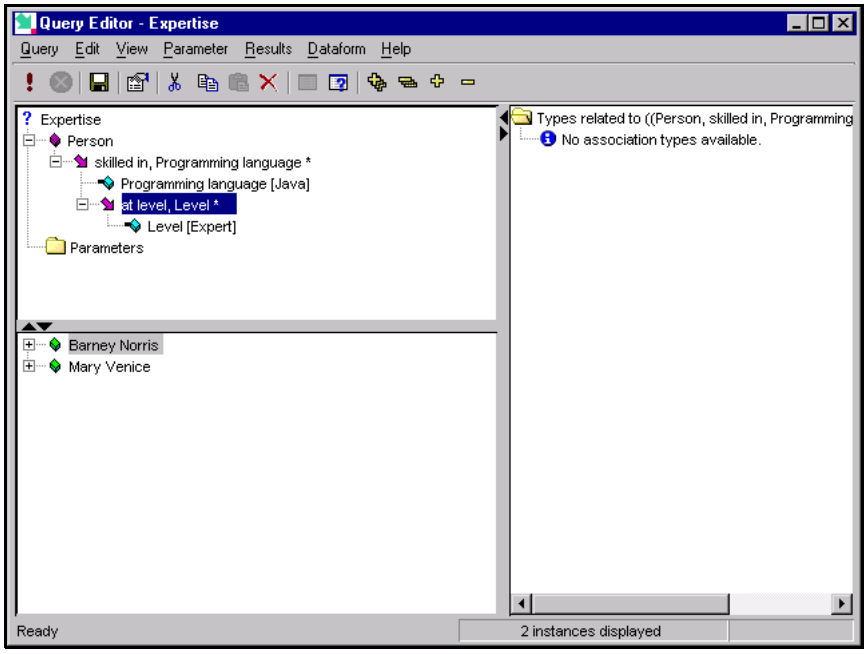

Figure 60 **results showing only expert Java programmers**

Using query parameters

You can make the query interactive by using parameters to replace node binding.

1. Highlight the Programming language node and select **Clear Instance Binding** from the Edit menu or the shortcut menu. Do the same for the Level node.
2. Drag the Programming language node and drop it onto the **Parameters** folder. Do the same with the Level node.

   There are now two parameters visible in the Query Editor as shown in Figure 61.

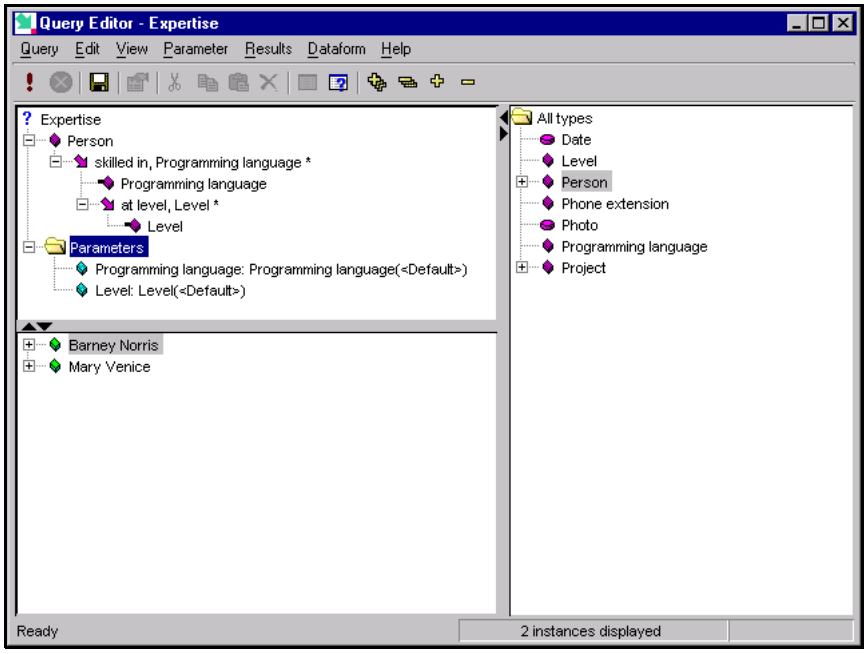

Figure 61 **Parameters folder showing two parameters**

3. Run the query. Instead of displaying the results immediately Sentences displays the **Parameters** dialog, as shown in Figure 62, and waits for your selections.

Figure 62 **The Parameters dialog**

You can now use the ellipsis buttons in the **Parameters** dialog to open a picker list and select values for Programming language and Level each time you run the query. You should save your query before moving on to the next section.

Note *You can set default values for* **Parameters**. *For details of how to do this, see the Sentences User's Guide.*

Using a derived type

You can add simple calculations to your query by adding a derived type node. A derived type has an expression which is calculated each time the query is run, but the resulting value is not saved.

Note *You can add more complex calculations to queries using other methods. For details see the Sentences User's Guide.*

In the current query, Expertise, you can add a derived type to calculate the number of people who have a specified skill at a specified level.

1. Highlight the query name Expertise and select **Add** then **Derived Type** from the **Edit** menu, or select **Add Derived Type** from the shortcut menu.

   Sentences adds an empty derived type node as a child of Expertise, as shown in Figure 63.

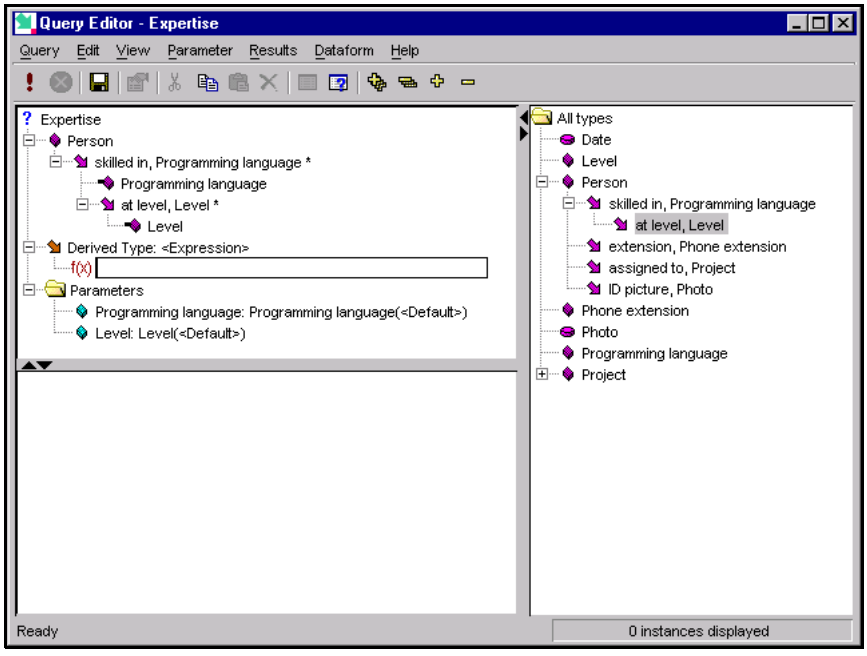

Figure 63 **Adding a derived type node**

2. Type in Count(Person) as the expression and press **Enter** or click outside the expression edit box.

3. Run the query. When Sentences displays the **Parameters** dialog, select Java and Competent. In the results shown in Figure 64, Sentences has calculated a value for the derived type.

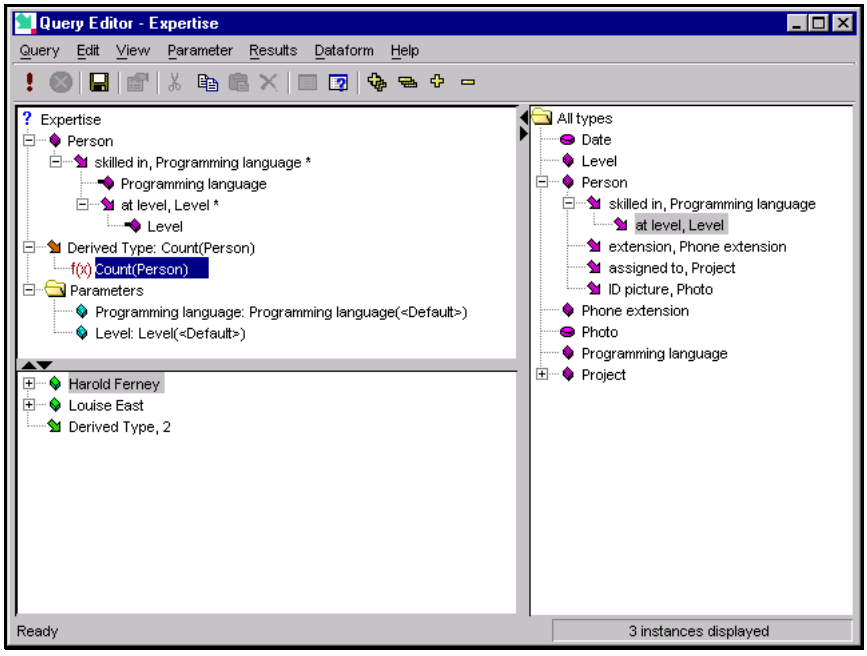

Figure 64 **Results showing the calculated derived type**

4. The line in the result that reads Derived Type, 2 is not very useful. To give the derived type a meaningful name, highlight the derived type line in the Query Editor and select **Rename** from the **Edit** menu or the shortcut menu. Type in the new name Number of people.

5. Run the query. When Sentences displays the parameters dialog, select Java and Competent again. In the results shown in Figure 65, Sentences shows a calculated a value for the derived type Number of People as well as the names of the programmers.

   You should save your query before moving on to the next section.

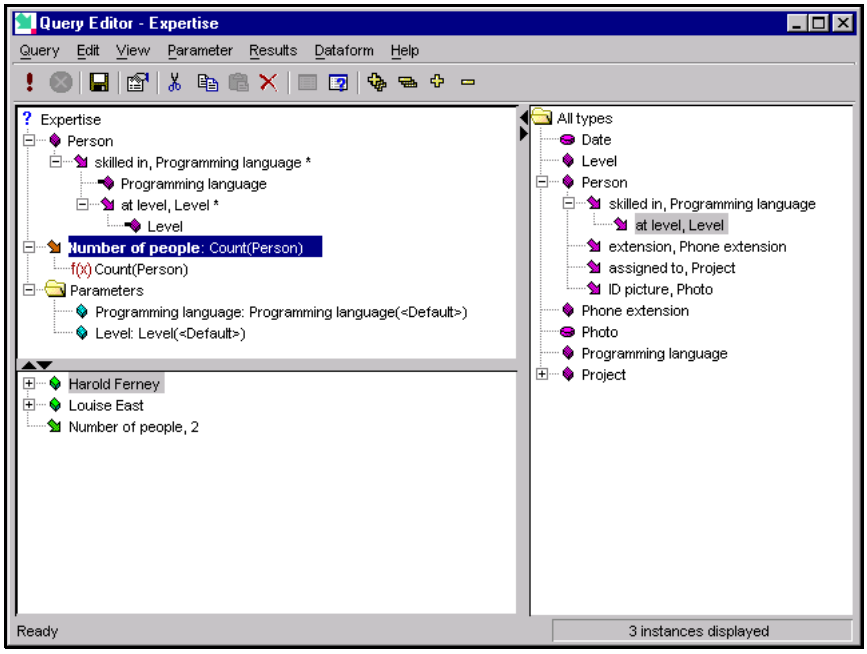

Figure 65 **Results showing the renamed derived type node**

Hiding a branch

It is possible that you are not interested in the names of the programmers who have a specified skill at a specified level, but only in the number of programmers. Sentences allows you to hide parts of the results tree. This does not affect the way the query is run or evaluated, but only the way the results are displayed.

1. In the Query Editor highlight the node Person, and select **Hide Branch** from the **Edit** menu or from the shortcut menu. Sentences now displays the Person node and all its child nodes in grey. These nodes are not be shown in the results display.

2. Run the query. When Sentences displays the parameters dialog, select Java and Competent. In the results shown in Figure 66, Sentences only shows the calculated value for the derived type Number of People, and does not show any names.

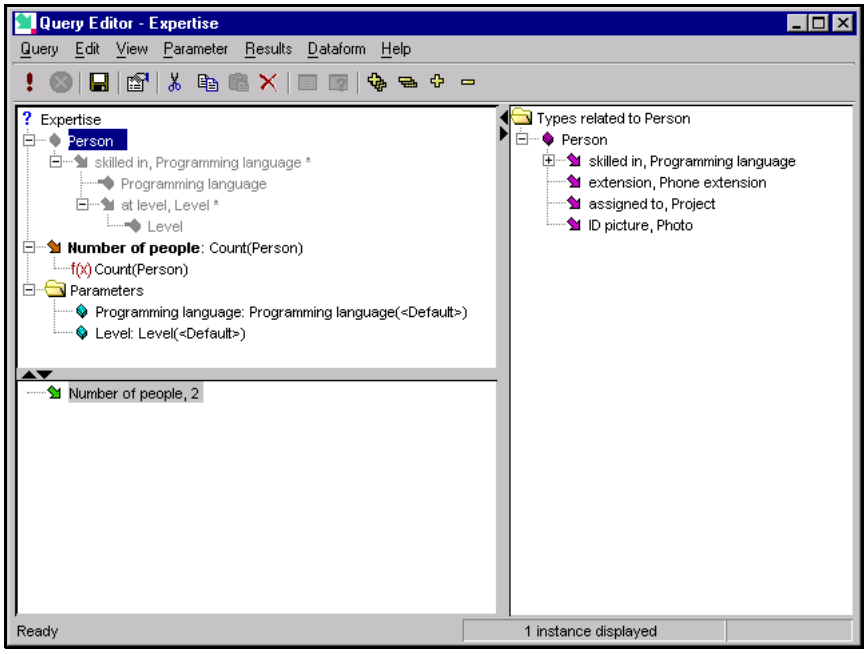

Figure 66 **Results after hiding a branch**

Viewing query results in the Sentences Explorer

You can now see what happens when you run this query from the Sentences Explorer.

1. Save the query to make sure the latest version is available in the Sentences Explorer, and then close the Query Editor.
2. In the Sentences Explorer expand the **All Queries** folder.
3. Highlight the query named Expertise and select **Execute Query** from the **Query** menu or from the shortcut menu. When Sentences displays the parameters dialog, select Java and Competent. The query result is displayed in the Explorer data pane as shown in Figure 67.

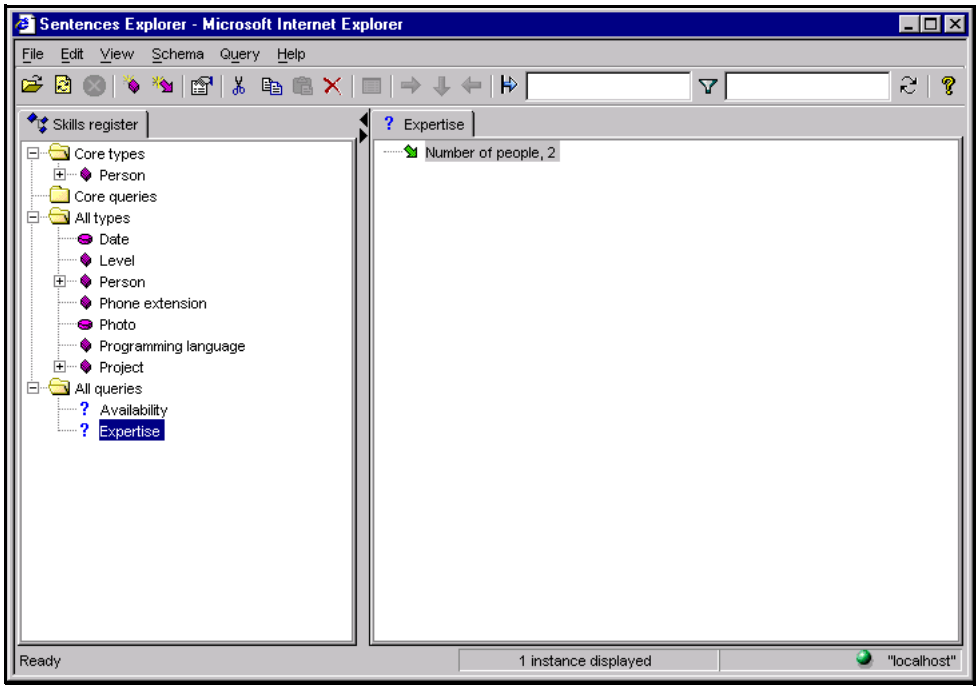

Figure 67 **Query results in the Explorer data pane.**

The results in the Explorer data pane are identical to those shown in the Query Editor. The way the results are displayed, using green icons, is also identical.

Viewing query results using XML

XML is the Extensible Markup Language, which is the universal format for structured data and documents on the Web, developed under the auspices of the World Wide Web Consortium, W3C. Full details of the ways in which Sentences works with XML are given in the *Sentences User's Guide*.

Sentences has the built-in ability to export query results as XML. This XML output can be formatted using XSL stylesheets, and presented in an HTML page in your browser. If you wish, you can then print a copy of the HTML page.

The following illustration of viewing query results using XML is based on the Availability query you created earlier in this tutorial.

1. In the Sentences Explorer, expand the **All Queries** folder and highlight the Availability query.

2. Select **Results**, then **View XML Results** from the shortcut menu. Alternatively, select **Results**, then **View XML Results** from the **Query** menu. (The **View XML Results** command is also available from the **Results** menu in the Query Editor.) Sentences displays the **View XML Results** dialog, as shown inFigure 68.

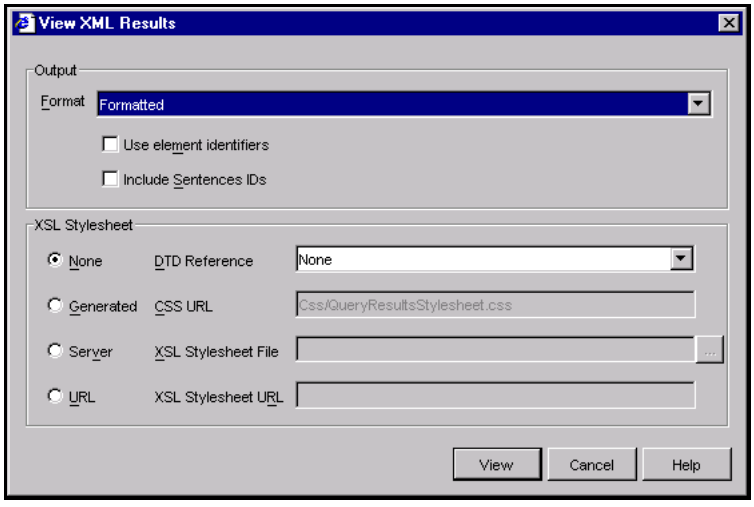

Figure 68 **The View XML Results dialog**

A full description of all the options in this dialog is given in the *Sentences User's Guide*. For the purposes of this tutorial only the **XSL Stylesheet** option is important.

XML documents, including those generated from Sentences queries, contain only data, and do not contain any formatting or display information. Users can add display information by using a stylesheet written in the XML Stylesheet Language, XSL.

Sentences provides two example XSL stylesheets, `master.xsl` and `table.xsl`. These can be found in the `<Sentences_home>\Stylesheets` directory. These are very simple stylesheets and are provided only as basic examples, and do not illustrate all the capabilities of XSL stylesheets.

3. Select the **Server** option on the **View XML Results** dialog and then click the ellipsis button to display the stylesheet picker shown in Figure 69, and select the stylesheet named `master.xsl`.

Sentences is also able to use a stylesheet referenced by a URL, or to generate a stylesheet based on your query. Details of these features are given in the *Sentences User's Guide*.

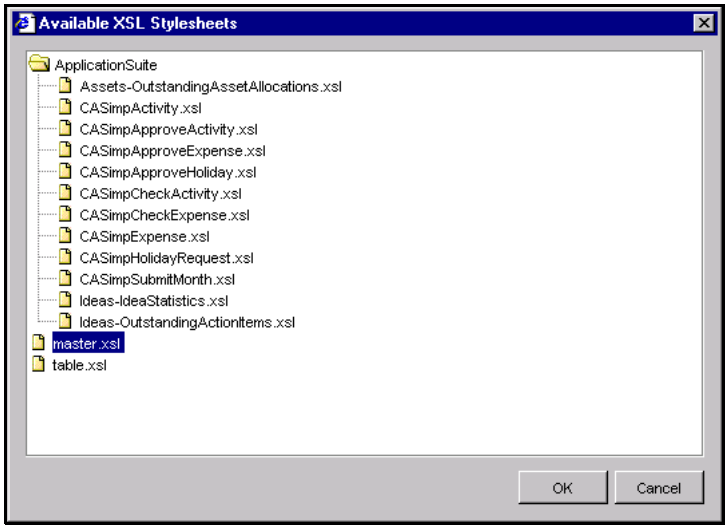

Figure 69 The XML Stylesheet picker

4. Click **View** on the **View XML Results** dialog. Sentences displays a message while processing the query results, and then displays the results in a browser window as shown in Figure 70.

Note *The Sentences display refreshes itself immediately after the results are displayed, and therefore the browser window with the XML results may be hidden by the browser window displaying the Sentences Explorer.*

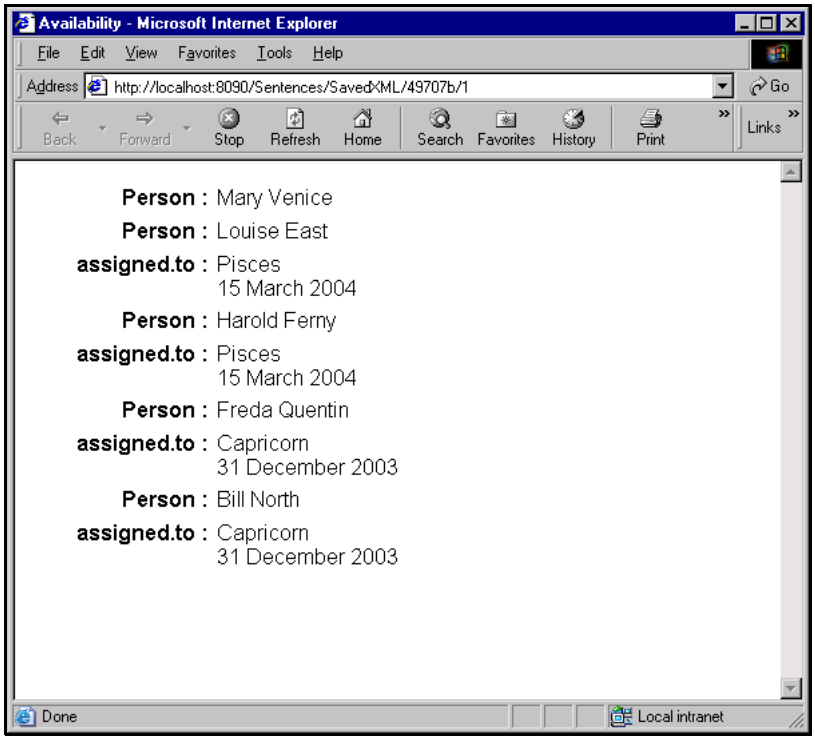

Figure 70 **Results of the Availability query using the master.xsl stylesheet**

The layout shown in Figure 70, including which data is shown with labels (Person and assigned.to) and which data does not have labels, is determined by the XSL stylesheet and can be customised.

To view an alternative format for the same set of query results, do the following:

1. In the Sentences Explorer, expand the **All Queries** folder and highlight the Availability query.
2. Select **Results**, then **View XML Results** from the shortcut menu. Alternatively, select **Results**, then **View XML Results** from the **Query** menu.
3. Select the **Server** option on the **View XML Results** dialog and then click the ellipsis button to display a list of existing stylesheets, and select the stylesheet named `table.xsl`.
4. Click **View**. Sentences displays a message while processing the query results, and then displays the results in a browser window as shown in Figure 71.

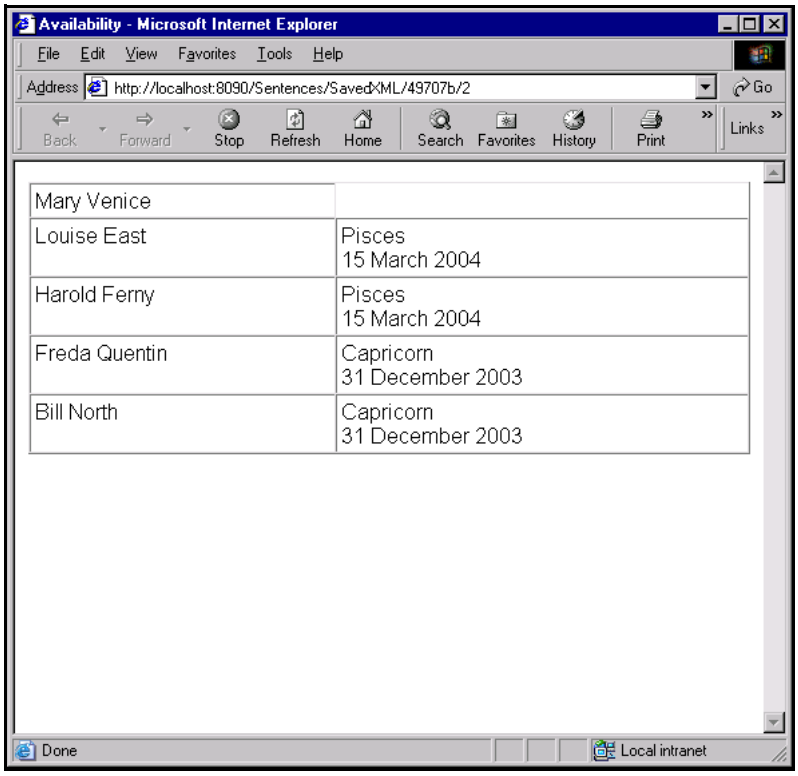

Figure 71 **Results of the availability query using the table.xsl stylesheet**

The table layout, including the width of the cell separator lines, is determined by settings in the XSL stylesheet.

You can create an XSL stylesheet that adds more information to the display, for example, creating a page title based on the query name.

The browser pages that are created by the **View XML Results** command are not saved automatically by Sentences although you can save pages as HTML files. You can also use the **Print** commands available in your browser to print the pages if you wish.

Where to go from here

This concludes the Sentences step-by-step tutorial. You have created a Skills register application which allows you immediate access to useful information about each member of staff using the Sentences Explorer or the Dataform.

You have also constructed a query which tells you the names of programmers available after a certain date, and a query with parameters that gives you the number of programmers with a specific skill.

You can now experiment with your Sentences application and add more entity and association types, and entity and associations instances, as you wish.

The tutorial has shown you just a few examples of what can be done with Sentences, and further sample applications are supplied with your copy of Sentences. Descriptions of these applications are available in the *Application Suite User's Guide* (for Enterprise Edition users) and in the *Personal Edition Supplement* (for Personal Edition users). In addition, to learn more about how to get the best results from Sentences, look at the advanced techniques and the worked examples included in the *User's Guide*.

If you have any questions please contact Lazy Software (`info@lazysoft.com`).

If you have any comments on this tutorial, please send them to the Documentation group at Lazy Software (`documentation@lazysoft.com`).