

# Real-world Apache Derby, Part 1: Apache Derby and OpenOffice.org Calc

## Use Derby as a data store for OpenOffice.org Calc

Skill Level: Intermediate

[Dave Warner \(david.warner@ngc.com\)](mailto:david.warner@ngc.com)  
Senior Database Administrator  
Northrop Grumman IT Solutions

26 Sep 2006

Apache Derby signals a sea change in both desktop and Web-based applications. For the first time, that elusive target for developers -- complete data portability -- is easily attainable. This tutorial, the first in a series, shows how you can use Derby as a data store to overcome row-length limitations in OpenOffice.org's Calc and use that program's interface for data analysis while leaving the storage to Derby.

## Section 1. Before you start

### About this series

Apache Derby is no longer waiting in the wings -- it has taken center stage as a mature, robust database that can be used almost anywhere. This [series of tutorials](#) is for developers or expert users who want to explore the future of data storage. On this journey, expect to combine Derby with other standard tools (both user and developer) to create solutions that solve problems you face every day: ad hoc analysis, document storage, and that newest bugbear, compliance.

### About this tutorial

This tutorial shows you how to set up a Derby database that you can use as a data store for analysis. The data consumer is immaterial, but you'll use one of the more flexible, powerful tools available: OpenOffice.org Calc. The strengths of this approach are data portability, separation of concerns, and the leveraging of standards and best practices. Most approaches to interfacing with data rely on the use of object-relational mapping tools. However, the goal of this tutorial is actually the opposite: no mapping and direct client access to the underlying data.

## Prerequisites

You should be reasonably comfortable with standard Java™ tools and have a smattering of Structured Query Language (SQL) experience. The ability to install and configure a Java Virtual Machine (JVM™), Apache Ant or Eclipse, and Derby is required.

## System requirements

To run the examples in this tutorial, you need:

- [JVM 1.4.x](#) or later (version 1.5 or later is recommended).
- Approximately 100 to 200MB of free space, depending on your choice of toolset.
- At least 64MB of RAM for your virtual machine (VM).

You should also have the following tools to ensure portability -- both between toolsets and operating systems. I've chosen the Eclipse platform and the Derby plug-ins, but you may want to add the Callisto Data Tools plug-ins as well (although you won't use them during this tutorial). You also need the OpenOffice.org suite of tools, chief among them Base and Calc.

Download and install the following programs:

- [Eclipse Workbench, Version 3.x](#)
- [Derby plug-ins for Eclipse](#)
- [OpenOffice.org Suite, Version 2.x](#)
- Optional:
  - [Eclipse Callisto Data Tools plug-ins](#)
  - Microsoft® Office Excel®

---

## Section 2. Set up your development environment

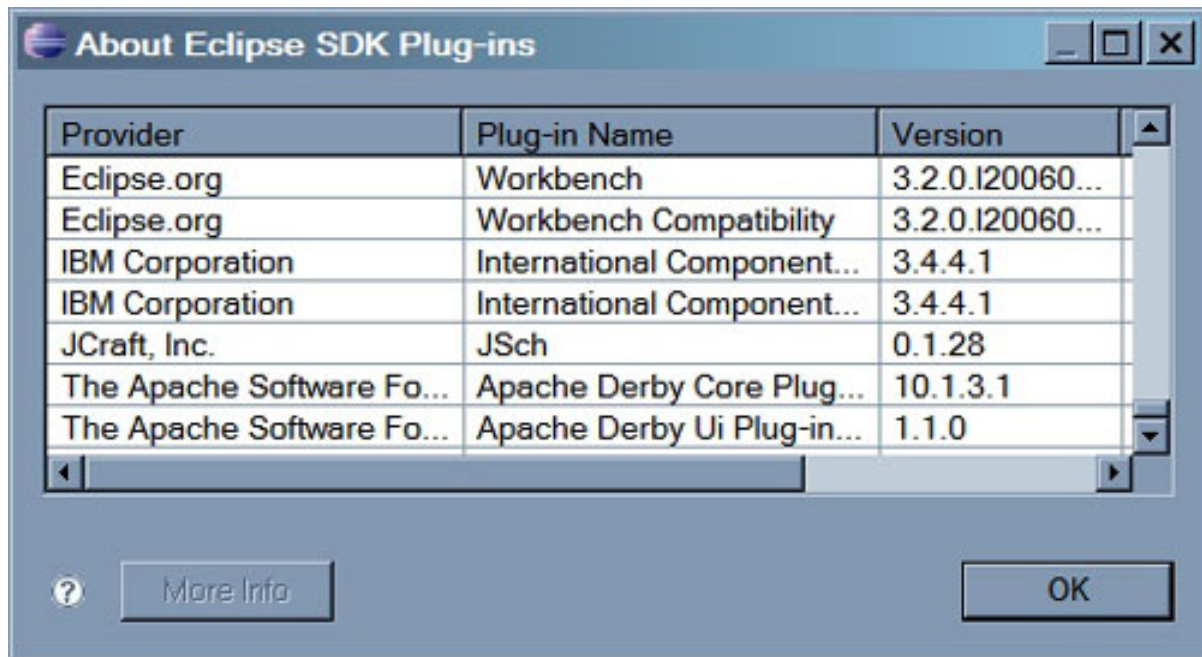
First, install Eclipse and create a new Java project.

### Install Eclipse

Begin by downloading the tools. You need the following:

- The Eclipse, Version 3.2 software development kit (SDK)
  - Apache Derby plug-ins (both core and user interface)
1. Extract the Eclipse SDK download into the directory of your choice. The Derby plug-ins should go into the plug-ins subdirectory of the Eclipse distribution.
  2. When extraction is complete, restart Eclipse.
  3. You can verify whether the Derby plug-ins were installed correctly by opening the Plugin Details window in Eclipse: Click **Help > About Eclipse SDK**, and then click **Plugin Details**. You should see something similar to [Figure 1](#).

#### Figure 1. Checking plug-in installation success



4. Navigate to the Derby Core Plugin subdirectory. You should see the following .jar files along with a plugin.xml document:

- derby.jar
- derbyclient.jar
- derbynet.jar
- derbytools.jar

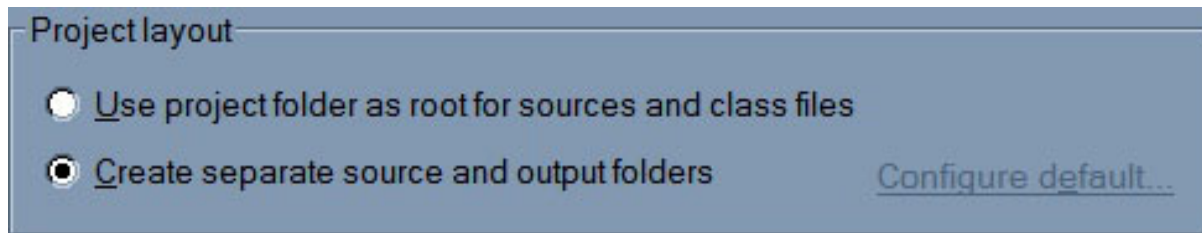
These files constitute the entire Derby distribution, clocking in at well under 3MB. Copy these files to a more convenient location for future use from within OpenOffice.org. Or, if you wish, download the entire Derby distribution, which also includes excellent documentation.

## Create a Java project

Next, set up a Java project.

1. Choose a project layout that uses separate folders for source code and output, as shown in [Figure 2](#).

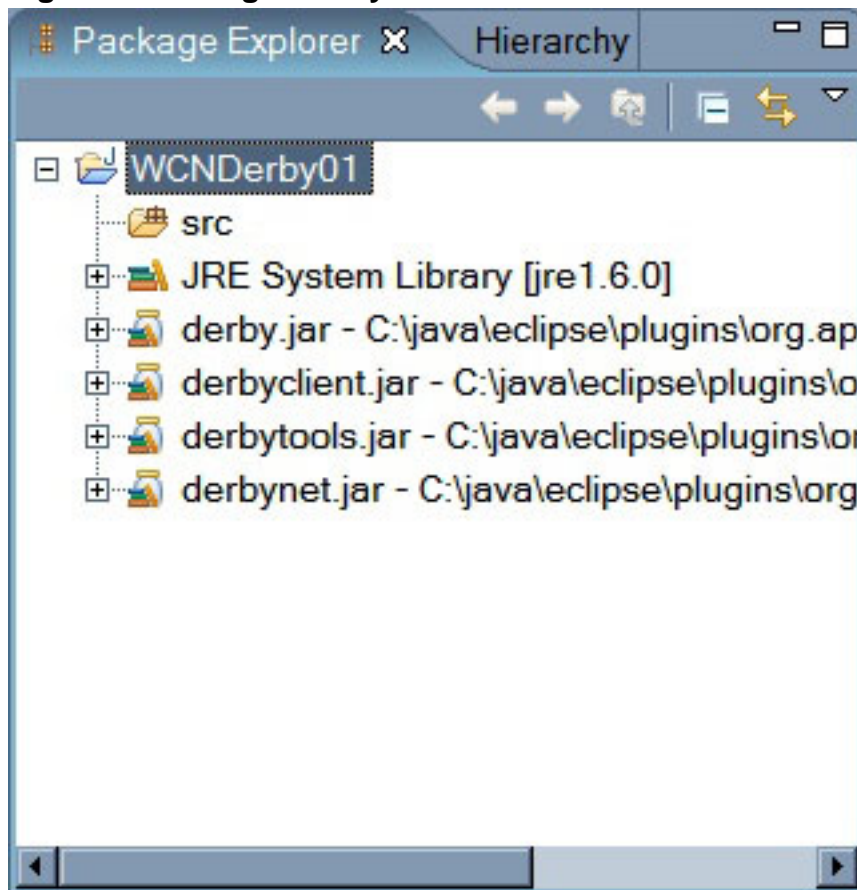
### Figure 2. Project layout



The Derby Eclipse user interface (UI) plug-in works by adding a *nature* to Java projects. By adding this nature, you place the Derby .jar files in the `classpath` variable and add several new commands and options to the project.

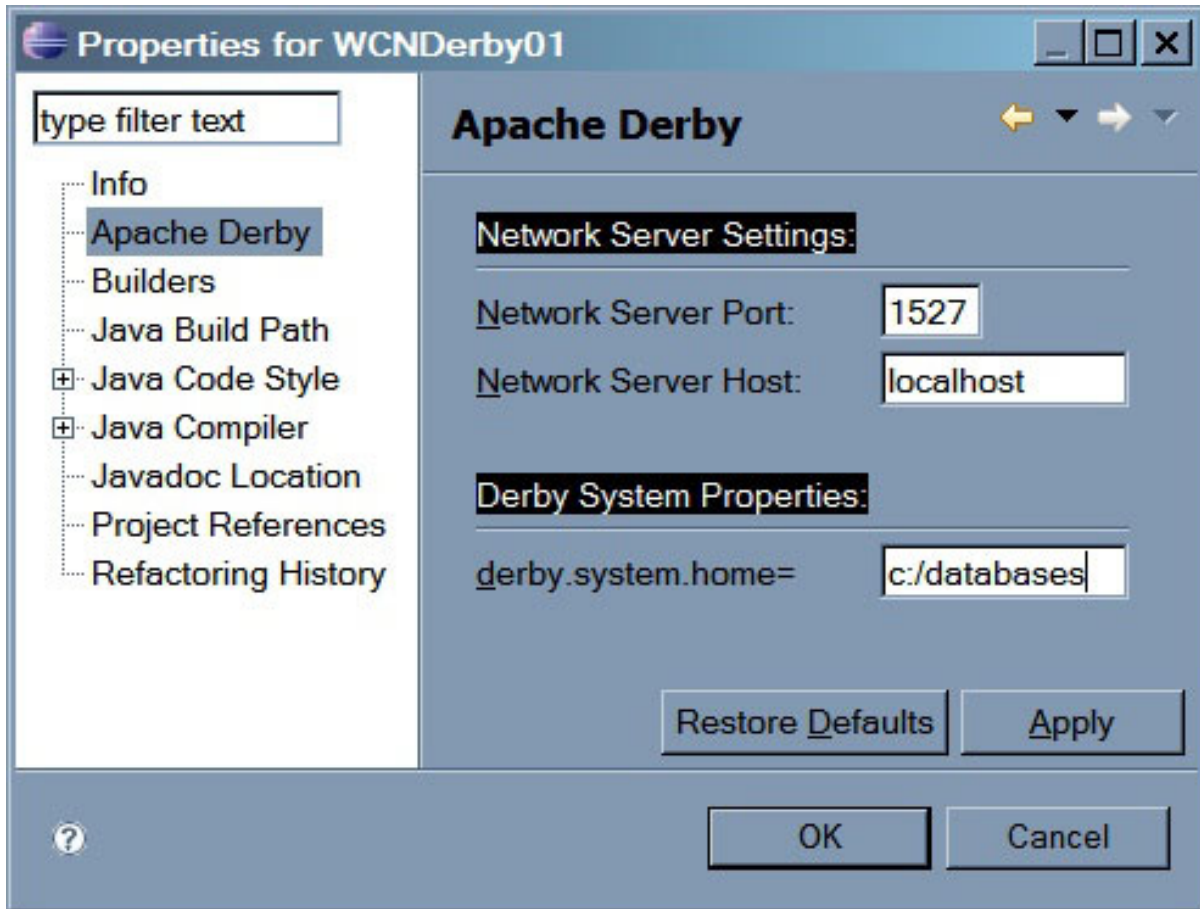
2. Right-click your newly created project within Package Explorer, and then click **Apache Derby > Add Apache Derby Nature**, as shown in [Figure 3](#).

**Figure 3. Adding a Derby nature**



3. Before continuing, right-click the project again, and then click **Properties**.
4. In the properties sheet, select **Apache Derby**, and set the home directory for Derby to something other than the project root directory -- for example, `c:/databases`, as shown in [Figure 4](#).

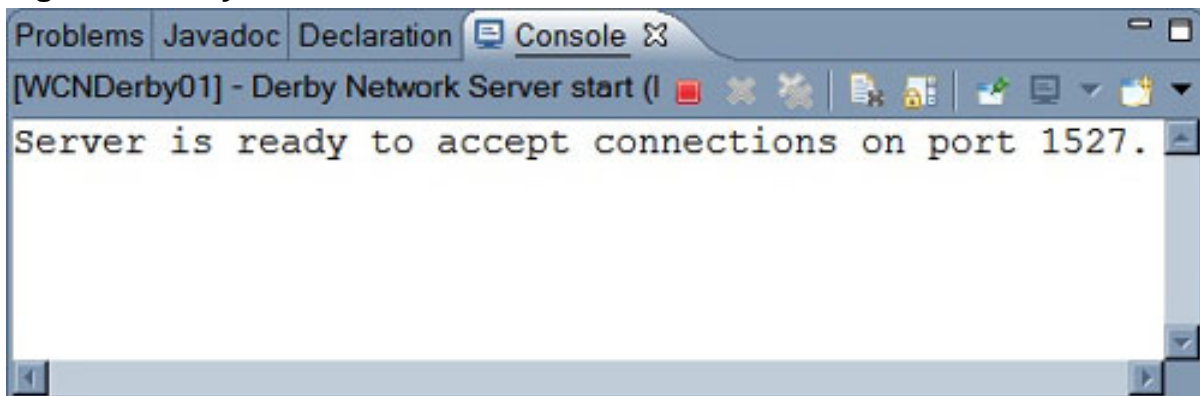
**Figure 4. Derby project settings**



You're now ready to start the Derby network server.

Right-click the project, and then click **Apache Derby > Start Derby Network Server**. The server starts and displays the window shown in [Figure 5](#).

**Figure 5. Derby network server started**



Other tools available to you as part of the Derby nature include commands to stop the network server, ij (an interactive SQL command-line tool), and the `sysinfo`

command, which displays environment settings. Rather than typing commands to create and populate your database, you'll use Ant to automate the process. Before that, however, here are a few best practices you should adopt for your project.

---

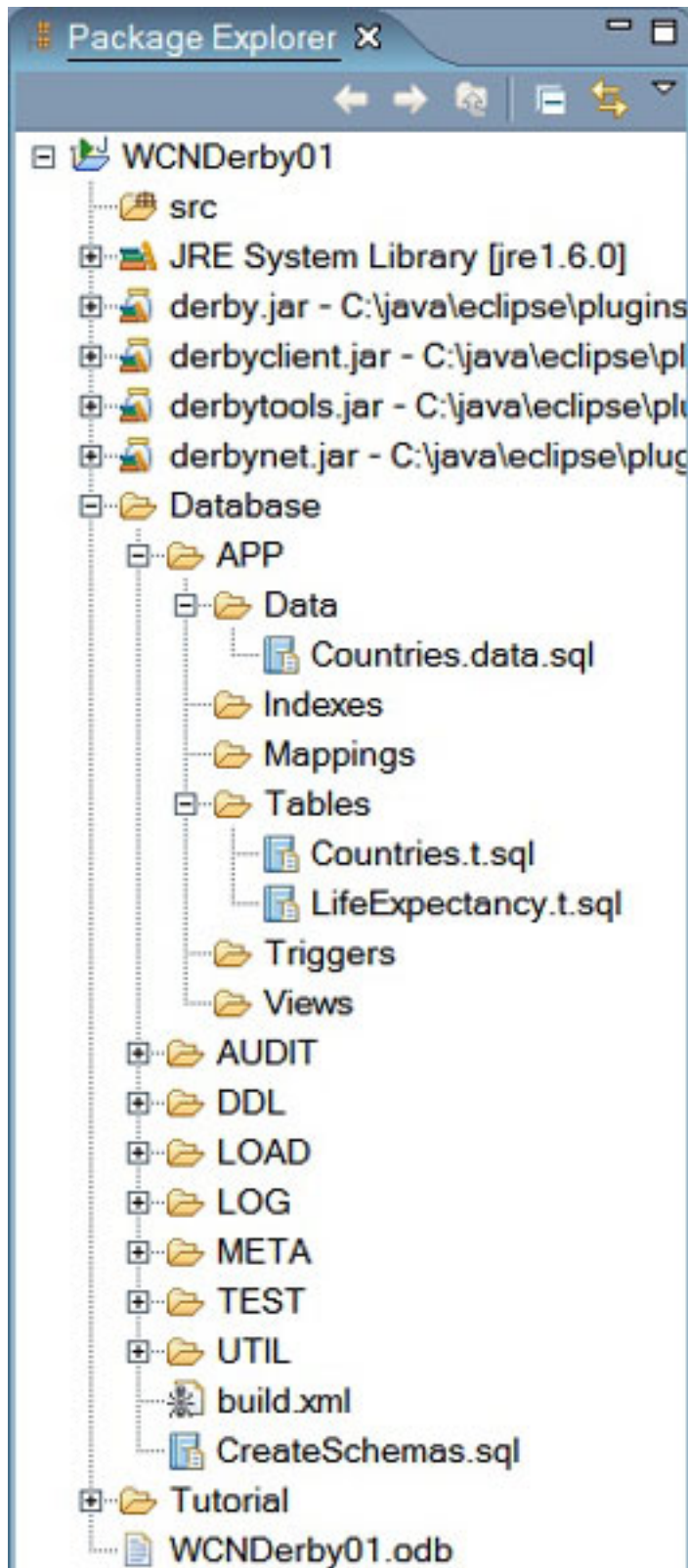
## Section 3. Best practices for getting organized

Set up your folder structure, and divide the database objects by type.

### Set up the partition

It pays to map out the arrangement of your project artifacts before beginning. Just as you would determine the layout of Java code, logically organize the SQL code that you'll create. The arrangement shown in [Figure 6](#) has served me well over the years.

#### **Figure 6. Sample folder layout**



In the Database folder, I've included subfolders detailing several logical partitions.

For Derby, you can implement these partitions as separate schemas; for another relational database management system (RDBMS), you might implement them as separate databases. All this may seem like overkill for a small project, but small projects have a habit of turning into rather large beasts -- best to be prepared from the outset. [Table 1](#) shows the recommended user schemas.

**Table 1. Recommended user schemas for the Database folder**

Schema	Purpose
APP	Contains application data
AUDIT	Holds audit-related database objects
DDL	Objects that allow for the manipulation of Data Definition Language (DDL) tasks
LOAD	Scripts, temporary tables, and so on, for loading data into other schemas
LOG	Objects that capture error or performance data
META	Contains data describing other data
TEST	Contains test data and objects that manipulate test data
UTIL	General objects that other schemas use

## Divide database objects by type

Within each logical partition, subfolders further divide database objects by type, as shown in [Table 2](#). This structure not only allows for ease in finding objects, but also allows you to tailor Ant scripts residing in a particular directory to that object type.

**Table 2. Recommended subfolders for the APP folder**

Subfolder	Purpose
Data	Contains load scripts, <code>insert</code> statements, and raw data
Indexes	Nonprimary and foreign key indexes should be stored here
Mappings	Short scripts that map procedure names to Java methods (Derby specific)
Tables	Remember to separate drop-and-create scripts

Triggers	Self-explanatory
Views	Self-explanatory

---

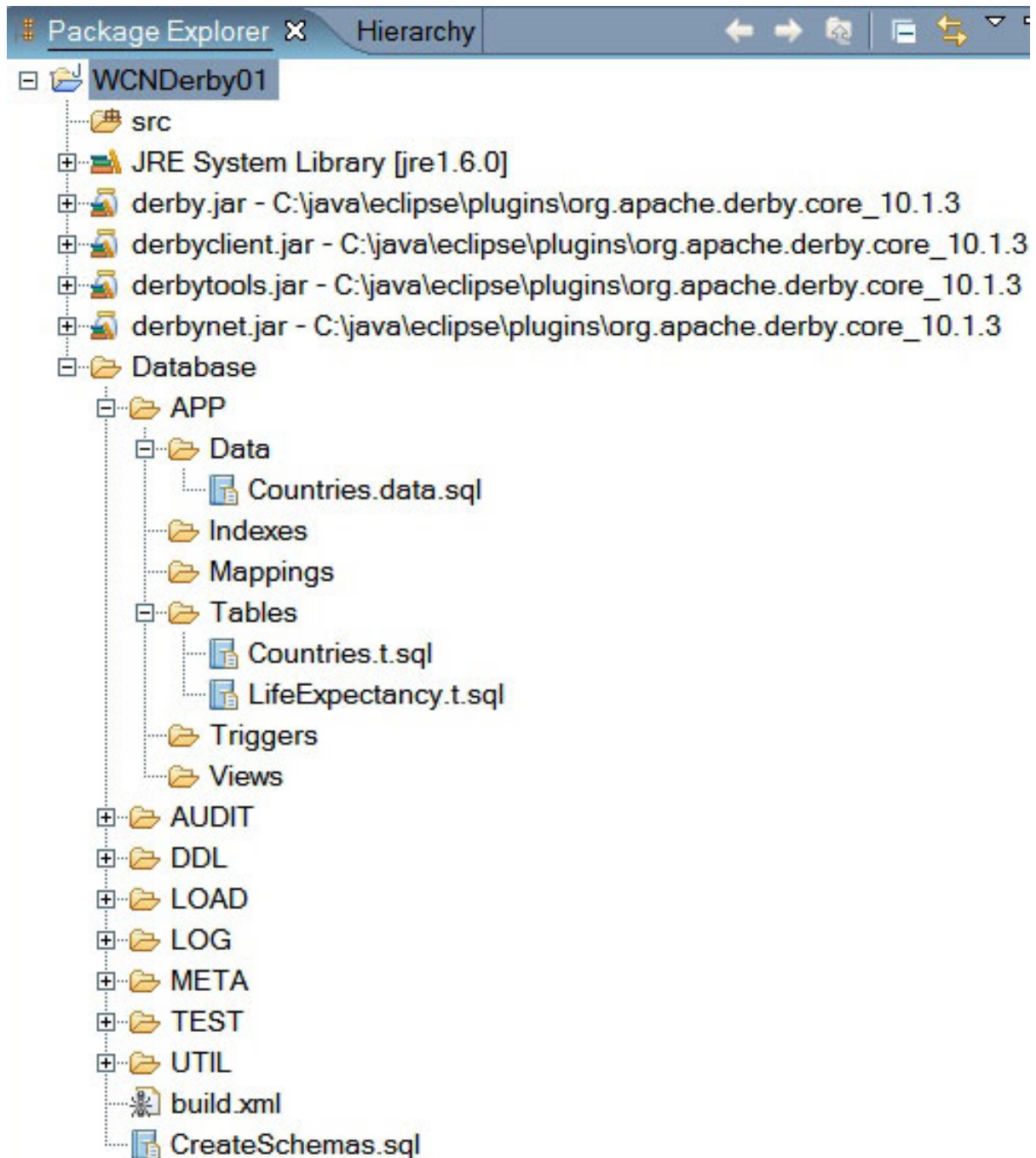
## Section 4. Use Ant to build your database

A major benefit of using a strong integrated development environment (IDE) like Eclipse is the inclusion of a full Ant environment out of the box. In this tutorial, you use Ant to automate the creation and population of your database.

### Download and install the sample database files

Extract the WCN01Database.zip file -- available in the [Download](#) section of this tutorial -- into your project directory. This file provides the example data along with the above file structure. Your project folder should look like [Figure 7](#).

#### **Figure 7. Project folder after extraction**



## Create the database

Find the build.xml file within the top level of the Database folder. Double-clicking this file opens an editor pane in Eclipse that displays the code shown in [Listing 1](#).

### Listing 1. Ant script for database creation

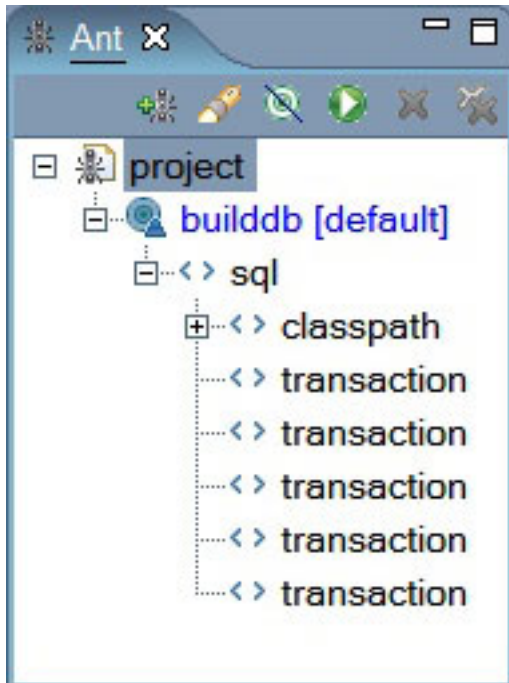
```
<?xml version="1.0" ?>
<project default="builddb">

  <target name="builddb">
    <sql
      driver="org.apache.derby.jdbc.ClientDriver"
      url="jdbc:derby://localhost:1527/WCNDerby01;create=true;"
      userid="derbyuser"
      password="derbyuser"
      onerror="continue" >
      <classpath>
        <path element location="c:/java/lib/derbyclient.jar"/>
      </classpath>
      <transaction src="CreateSchemas.sql"></transaction>
      <transaction>set schema APP;</transaction>
      <transaction src="APP/Tables/Countries.t.sql"></transaction>
      <transaction src="APP/Tables/LifeExpectancy.t.sql"></transaction>
      <transaction src="APP/Data/Countries.data.sql"></transaction>
    </sql>
  </target>
</project>
```

Look at the attributes for the standard Ant tag, `<sql>`. Most of this should be familiar to you if you've worked with Java Database Connectivity (JDBC). For convenience, I'm using the Derby network client; in a production environment, I would switch to the Derby embedded driver to prevent DDL scripts from running during network operations. Also, note the `create=true` option used in the connection URL; this option creates the database if one doesn't already exist but has no effect if the database has already been created in the `derby.system.home` directory you set up earlier.

Make sure that the Derby network server is running: Right-click **build.xml** in the Eclipse Package Explorer view, click **Run As**, and then select the **1 Ant Build** option. Alternately, you can display the Ant view and, using the toolbar buttons for that view, add and run the build.xml file, as shown in [Figure 8](#).

### Figure 8. The Ant view in Eclipse



If you run the script again, nothing damaging happens to the database. It's merely a best practices concept for database administration. Write your DDL scripts so that they can be run repeatedly with no adverse effects. Far too many times, I see DDL scripts that drop a table just before recreating it. Always separate `drop` actions from `create` actions. Also note the `onerror` attribute to the `<sql>` tag; I set this attribute to `continue` during development, and then change it to `stop` when all scripts have stabilized. Doing so allows for incremental development.

Look at the directory that you set earlier as `derby.system.home`. You can now see a subfolder with the database name. Below that are the files associated with the database. The Ant script created all of this.

Next, you use OpenOffice.org's Base program to see what you've wrought.

---

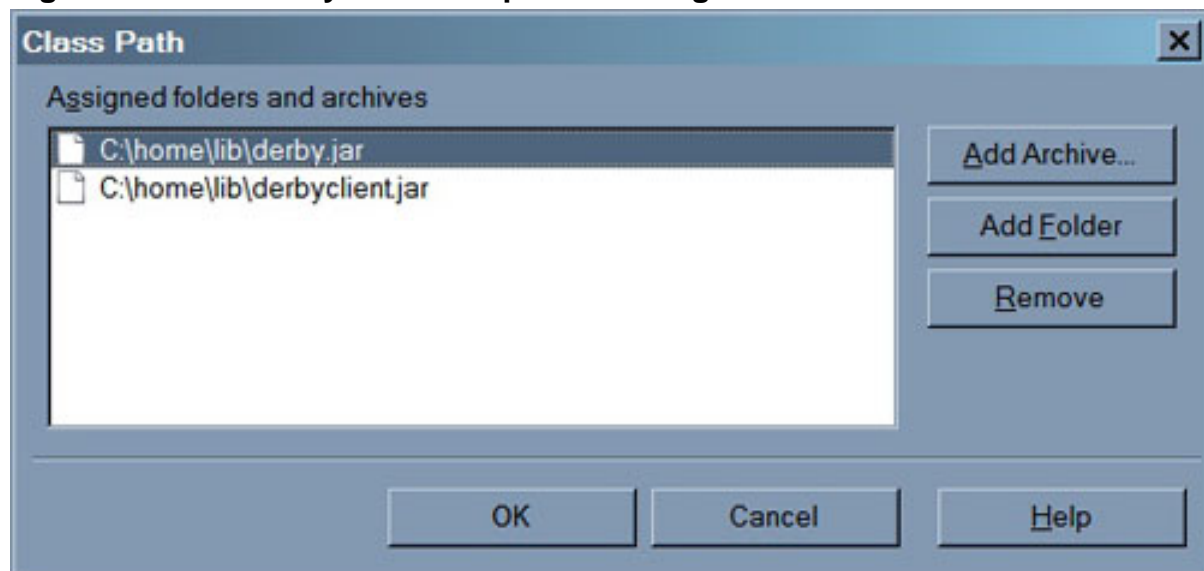
## Section 5. OpenOffice.org Base

With the release of OpenOffice.org, Version 2.0, OpenOffice.org Base became the central point for managing database connectivity with the entire OpenOffice.org suite. First however, a necessary digression to get OpenOffice.org to recognize your Derby `.jar` files.

## Add the Derby .jar files to OpenOffice.org

1. Start OpenOffice.org Calc, and then click **Tools > Options**.
2. In the Options window, choose **Java** under the OpenOffice.org root in the tree view.
3. Click the **Class Path** button to the right, and then add the Derby .jar files in the window that appears. Your entries should look similar to [Figure 9](#).

**Figure 9. Add a Derby class to OpenOffice.org**

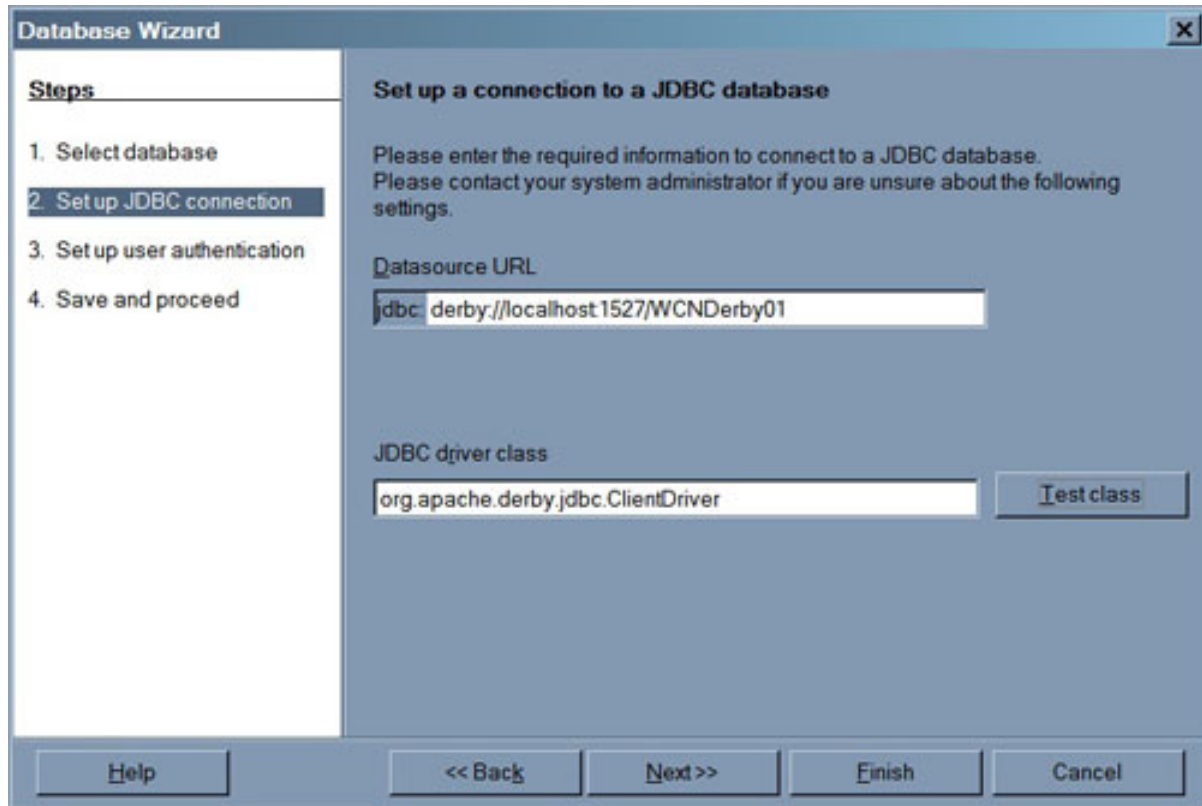


## Connect to your database

Now start OpenOffice.org Base.

1. Select the **Connecting to an existing database** option, and then choose **JDBC** from the list.
2. In the next window, add the driver and URL information used in the Ant script in [Listing 1](#). Your screen should look like [Figure 10](#).

**Figure 10. OpenOffice.org Base JDBC settings**

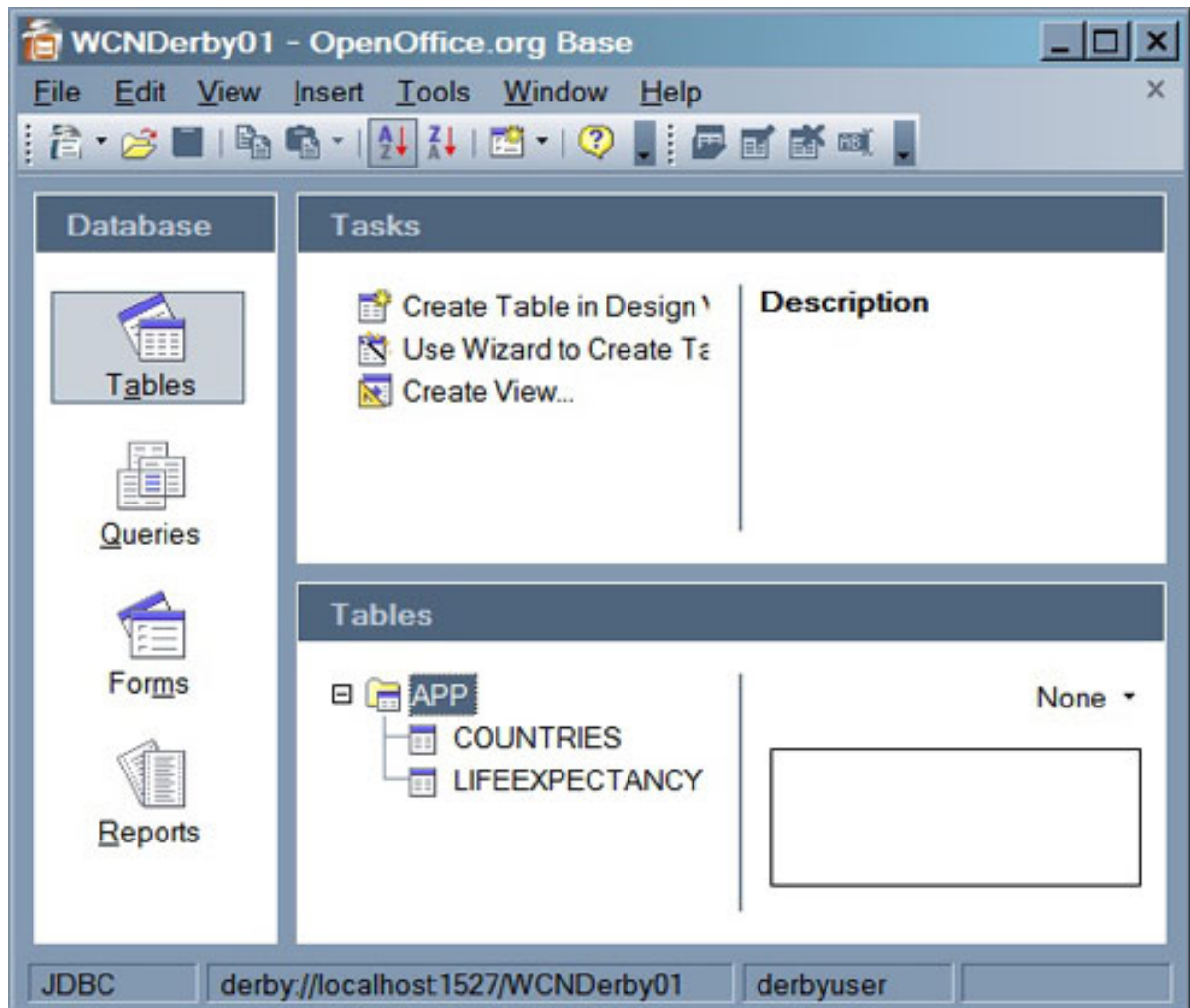


### OpenOffice.org Base

Although this tutorial doesn't examine it closely, OpenOffice.org Base is a powerful system that you can use to add, delete, and modify data and data definitions. It can also create data-entry forms and reports as well as generic queries that you can access from other applications in the suite through the Database Explorer.

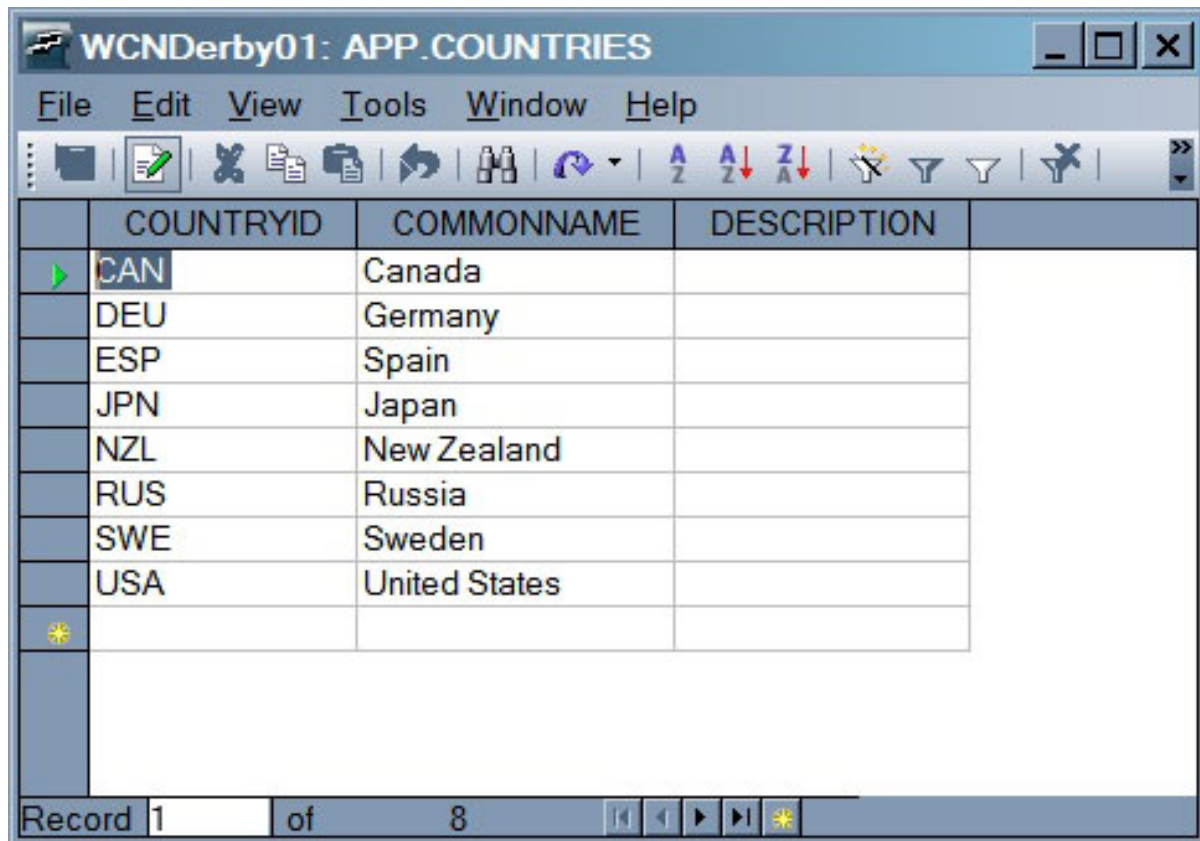
3. Click **Next**, and add your user information.
4. Select the **Password required** check box, and then test the connection after ensuring that the Derby network server is running. If you encounter any problems, verify that the network server is running and that all the settings are correct.
5. Answer **yes** to have OpenOffice.org register the database for you, and save the completed OpenOffice.org Base file in a convenient location.
6. Click **Tables** on the left side of the OpenOffice.org Base main window, and you should see something similar to [Figure 11](#).

**Figure 11. OpenOffice.org Base main window**



In the bottom center, the tables that were created appear beneath the APP schema. Double-clicking the **COUNTRIES** table opens a window that displays the information you loaded earlier through the Ant script, as shown in [Figure 12](#).

**Figure 12. OpenOffice.org Base showing the COUNTRIES table**



	COUNTRYID	COMMONNAME	DESCRIPTION
▶	CAN	Canada	
	DEU	Germany	
	ESP	Spain	
	JPN	Japan	
	NZL	New Zealand	
	RUS	Russia	
	SWE	Sweden	
	USA	United States	
✱			

Record 1 of 8

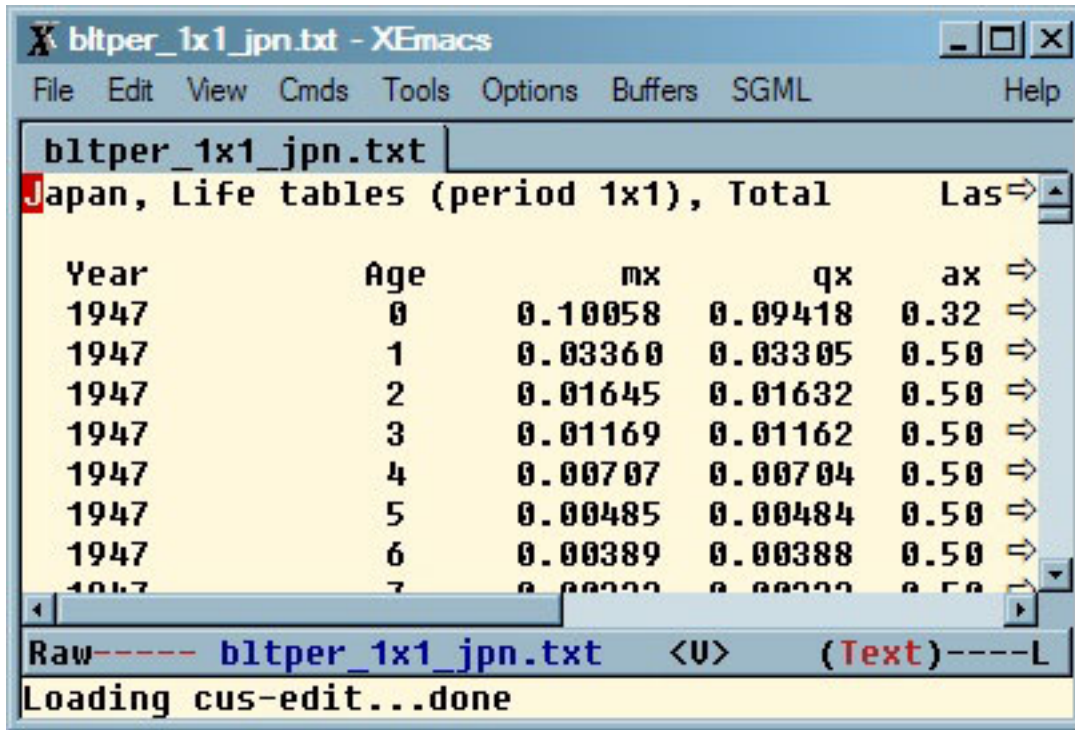
## Section 6. Prepare the raw data for loading

Now you'll load raw data into OpenOffice.org Calc.

### Load your data into OpenOffice.org Calc

To provide a sufficiently large data set for the tutorial's examples, I turned to the Human Mortality Database (HMD) (see [Resources](#) for more information), a database developed jointly by the University of California at Berkeley and the Max Planck Institute for Demographic Research in Rostock, Germany. The life-expectancy rates for several countries were downloaded as fixed-length text files. OpenOffice.org Calc allows for the import of fixed-length data in just a few easy steps, as shown in [Figure 13](#).

**Figure 13. Raw data from the HMD**

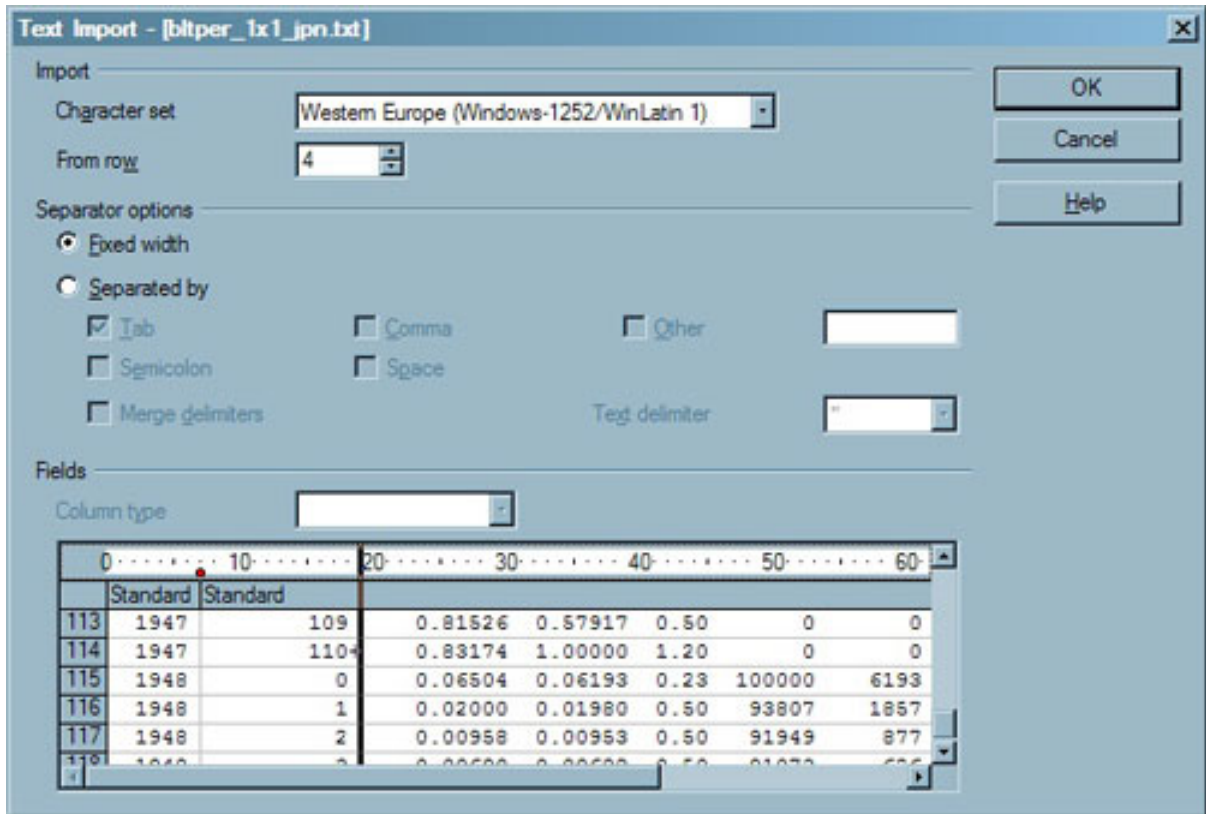


1. To load the data into OpenOffice.org Calc, click **File > Open**.
2. For OpenOffice.org to correctly identify the destination of the text file, configure it to treat the file as **Text CSV**.
3. Be sure to set the type accordingly in the Open File window, or the file will open in OpenOffice.org Writer, instead. In the window that appears, select the data type and where in the file you want OpenOffice.org Calc to start importing.

Figure 14 shows a completed input sheet. The first row to input is 4, and I've specified **Fixed width**. To determine where the data should be separated into columns, OpenOffice.org employs the bottom section of the window. Hover your mouse over the ruler just above the data. A thick black line appears, indicating where you want to place a column. In Figure 14, I've selected to delimit the data at column 7 (note the lighter black line topped with a red point). The large black line at column 19 is the next place to delimit the data.

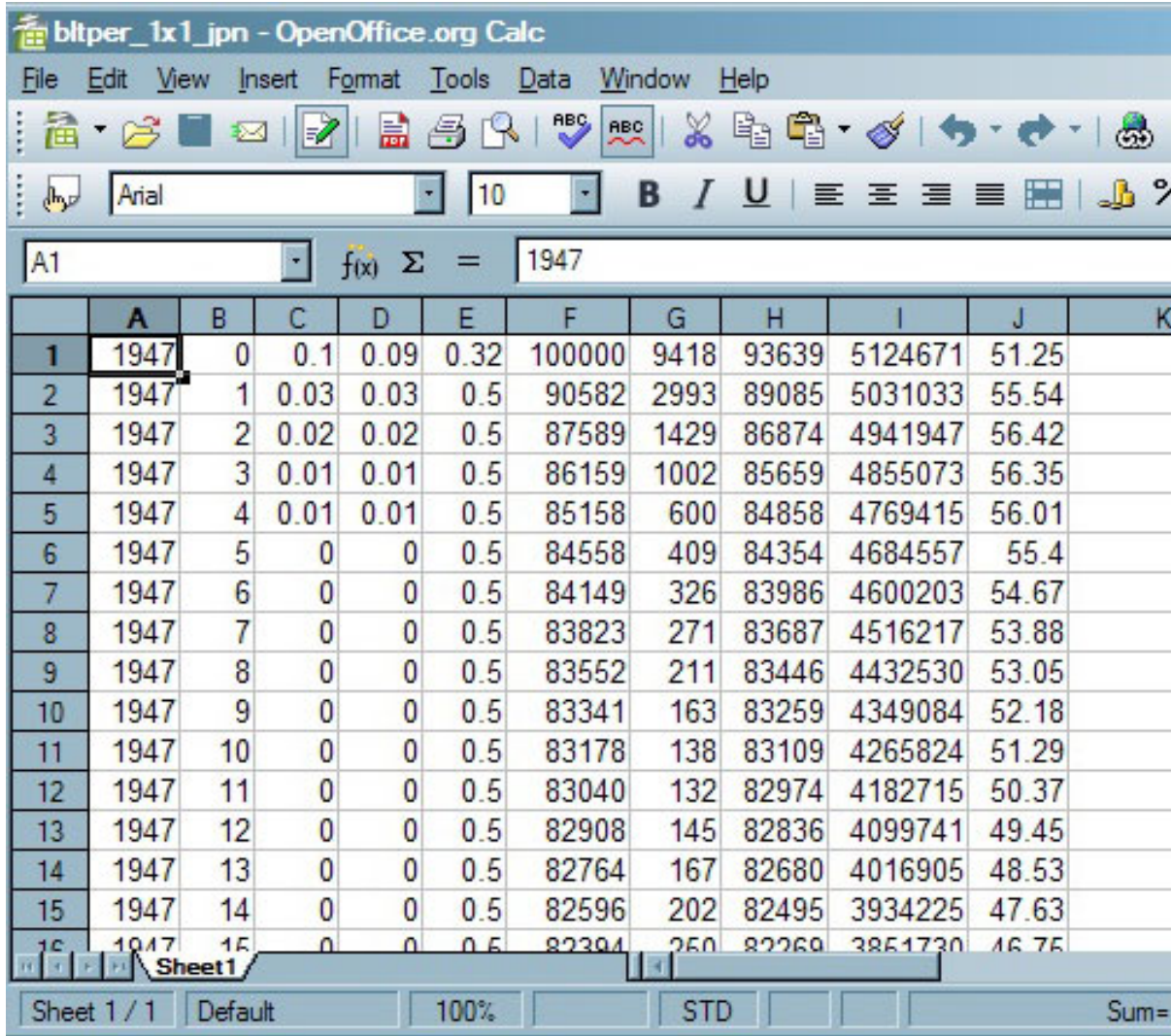
4. Continue delimiting the data, taking care not to slice the data incorrectly. If you place the delimiter in the wrong column, you can adjust it by hovering your mouse over the line while in the ruler. Your cursor turns into a crossbar, and you can move the line.

**Figure 14. Placing delimiters on the raw data**



- When you've finished delimiting the data, click **OK**. The data is brought into OpenOffice.org Calc. Your results should look like [Figure 15](#).

**Figure 15. Raw data in OpenOffice.org Calc**



## Add and populate columns

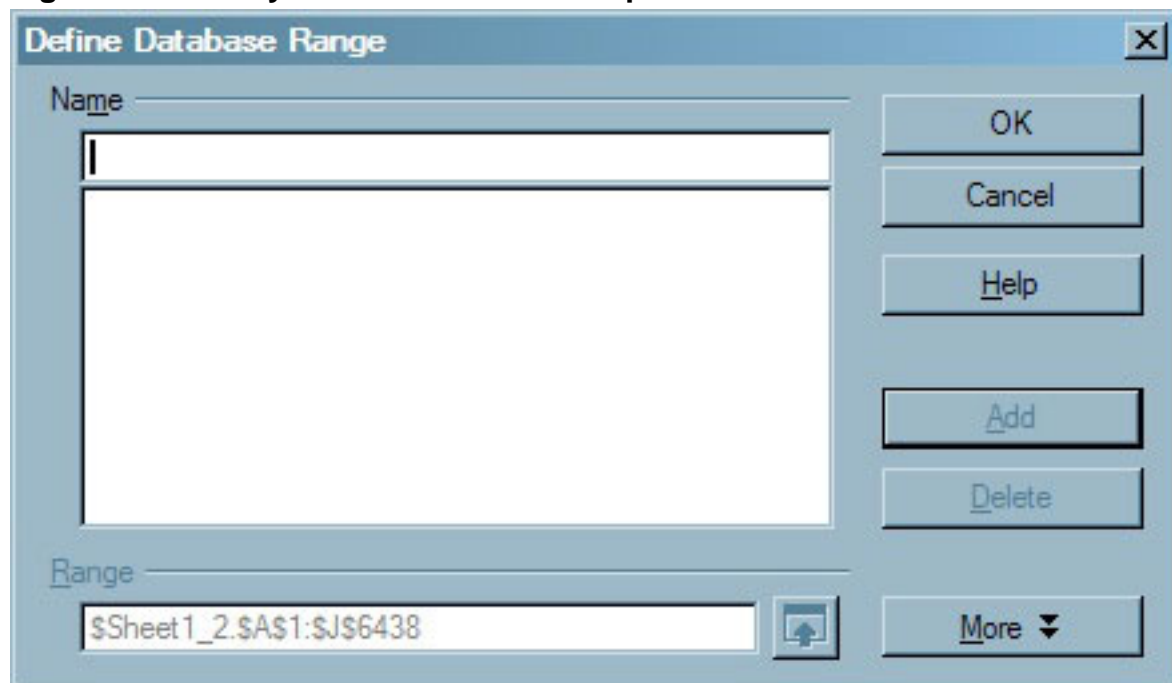
Because the country isn't included in the raw data, the next step is to add a column and populate it with the standard Country ID. You've already seen the identifiers in [Figure 12](#). Adding the Country ID just to the rows of data contained in your raw data is certainly nonintuitive to the casual spreadsheet user. However, the concept of the *range* within OpenOffice.org Calc (and Excel) is extremely flexible and can control actions within the spreadsheet both from an interactive and a programmatic approach. When you get used to this functionality, it's a joy to use.

In the uppermost left corner of the spreadsheet, notice the drop-down list that typically contains either the name or cell location of the currently selected range. What isn't evident is that you can type a range into this box to limit the extent of a subsequent action. Your goal is to add the Country ID just to those rows that you've imported. But how do you know how many rows have been imported? Here's a

shortcut that works every time:

1. Click the cell containing the first row and column of data, then, click **Data > Define Range**.
2. OpenOffice.org Calc automatically defines a range that includes all data up to, but not including, empty rows or columns. You can determine the range by looking at the last number in the Define Database Range window, as shown in [Figure 16](#).

**Figure 16. Quickly find the last row of imported data**



3. Note this number (in this example, 6438), and close this window.
4. Select cell **A1** to clear the range, and then insert a column to the left.
5. Type the Country ID into cell A1, and then move to the Range box.
6. Change the box from **A1** to **A1:A6438**, and then press **Return** to activate the selection.
7. Next click **Edit > Fill > Down**. The Country ID in cell A1 will be repeated down to cell A6438. Before continuing, save your work as a standard OpenOffice.org spreadsheet.
8. Now that you've used OpenOffice.org Calc to reformat the data, you can

save the data for loading into Derby. To do so, click **File > Save As**, and make sure to choose **Text CSV** as the format type.

9. Answer the format warning, and accept the defaults for the comma-delimited format. Your file is now ready to be loaded into the LifeExpectancy table in your Derby database.

---

## Section 7. Bulk-load data into Derby using Eclipse and Ant

In this section, you'll use Eclipse and Ant to load data into your Derby database.

### Use an SQL script to load data into Derby

In the Database/LOAD/Data directory included in the supplied .zip file, there's a comma-delimited file containing approximately 75,000 rows of data showing life-expectancy rates for various countries. The data is a small portion of the HMD. You can load the data using the same method you used earlier to create the database, schema, and tables. The small SQL script shown in [Listing 2](#) calls Derby's load procedure.

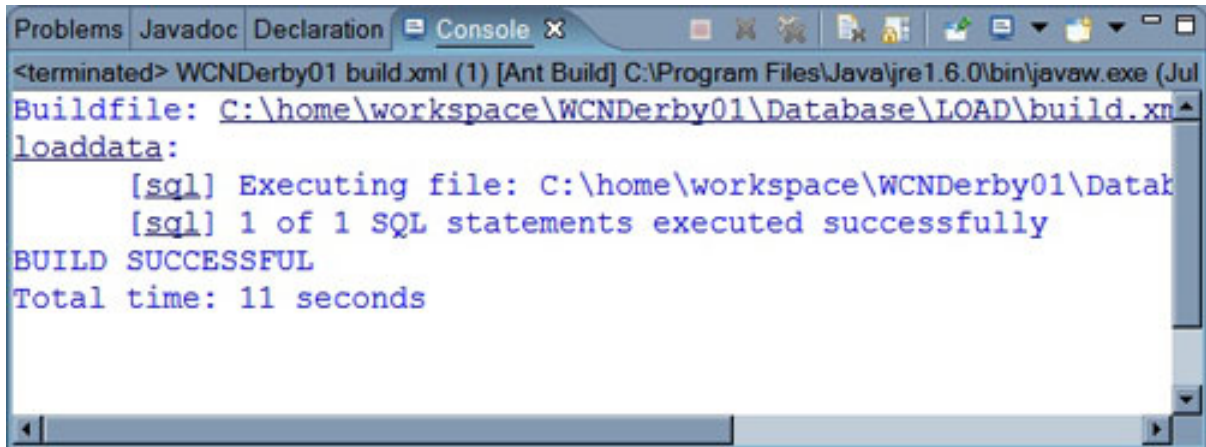
#### Listing 2. Load script for life-expectancy data

```
CALL SYSCS_UTIL.SYSCS_IMPORT_TABLE(  
    'APP',  
    'LIFEEXPECTANCY',  
    'c:/home/workspace/WCNDerby01/Database/LOAD/Data/expectancy.csv',  
    NULL,  
    NULL,  
    NULL,  
    0  
);
```

**Note:** You may need to adjust the full path to the comma-delimited file to match your setup.

Run the supplied build.xml Ant script located in the Database/LOAD directory to load the data. Your output should look similar [Figure 17](#).

#### Figure 17. Output of the Ant load target



```
Problems Javadoc Declaration Console X
<terminated> WCNDerby01 build.xml (1) [Ant Build] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Jul
Buildfile: C:\home\workspace\WCNDerby01\Database\LOAD\build.xml
loaddata:
    [sql] Executing file: C:\home\workspace\WCNDerby01\Datak
    [sql] 1 of 1 SQL statements executed successfully
BUILD SUCCESSFUL
Total time: 11 seconds
```

---

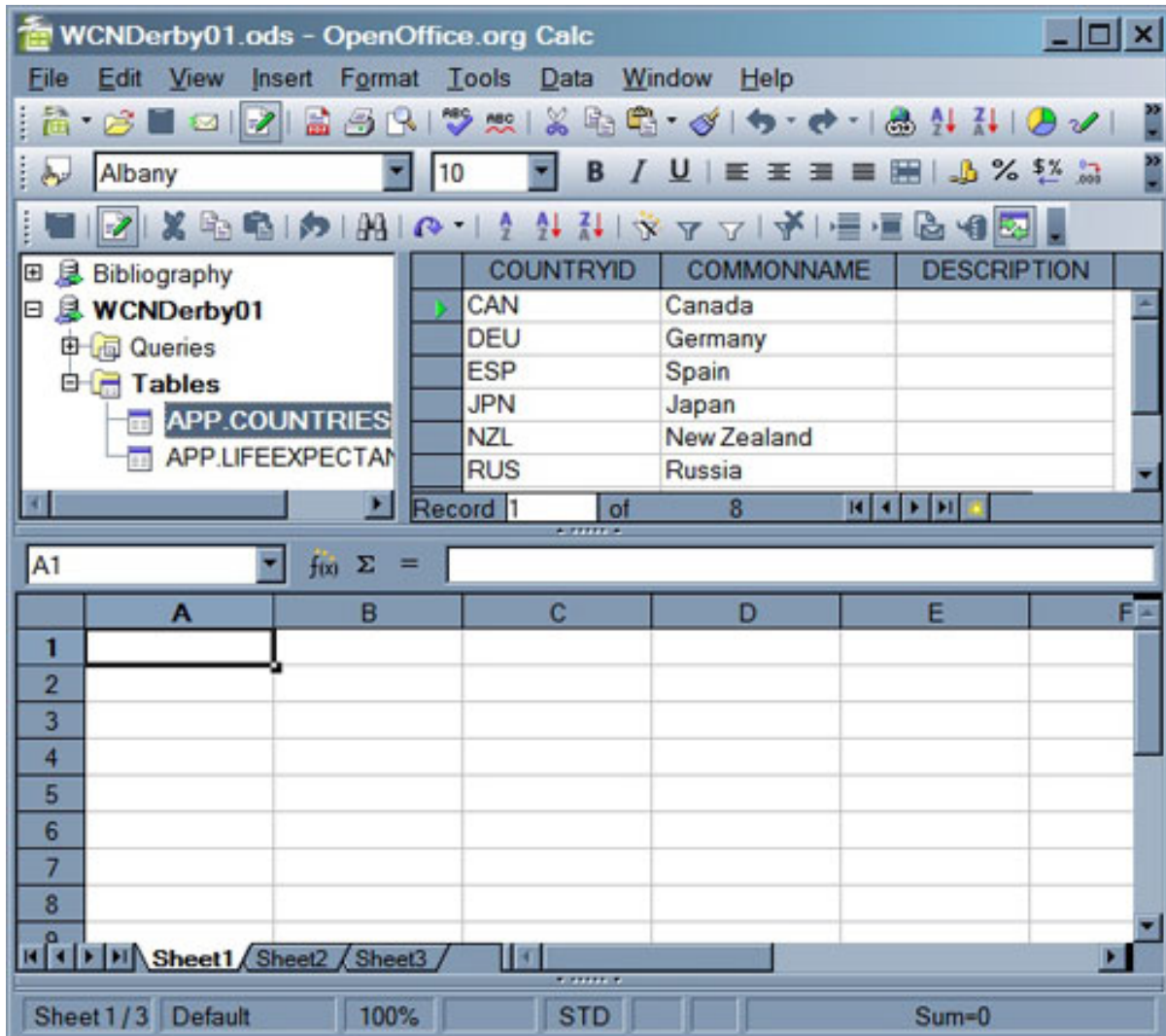
## Section 8. View data from OpenOffice.org Calc

Now that you've laid the groundwork, you can begin analysis.

### View the data from within OpenOffice.org Calc

First, view your data from within OpenOffice.org Calc by bringing up a blank worksheet and pressing **F4**. Your screen should look similar to [Figure 18](#).

#### **Figure 18. Viewing data from OpenOffice.org Calc**



From here, you can drag data into the spreadsheet, sort and filter results, change the format of columns, and choose different tables. But you're still far away from being able to deal efficiently with 75,000 rows of life-expectancy data. Simple things first:

Use the SQL VIEW statement to filter out data on the back end rather than placing the burden on the data consumer. The view in Listing 3 shows just the data for 50-year-old people.

**Listing 3. View-creation script for limiting expectancy data**

```
CONNECT 'jdbc:derby://localhost:1527/WCNDerby01';
CREATE VIEW APP.FIFTYYEAROLDS AS
  SELECT COUNTRYID,
         YR,
         AGE,
         DEATHRATE,
         AVGININTERVAL,
```

```
NRSURVIVORS ,  
NRDEATHS ,  
NRLIVED ,  
NRREMAINING ,  
EXPECTANCY  
FROM APP.LIFEEEXPECTANCY  
WHERE AGE = '50' ;
```

### Using OpenOffice.org queries, instead

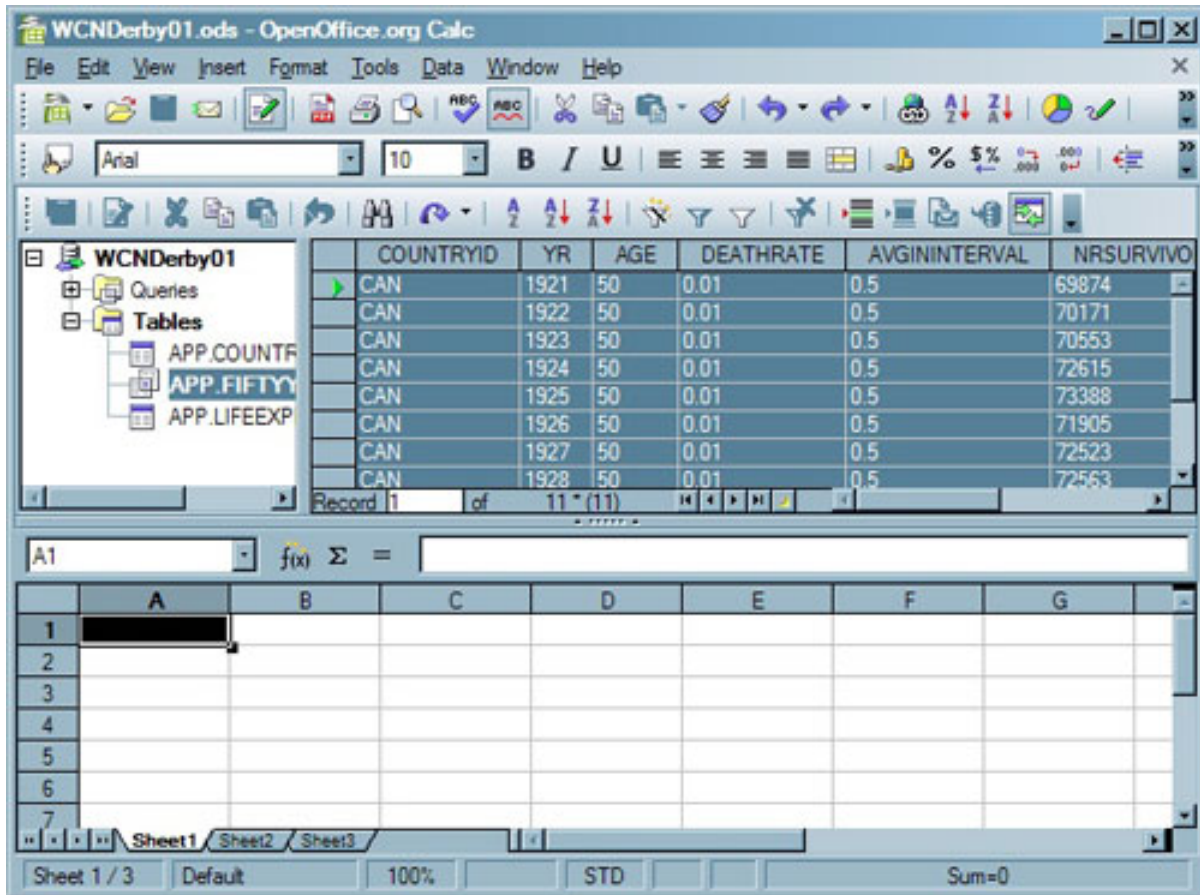
Rather than modifying the view stored in Derby, you could create an OpenOffice.org Base query. The deciding factors should be bulk, portability, and permanency. Large data sets are better filtered within the database. If you're going to use multiple front-end applications for the data or use the view again and again, I recommend using a view stored in Derby. Otherwise, using the OpenOffice.org Base query builder is fine.

Note the `CONNECT` statement at the top of the file. I've included this statement to demonstrate another possible way to run scripts using the Derby UI plug-in:

1. Navigate to the script using the Eclipse Package Explorer view. (The script is in Database/APP/Views.)
2. Right-click the file in the Package Explorer, and point to **Apache Derby**. Hovering over that item automatically displays the option to *Run SQL Script Using 'ij'*.
3. Verify that the network server is running.
4. Click **Run SQL Script Using 'ij'**. The ij tool runs the script, and then terminates itself.
5. Return to OpenOffice.org Calc.
6. Disconnect from the WCNDerby01 database by hovering your mouse over its entry in the Database Explorer and clicking **Disconnect**.
7. Reconnect to the database by expanding the entry again.

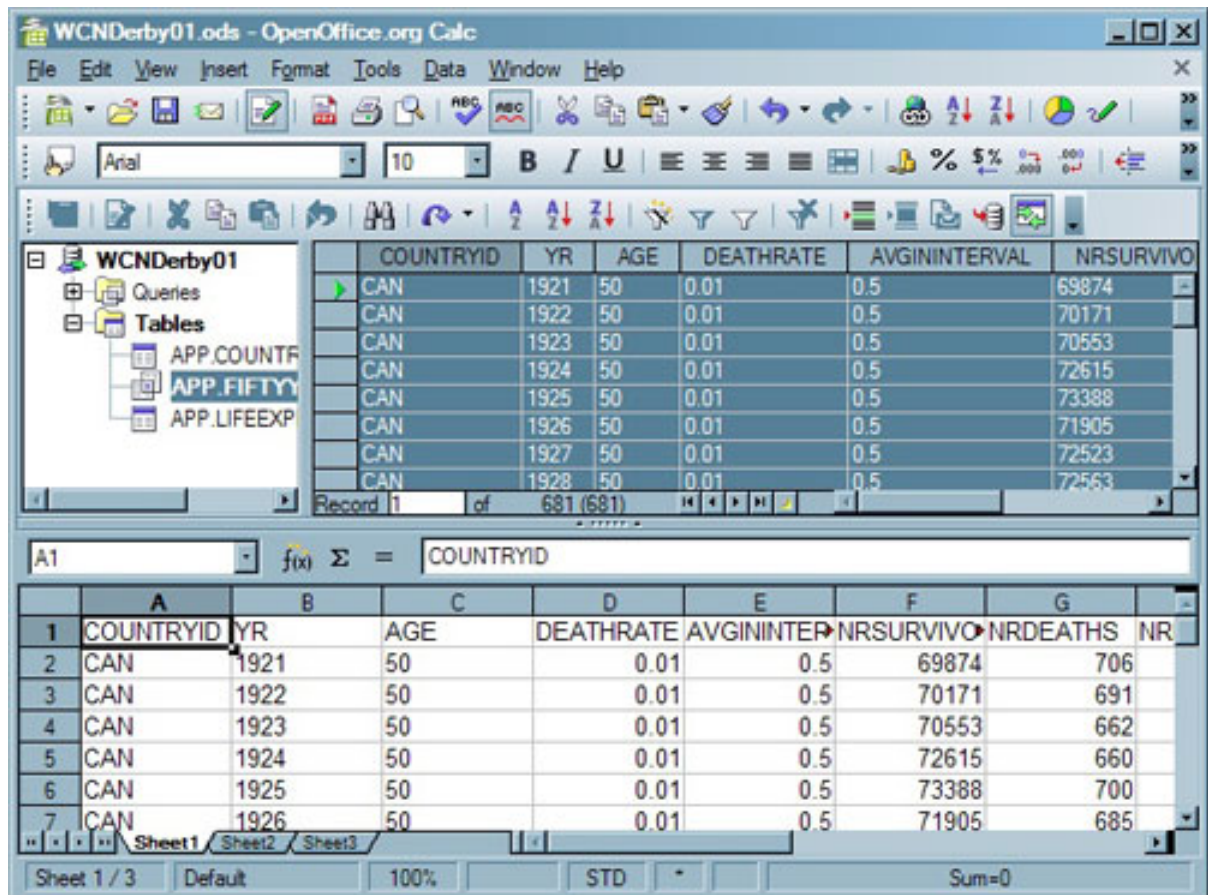
Your new database view appears under Tables, as shown in [Figure 19](#).

### Figure 19. Initial retrieval of the view



1. If you select the view in the Database Explorer, a small subset is loaded into the left pane. Just below the data is a VCR-type control. To retrieve all the data in the view, click the next-to-last button in the control. The count should change from 11 to approximately 681.
2. After you've retrieved all the data, click the empty title cell just to the left of the COUNTRYID title to select all data.
3. Now click once in the blank cells to the left of the COUNTRYID column, and, while holding the button down, drag the data into the spreadsheet below the Database Explorer, as shown in [Figure 20](#).

**Figure 20. The view after dragging in the data**

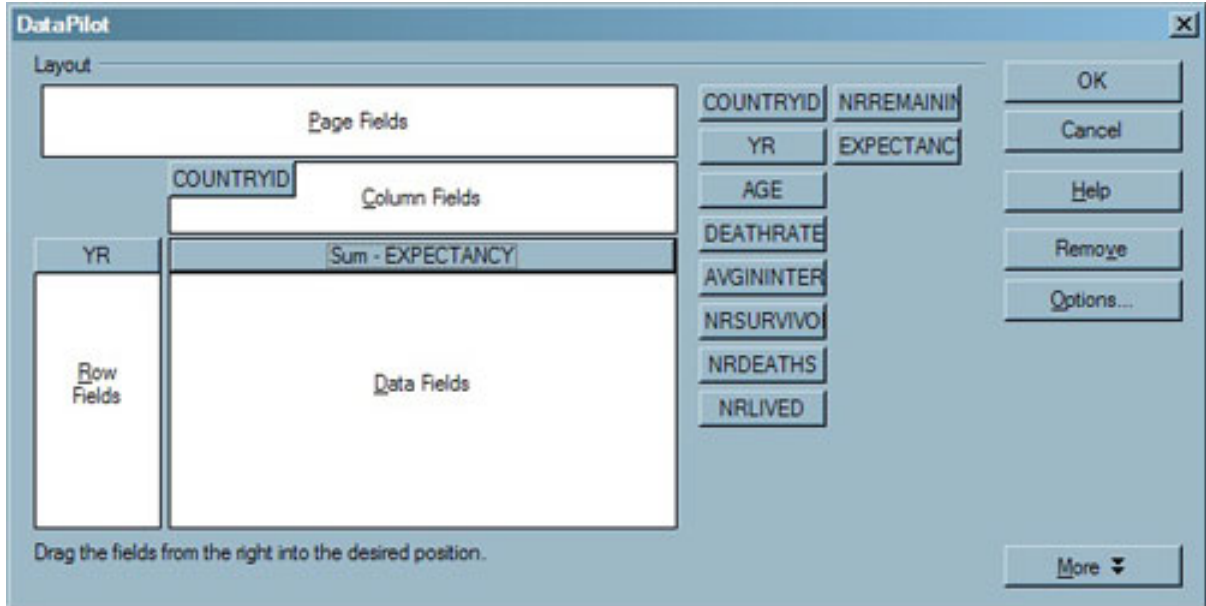


## Analyze the data

On to analyzing the data. The goal is to obtain meaningful results from your view using the power of OpenOffice.org Calc's DataPilot to drive your exploration. To begin analyzing the data, complete these steps:

1. Select all the data contained in the sheet you created earlier (see [Figure 14](#)) by clicking the anchoring cell in the upper left corner.
2. Click **Data > DataPilot > Start** to start the DataPilot Wizard.
3. Click **OK** to choose the current selection for your data source and bring up the layout sheet.
4. Drag and drop the columns display on the right side of the window to create the layout shown in [Figure 21](#).

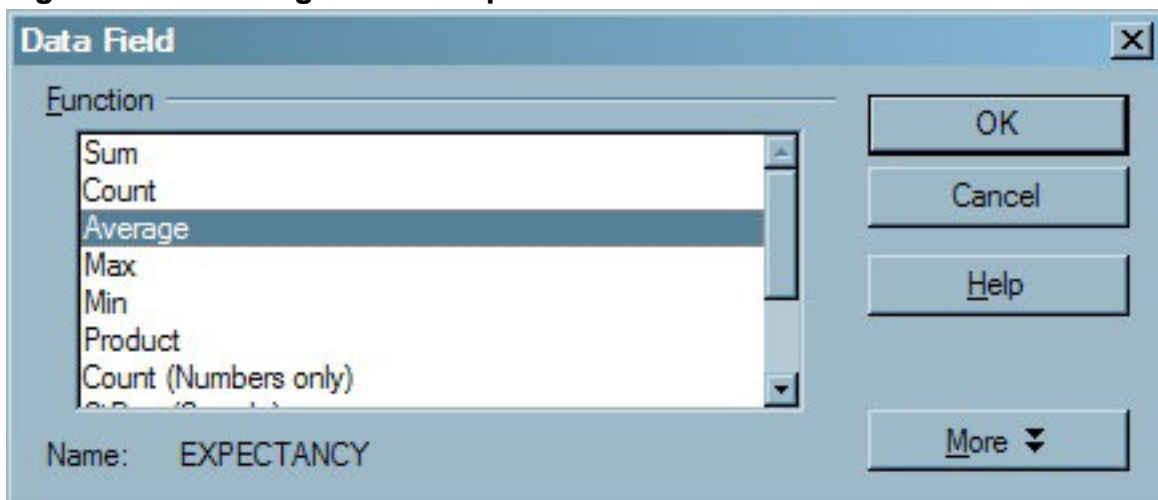
**Figure 21. Initial DataPilot layout**



A couple of problems exist with this initial attempt -- one obvious and one not so obvious.

5. To fix the computation for EXPECTANCY, click **Options**, and then choose **Average** in the Function area of the Data Field window shown in [Figure 22](#).

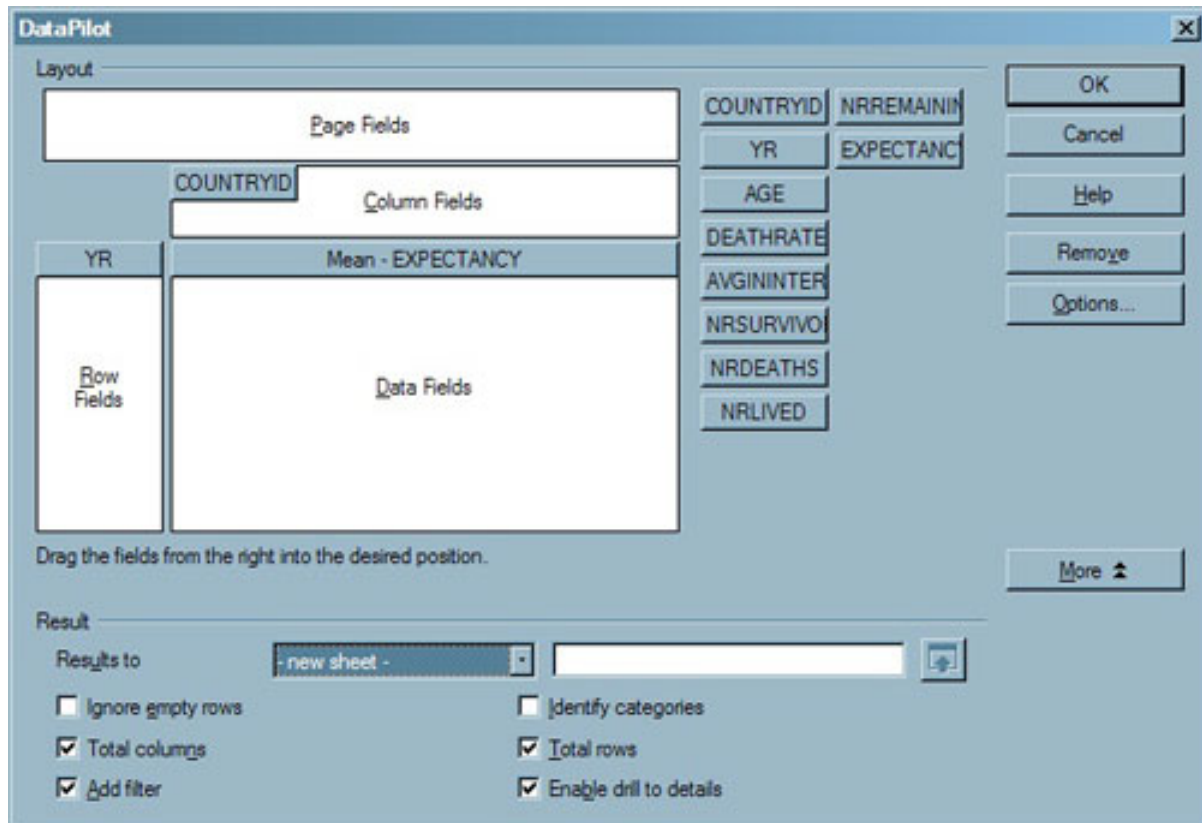
**Figure 22. Choosing a new computation**



The default settings for the DataPilot are to overwrite data in the selected sheet. This is most emphatically what you do *not* want.

6. Click **More** in the layout window, and then choose **new sheet** from the **Results** drop-down list as the target for your results, as shown in [Figure](#)

## 23.

**Figure 23. Selecting the target sheet**

After you click **OK**, the DataPilot populates a new sheet with the initial analysis. Glancing at the data shows that mortality data for Sweden (SWE) goes back to 1751 -- centuries before the other countries in the sample. It isn't until 1959 that we have data for this age across all countries, and the data is complete from that point until 2001.

## Limit your results

Now limit the data set to only that time period where you have full results. You could go back and modify the view you created earlier or filter the data from within OpenOffice.org Calc. In this case, because you've determined the exact parameters of the data, you'll modify the view.

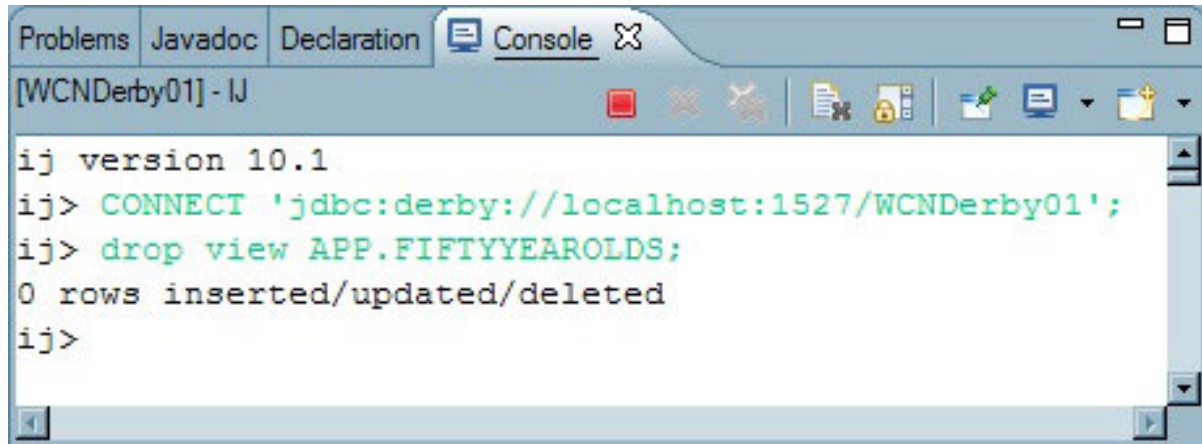
First you must drop the view from the database. Use *ij* interactively for this task:

1. Verify that the network server is running.
2. In Eclipse, right-click the project, and then click **ij (Interactive SQL)** from

the Apache Derby menu.

3. Connect to the database and drop the view, as shown in [Figure 24](#).

**Figure 24. Dropping the view**



[Listing 4](#) shows the modified view-creation script.

**Listing 4. Modified view-creation script**

```

CONNECT 'jdbc:derby://localhost:1527/WCN Derby01';
CREATE VIEW APP.FIFTYYEAROLDS AS
  SELECT COUNTRYID,
         YR,
         AGE,
         DEATHRATE,
         AVGININTERVAL,
         NRSURVIVORS,
         NRDEATHS,
         NRLIVED,
         NRREMAINING,
         EXPECTANCY
  FROM APP.LIFEEXPECTANCY
  WHERE AGE = '50'
        AND (YR > '1958' AND YR < '2002');
  
```

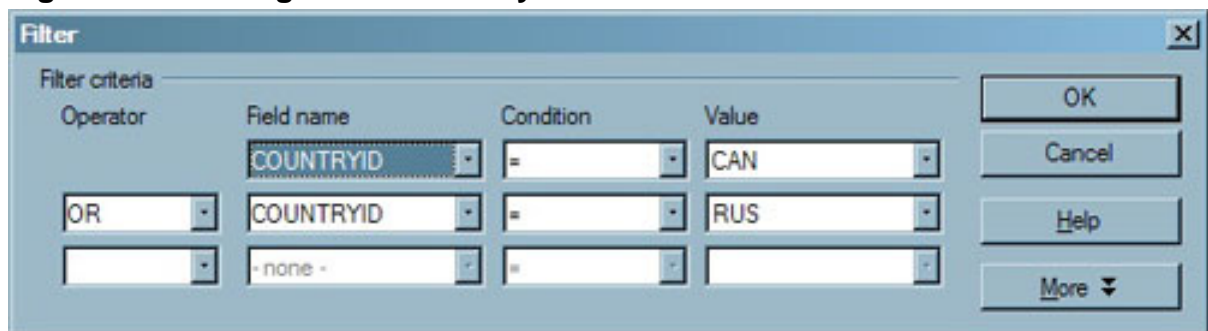
4. To reinstate the view, run the script using ij.
5. Return to OpenOffice.org Calc, right-click the database name in the Database Explorer, and then click **Disconnect**.
6. Reconnect to the database by clicking the plus sign next to the database name, which expands the tree view.
7. Click **APP.FiftyYearOlds** beneath Tables to redisplay the data, with the new view applied.
8. Using the VCR control, bring all the data into OpenOffice.org Calc.

9. Review the data and ensure that you've captured the extent you wanted:
  - a. In the sheet from which you previously dragged the view data, click the uppermost left corner of the spreadsheet to select all data.
  - b. Click **Delete**, and then select the defaults in the window that appears to clear the old data from your spreadsheet.
  - c. Drag and drop the data from the new view into the spreadsheet. The DataPilot should automatically refresh with the new data.

Now, you can start viewing the data interactively from within OpenOffice.org Calc. First, filter the data from within the application. To do so, complete these steps:

1. In cell A1, click the **Filter** button that the DataPilot placed there.
2. Limit the data by selecting the CAN and RUS Country IDs, as shown in [Figure 25](#).

**Figure 25. Filtering the data locally**



3. Click **OK**.

Now only two columns appear. Viewing the data suggests that things are not getting better in Russia as far as life expectancy, but just how bad are things?

1. Select the **E5** cell, type the formula (  $B5-C5$  ), and then press **Enter**.
2. To fill in the rest of the formulae, type  $E5:E48$  in the **Name** box you used earlier to select a range, and then press **Enter**. Doing so selects the range that you want to fill with the formula.
3. Now click **Edit > Fill > Down** to populate all the rows with the formula.

It appears that, from a near-identical life expectancy in 1959, it's now more likely that

individuals in Canada will live almost nine years longer than those in Russia. Remember that this is just speculation and data exploration; much more research (into collection methods, population differences, and so on) would have to be conducted before you could make any firm conclusions.

---

## Section 9. Other ways to access Derby and view data

Discover alternative ways of accessing Derby and viewing your data.

### Alternative methods for accessing Derby from OpenOffice.org

Throughout this tutorial, you've been using the localhost network server method to access Derby. But two other methods work as well. The first is a variation on the network server. By replacing localhost with the Domain Name System (DNS) entry for the computer, you can access the Derby database from other computers. (I wouldn't recommend doing this unless you implement a strong authorization/authentication system for your database.) Using this method, you can centrally locate the database along with the OpenOffice.org Base and Calc files necessary to access the database.

However, if, as in this tutorial, you're working with relatively static data and performing analysis, it's best to run Derby in embedded mode. Doing so allows you to fully integrate Derby with OpenOffice.org and releases you from the requirement of having the network server running at all times. Although only one JVM can access the database at a time when running in embedded mode, the good news is that OpenOffice.org uses only one JVM during execution, so you can still access the database from multiple documents and applications. For your purposes, the gains in portability and ease of use far outweigh the disadvantages.

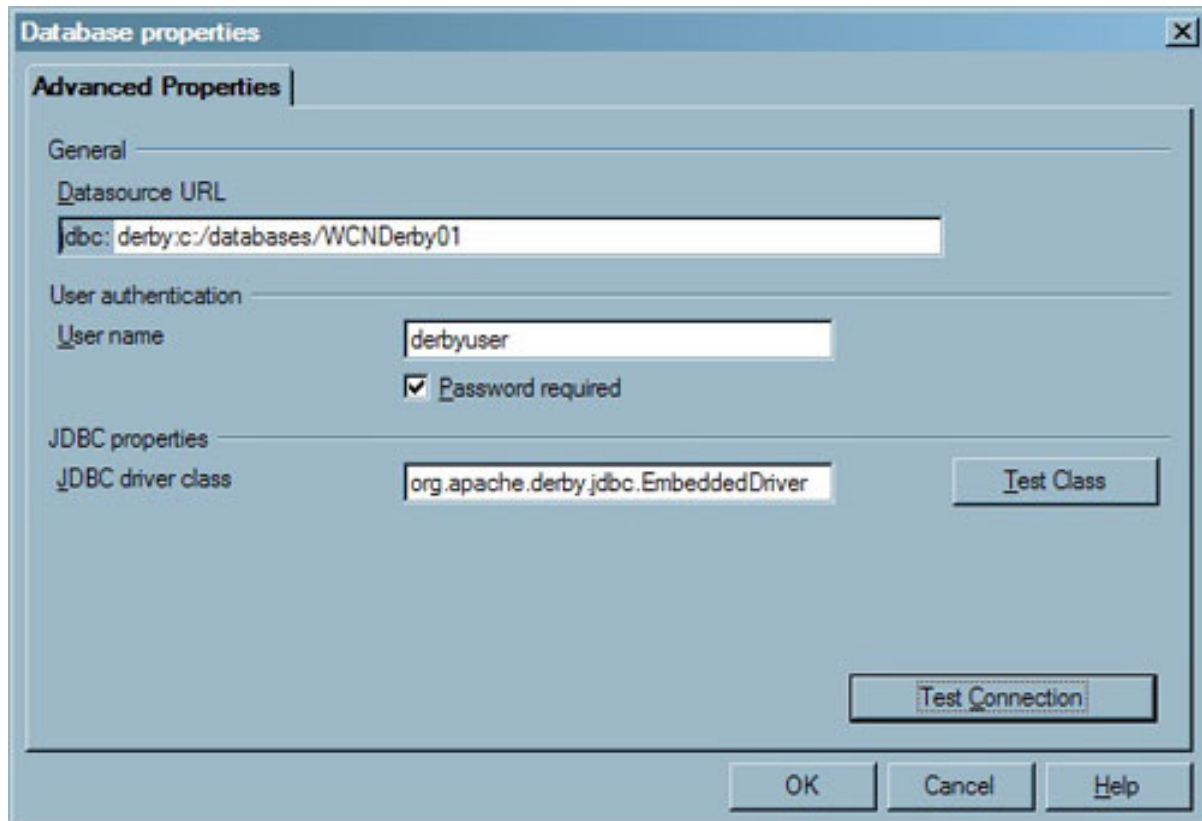
#### Use OpenOffice.org Base in embedded mode

To switch OpenOffice.org Base to use the database in embedded mode, complete these steps:

1. Start OpenOffice.org Base.
2. Click **Edit > Database > Properties**.
3. Replace the localhost and port in the connection string with the location of the file (in your case, C:/databases/WCNDerby01).

4. Change the driver (located in the Derby.jar file you added to the OpenOffice.org classpath variable earlier) to `org.apache.derby.jdbc.EmbeddedDriver`. Your entry should look like that shown in [Figure 26](#).

**Figure 26. Changing to embedded Derby**

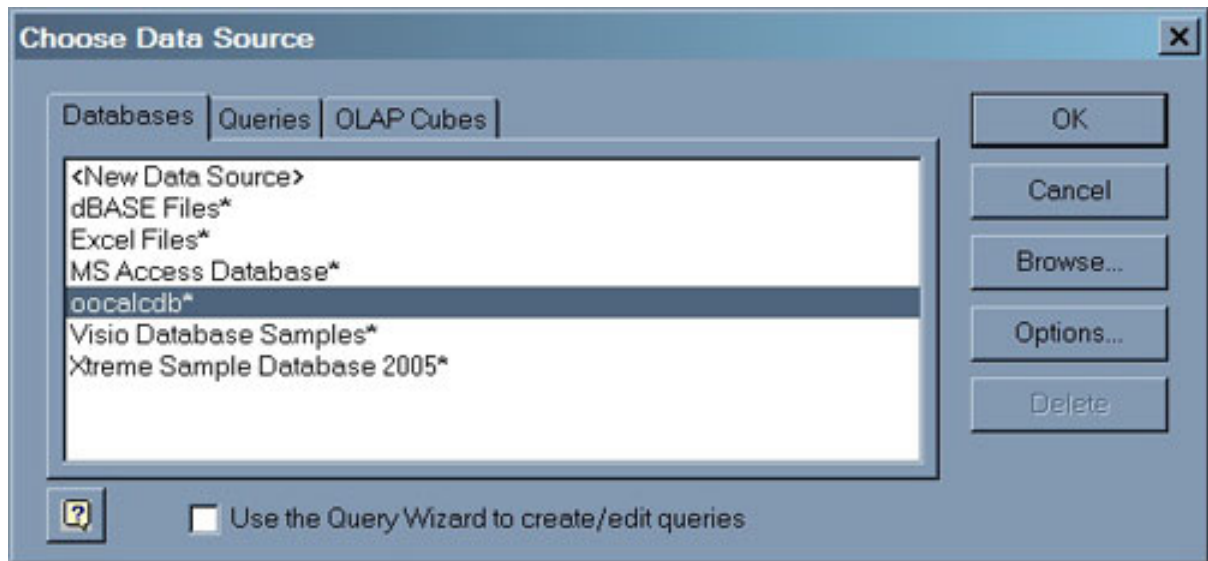


## View data from Microsoft Excel

If you're using Excel, Derby can fit in with your plans. Before you can connect to Derby from Excel, you must install several components. (See [Resources](#) for details.) After following the instructions for setting up Derby Open Database Connectivity (ODBC), you can import the data directly into Excel. To do so, complete these steps:

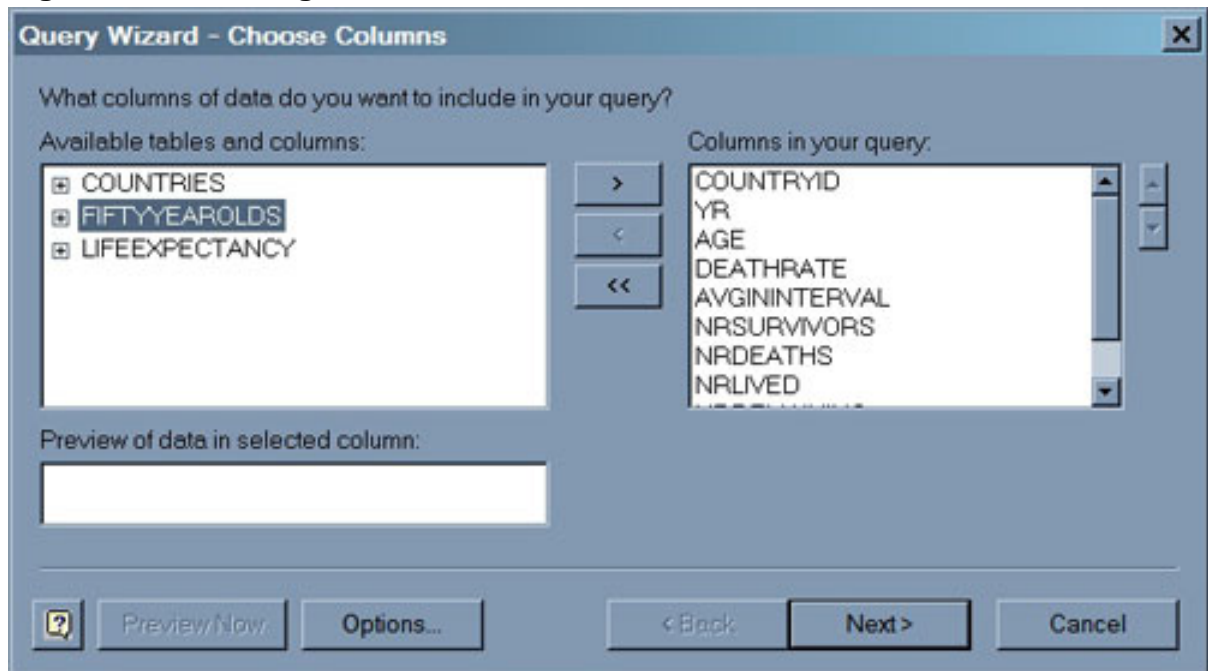
1. Using a blank workbook, click **Data > Import External Data > New Database Query**.
2. In the Choose Data Source window, choose the ODBC DSN (that is, **ooalcdb**) that you created earlier, as shown in [Figure 27](#).

**Figure 27. Choosing the ODBC DSN**



3. Choose the view you created earlier, and select to retrieve all the columns, as shown in [Figure 28](#).

**Figure 28. Choosing the view**



The result is returned directly to a worksheet, where you can use Excel tools to analyze it. [Figure 29](#) shows the data in Excel.

**Figure 29. Viewing the data in Excel**

Microsoft Excel - oocalcdb.xls

File Edit View Insert Format Tools Data Window Help

Arial 10 B I U

A1

	A	B	C	D	E	F
1	COUNTRYID	YR	AGE	DEATHRATE	AVGININTERVAL	NRSURV
2	CAN	1921	50	0.01016	0.5	
3	CAN	1922	50	0.0099	0.5	
4	CAN	1923	50	0.00943	0.5	
5	CAN	1924	50	0.00913	0.5	
6	CAN	1925	50	0.00958	0.5	
7	CAN	1926	50	0.00957	0.5	
8	CAN	1927	50	0.00895	0.5	
9	CAN	1928	50	0.00965	0.5	
10	CAN	1929	50	0.00969	0.5	
11	CAN	1930	50	0.00991	0.5	
12	CAN	1931	50	0.00948	0.5	

Sheet1 Sheet2 Sheet3

Ready

Now, you're free to use the many powerful analysis tools available in Excel. (Note that embedded mode is not available to you when accessing the data from Excel.)

## Section 10. Summary

You've seen how Derby can be a portable, robust database component for data analysis. Taking a data set that you cannot easily use within OpenOffice.org Calc or Microsoft Excel, you were able to filter it using standard, time-tested SQL. Because Derby is standards based, you can use numerous tools to view, extract, and manipulate your data. Stay tuned for the next installment of this series, Part 2, in which you'll use Derby to replace most of the need for Asynchronous JavaScript + XML (Ajax).

## Downloads

Description	Name	Size	Download method
Example data and code	WCN01Database.zip	1638KB	<a href="#">HTTP</a>

[Information about download methods](#)

# Resources

## Learn

- Visit the official [Apache Derby](#) Web site to download the software and Eclipse plug-ins.
- Check out the [Apache Derby wiki](#), an excellent place to find tips and tricks. Plus, you can contribute!
- Check out the [developerWorks Apache Derby project area](#) for articles, tutorials, and other resources to help you get started with Derby.
- Read "[ODBC programming using Apache Derby](#)" (developerWorks, September 2004) for step-by-step procedures to set up ODBC against the Derby network server using the freely downloadable IBM® DB2® Universal Database™ Runtime Client.
- Visit the [Human Mortality Database](#), which provides a wealth of raw and statistically adjusted data of interest to both researchers and laymen alike. A minute portion of the data was used as example data for this project. (Human Mortality Database: University of California, Berkeley [USA], and Max Planck Institute for Demographic Research [Germany]. Available at <http://www.mortality.org> or [www.humanmortality.de](http://www.humanmortality.de) [data downloaded on 7 July 2006].)
- Visit the [developerWorks Open source zone](#) for extensive how-to information, tools, and project updates to help you develop with open source technologies and use them with IBM's products.
- Browse all the [Apache articles](#) and [free Apache tutorials](#) available in the developerWorks Open source zone.
- Browse for books on these and other technical topics at the [Safari bookstore](#).
- Learn more about the [IBM Cloudscape™ database](#), which is built using the Apache Derby code base.

## Get products and technologies

- Download the latest version of the [Eclipse Java SDK](#).
- Add the many features available for Eclipse's [Callisto Simultaneous Release](#) onto Eclipse.
- Discover the powerful [OpenOffice.org](#) suite for your operating system.
- Check out [Microsoft Excel](#) spreadsheet software, a core part of Microsoft Office.
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

## Discuss

- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

## About the author

### Dave Warner

Dave Warner has been working with databases since the early 1980s, focusing on business productivity and analysis. A Sun Certified Java Programmer, he's also certified in Microsoft SQL Server and works with Sybase tools extensively. He was formerly a chief technology officer at a small medical software company and presently works at Northrop Grumman IT Solutions as a senior database administrator.

## Trademarks

Cloudscape, DB2, DB2 Universal Database, and IBM are registered trademarks of IBM in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Excel are trademarks of Microsoft Corporation in the United States, other countries, or both.