User Manual for datatooltk version 0.5b

Nicola L. C. Talbot
www.dickimaw-books.com

2013-07-11

Contents

1 Introduction

The LaTeX datatool package is able to save databases in its own internal format to allow for rapid loading (using `\DTLsaverawdb` or `\DTLprotectedsaverawdb`). Files in this format are difficult to edit and only a TeXpert should attempt it, but they are by far the fastest way of loading a datatool database in LaTeX. This application provides a graphical user interface (GUI) making it easier to edit these files. It can also import data from comma-separated values (CSV) files, from structured query language (SQL) databases and from probsoln databases. This manual assumes the user has some knowledge of the datatool package. Please ensure you have at least version 2.15 of datatool installed in your TeX distribution.

The `datatooltk` application can run in either batch mode (default) or GUI mode (see chapter 2). Command line invocation:

`datatooltk` [⟨*options*⟩]

Available options:

`--gui` **(or** `-g`**)** Invoke `datatooltk` in GUI mode. (The command line invocation

`datatooltk-gui` [⟨*options*⟩]

is equivalent to

`datatooltk --gui` [⟨*options*⟩]

but additionally has a splash screen.)

`--batch` **(or** `-b`**)** Invoke `datatooltk` in batch mode (default).

`--out` ⟨***filename***⟩ **(or** `-o` ⟨***filename***⟩**)** Save the database to ⟨*filename*⟩ (batch mode only).

`--in` **(or** `-i`**)** ⟨***datatool file***⟩ Load ⟨*datatool file*⟩. The switch `--in` (or `-i`) is optional, so `datatooltk` ⟨*file*⟩ is equivalent to `datatooltk --in` ⟨*file*⟩.

`--name` ⟨***name***⟩ If used with `--in`, `--csv`, `--sql` or `--probsoln`, sets the database label to ⟨*name*⟩. (See section 1.2.)

`--version` **(or** `-v`**)** Print the version details to STDOUT and exit.

`--help` **(or** `-h`**)** Print a brief summary of available options to STDOUT and exit.

`--debug` Enable debug mode.

`--nodebug` Disable debug mode. (Default.)

`--delete-tmp-files` Delete temporary files on exit. (Default.)

`--nodelete-tmp-files` Don't delete temporary files on exit.

`--map-tex-specials` Map TEX special characters when importing data from CSV or SQL.

`--nomap-tex-specials` Don't map TEX special characters when importing data from CSV or SQL. (Default.)

`--seed` ⟨***number***⟩ Set the random generator seed to ⟨*number*⟩ or clear it if ⟨*number*⟩ is `""`. (See section 3.2.)

`--shuffle-iterations` ⟨***number***⟩ Set the number of iterations to perform in a shuffle to ⟨*number*⟩. (See section 3.2.)

`--shuffle` Shuffle the database. (Shuffle is always performed after sort, regardless of the option order.)

`--noshuffle` Don't shuffle the database. (Default.)

`--sort` **[**⟨***prefix***⟩**]**⟨***field***⟩ Sort the database according to the column whose label is ⟨*field*⟩. Optionally, ⟨*prefix*⟩ may be `+` (ascending order) or `-` (descending order). If ⟨*prefix*⟩ is omitted, ascending is assumed. (See section 3.1.)

`--sort-case-sensitive` Use case-sensitive comparison when sorting strings.

`--sort-case-insensitive` Use case-insensitive comparison when sorting strings. (Default.)

`--csv` ⟨***csv file***⟩ Import data from the given CSV file. (See section 4.1)

`--sep` ⟨***character***⟩ Specify the character used to separate values in the CSV file. (Defaults to a comma)

`--delim` ⟨***character***⟩ Specify the character used to delimit values in the CSV file. (Defaults to a double quote)

`--csvheader` The CSV file has a header row. (Default.)

`--nocsvheader` The CSV file doesn't have a header row.

`--sql` ⟨***statement***⟩ Import data from an SQL database where ⟨*statement*⟩ is an SQL SELECT statement. (See section 4.2)

`--sqldb` ⟨***name***⟩ The SQL database name.

`--sqlprefix` ⟨***prefix***⟩ The Java SQL prefix. (Default: "`jdbc:mysql://`".) Currently, only MySQL is supported. Additional libraries will be required for other SQL databases.

`--sqlport` ⟨***port***⟩ The SQL port number. (Default: 3306.)

`--sqlhost` ⟨***host***⟩ The SQL host. (Default: "`localhost`".)

`--sqluser` ⟨***user name***⟩ The SQL user name.

`--sqlpassword` ⟨***password***⟩ The SQL password (insecure). If omitted, you will be prompted for a password if you try to import data from an SQL database.

`--wipepassword` For extra security, wipe the password from memory as soon as it has been used to connect to an SQL database. (Default.)

`--nowipepassword` Don't wipe the password from memory as soon as it has been used to connect to an SQL database.

`--probsoln` ⟨***filename***⟩ Import probsoln data from ⟨*filename*⟩. (See section 4.3.)

You can't combine any of the following options: `--in`, `--csv`, `--sql`, `--probsoln`.

1.1 What it isn't

The `datatooltk` application isn't intended to have the full functionality of a spreadsheet. Its purpose is to allow you to edit datatool databases with multilined entries. If your data just consists of numbers or short single-lined text, then you'll probably be better off just using a spreadsheet to input the data and use `datatooltk` in batch mode to convert from CSV to a datatool file.

1.2 File Extensions

The datatool database files loaded and saved by `datatooltk` are just LaTeX files, so they could simply have the standard `.tex` extension, but to help differentiate the database files from other files containing TeX/LaTeX code (such as picture-drawing code), `datatooltk` assumes a default extension of `.dbtex`. If you use this extension, remember to include it in the argument of `\input`. Note that the database label (as used in commands like `\DTLnewdb`) is independent of the file name (although when importing data, it defaults to the file base name). The database label can be changed using Edit→Edit Database Name in GUI mode or via the command line option `--name` ⟨*label*⟩.

Example 1.
Suppose you have a database file called `my-data.dbtex` and you have set the database label to just "`data`" (as shown in Figure 1.1). Then you can load and display the data using:

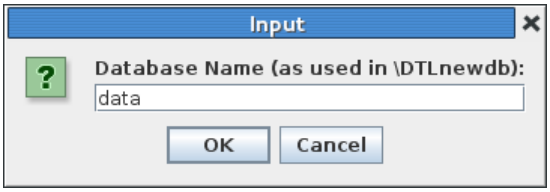

Figure 1.1: Setting the Database Name

```
\documentclass{article}
\usepackage{datatool}% remember to load the datatool package

\input{my-data.dbtex}% load the database from file 'my-data.dbtex'

\begin{document}

\DTLdisplaydb{data}% Display the database identified by the name 'data'

\end{document}
```

If you can't remember the name you assigned to the database, you can access it using `\dtllastloadeddb`.

```
\documentclass{article}
\usepackage{datatool}% remember to load the datatool package

\input{my-data.dbtex}% load the database from file 'my-data.dbtex'

\begin{document}

\DTLdisplaydb{\dtllastloadeddb}% Display the last loaded database

\end{document}
```

1.3 Verbatim

Since the contents of the database are stored in a TEX token register, and assigned to control sequences via commands like `\DTLforeach`, verbatim text is not permitted. This is a common problem when attempting to use verbatim text within a command and is covered in the UK List of TEX Frequently Asked Questions (Why doesn't verbatim

work within...?). The `datatooltk` application checks for verbatim text[1] when you load a database or import data (unless the "map TEX special characters" property is set for CSV or SQL imports). Also, `datatooltk` checks for verbatim text when you edit the contents of a cell. If it detects any, it will give a warning. If you ignore the warning, TEX will give an error if you then attempt to load the database into a document.

If you just have a short fragment of inline verbatim text, consider one of the alternatives listed in the FAQ. If on the other hand you have a block of verbatim text you'll have to put the verbatim text in a separate file and load it using `\verbatiminput` (from the verbatim package) or `\lstinputlisting` (from the listings package). For example, in Figure 1.2 I have used `\lstinputlisting`.

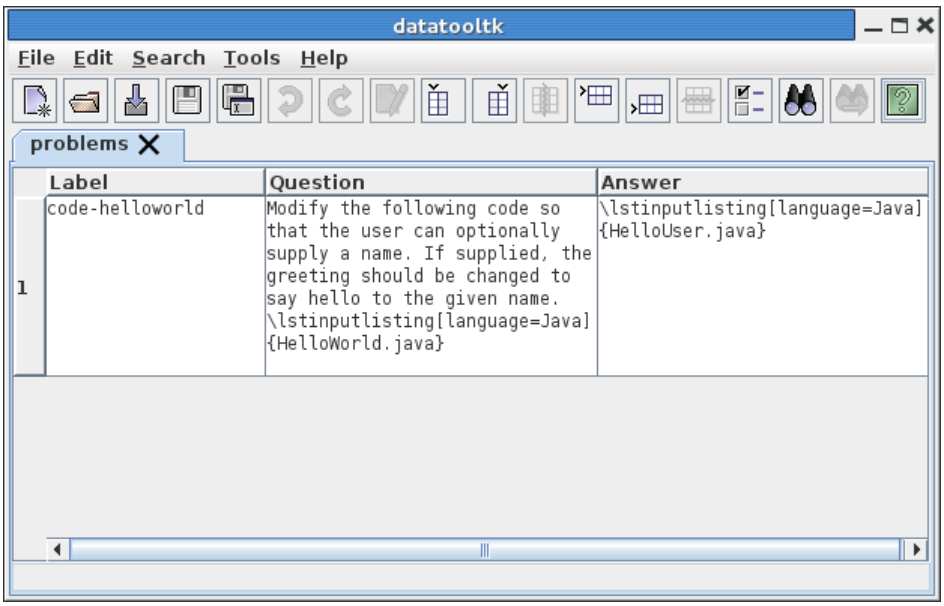

Figure 1.2: Verbatim Blocks Need to be in Separate Files

That database requires two files: `HelloWorld.java`

```
public class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.println("Hello World!");
   }
}
```

and `HelloUser.java`:

[1] More specifically, it checks for any occurrences of `\verb`, `\lstinline` or the beginning of the verbatim, lstlisting or alltt environments.

```
public class HelloUser
{
   public static void main(String[] args)
   {
      System.out.println("Hello "
         + (args.length==0 ? "anon" : args[0])+"!");
   }
}
```

Assuming that I've saved my database in a file called `prob-verb.dbtex` with database label "`problems`", here's a sample document:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}
\usepackage{listings}

\newtoggle{showanswers}
\toggletrue{showanswers}

\input{prob-verb.dbtex}

\begin{document}

\begin{enumerate}
  \DTLforeach*{problems}{\Question=Question,\Answer=Answer}%
  {%
    \item \Question

    \iftoggle{showanswers}{Answer: \Answer}{}
  }
\end{enumerate}

\end{document}
```

See also:

- Shuffling the Data
- Sorting and Shuffling
- Import probsoln Data

1.4 Null Values

Empty entries aren't the same as null entries. If you want a null entry, set the entry to `\@dtlnovalue`. A convenient way to do this is to select the cell and use Edit→Set Cell to Null. Alternatively, you can set all entries in a selected column to null with Edit→Column→Nullify Column and similiarly for a selected row with Edit→Row→Nullify Row.

In your LaTeX document, you can check for null values using datatool's `\DTLifnull` command. To check for empty values you can use one of etoolbox's commands, such as `\ifdefempty`.

2 Graphical Mode

To run `datatooltk` in graphical mode you must invoke it with either `datatooltk-gui` or `datatooltk --gui`. The main window is shown in Figure 2.1. Each database is in a tabbed pane, with the name of the database in the tab. Note that the name corresponds to the database's identifying label, as used in commands like `\DTLnewdb`. This is not necessarily the same as the filename (see section 1.2). Since this name is used as a label, it shouldn't contain any of TEX's special characters or any other active characters that could cause problems. An asterisk * following the label in the tab indicates that the database has been modified. If you move the mouse over the tab, you will see the filename appear in a tooltip, if the database has been saved to a datatool file.

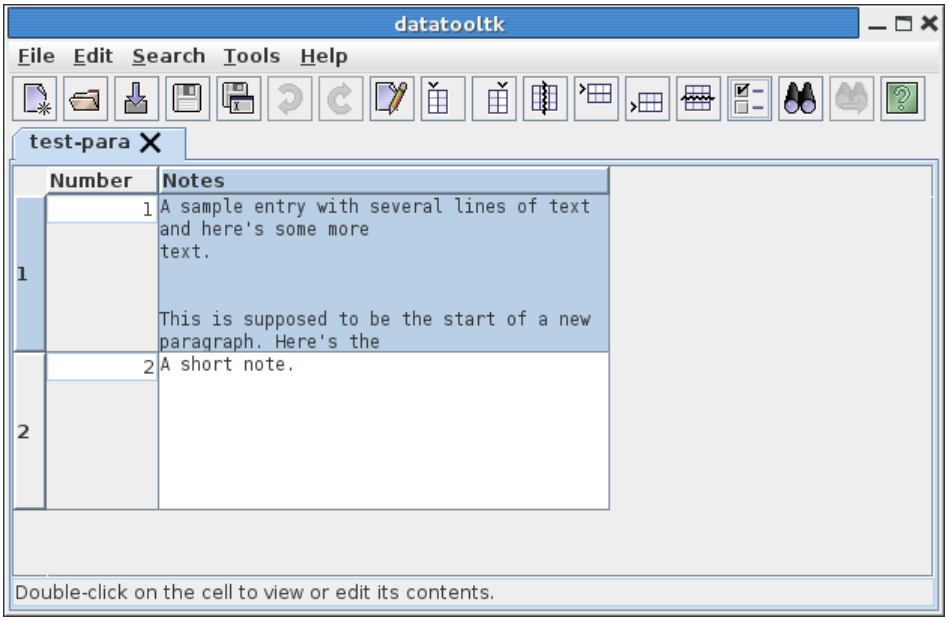

Figure 2.1: Main Window

You can use the File menu to create a new database, load an existing database or import data (see chapter 4). To load an existing database, use File→Open. These database files contain LATEX code in a specific format. The `datatooltk` application assumes a `.dbtex` file extension (see section 1.2). You can load these files into a LATEX document using `\input`, but remember to specify the `.dbtex` extension. (Also remember to load the datatool package.)

Each column has a corresponding data type: string, integer, real or currency. The type is automatically detected from the column data, but can be changed, as described in section 2.2.

Non-string entries can be edited by double-clicking on the relevant cell, or you can select the relevant cell and use Edit→Edit Cell. In the first case, a cursor will appear in the cell and you can edit the numerical value and press "Enter" to finish editing. In the second case, the cell editor dialog box will open, see section 2.1.

Only the first few lines of a string entry are visible in the main window. If an entry has more than that number of lines, you will need to use the cell editor dialog box to view the entire contents of that cell. The default row height can be changed in the Preferences dialog box (see chapter 6). Columns set to integer or real data types have single-lined cells with no line wrap. Columns set to currency data type may wrap, but using "Enter" will finish editing the cell (unless you're using the cell editor dialog box). If you insert a newline character in the cell edit dialog box (for any data type), the type for that column will be converted to "string".

To edit or view an entry in a column with the "string" data type, double-click on the relevant cell or select the cell and use Edit→Edit Cell to open the cell editor dialog box (see section 2.1). You can now scroll through the cell contents.

2.1 Cell Editor

To open the cell editor dialog box (see Figure 2.2) double-click on the required cell, which must be in a column with a string data type. Alternatively, select the cell (of any type) and use Edit→Edit Cell.

Remember that the contents of the cell should be LaTeX code, so be careful if you use any of TeX's special characters. Also, see the section on verbatim text (section 1.3) if you haven't already read it. Once you have made your edits, click on **Okay** to update the database. To discard the edits, click **Cancel**.

If you've used datatool, you will probably know that if you want a paragraph break in your cell entries you need to use `\DTLpar`, but with `datatooltk` you don't need to worry about it as blank lines in an entry will automatically be converted behind the scenes. Note that redundant blank lines will be removed.

Important: if you use datatool's `\DTLsaverawdb` or `\DTLprotectedsaverawdb` commands to overwrite your file, you will lose any pretty-printing spaces or comments in your code.

2.2 Header Dialog

Each column has a title, a uniquely identifying label and an associated type. The type can be one of: **String**, **Integer**, **Real** or **Currency**. The type is automatically detected from the column data, but can be changed using the Edit→Column→Edit Header menu item or by double-clicking on the column header which opens the header dialog box (see Figure 2.3). The label corresponds to the label used to identify the column in commands

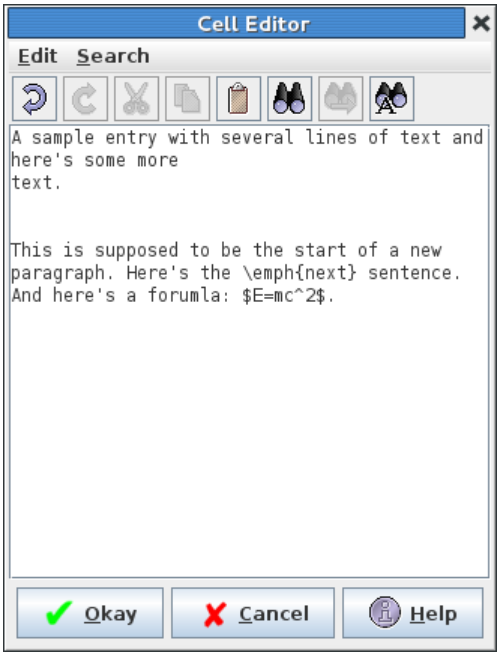

Figure 2.2: Cell Editor Dialog

such as `\DTLforeach`. The title is used in commands like `\DTLdisplaydb`. See chapter 6 for currency mappings.

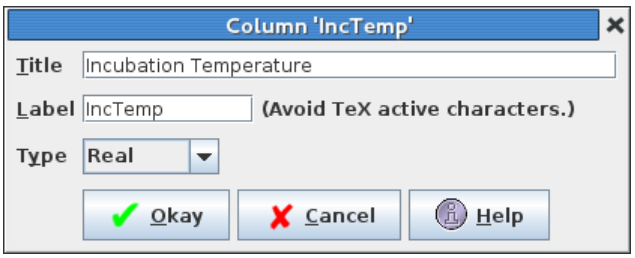

Figure 2.3: Header Dialog

In GUI mode, column headers show the title. If you move the mouse over the column header, you will see the label and type displayed in a tooltip (see Figure 2.4).

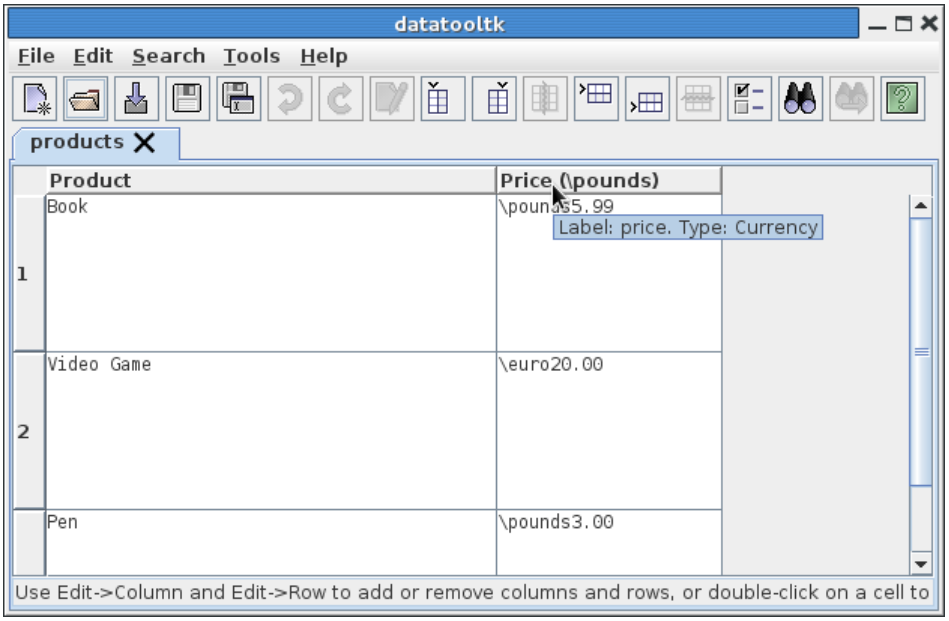

Figure 2.4: Header Details Shown in Tooltip

3 Tools

There are currently two tools available: sort (see section 3.1) and shuffle (see section 3.2). These both reorder the rows of the database and can be invoked either from the Tools menu or from the command line (as long as you have also loaded a database using `--in` or one of the import options). If you use both `--sort` and `--shuffle` in the command line invocation, sort will always be performed first, regardless of the option order.

3.1 Sorting the Data

Although you can sort data in datatool using `\DTLsort`, it's far more efficient to sort it in `datatooltk`.[1] So instead of doing, say,

```
\input{mydata.dbtex}% loads database labelled 'data' from file 'mydata.dbtex'
\DTLsortdb{Surname}{data}% sort data on 'Title' field
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

It's better to run, say,

```
datatooltk --in mydata.dbtex --sort Surname --out mydata-sorted.dbtex
```

Then in the document, just load `mydata-sorted.dbtex`:

```
\input{mydata-sorted.dbtex}
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

or, if you have shell escape enabled you can used TEX's `\write18` mechanism:

```
\immediate\write18{datatooltk --in mydata.dbtex --sort Surname
--out mydata-sorted.dbtex}

\input{mydata-sorted.dbtex}
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

[1] If the original data is in an SQL database, it's even more efficient to do the sorting in the SELECT statement when you import the data (see section 4.2).

A database can be sorted according to a particular column in either ascending or descending order. In batch mode, this is done with the `--sort` option, as shown above, where the sort column is identified by the column's unique label. If the label is preceded by `-` then descending order is used (for example, `--sort -Surname`). If the label is preceded by `+` (or has no prefix) then ascending order is used. For alphabetical comparisons you can also use `--sort-case-sensitive` for case-sensitive comparisons and `--sort-case-insensitive` for case-insensitive comparisons. The default is case-insensitive.

In GUI mode, sorting is done using the Tools→Sort menu item which opens the **Sort Database** dialog box (see Figure 3.1).

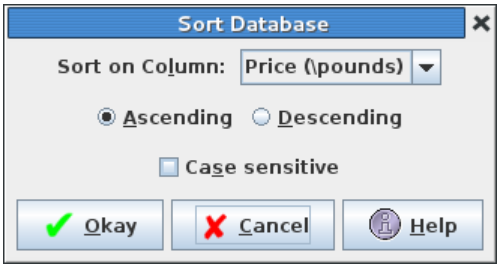

Figure 3.1: Sort Dialog

Select the column you wish to sort by from the drop-down list of column titles, and check the appropriate radio button for ascending or descending sort. If the column has the string data type, you also need to specify whether or not you want to use case-sensitive comparisons by checking or unchecking the **Case sensitive** box. If the column type has a numerical type, the entries will be sorted via a numerical comparison (10 is greater than 2) and the case-sensitive option is ignored. If the column type is a string type, the entries will be sorted via an alphabetical comparison ("10" comes before "2").

Example 2.
Consider the data shown in Figure 3.2 and reproduced in Table 3.1.

Table 3.1: Original Data

`Book`	`\pounds5.99`
`Video Game`	`\euro20.00`
`Pen`	`\pounds3.00`

The first column has a string data type and the second has a currency data type. Sorting in ascending order on the second column, will sort numerically on just the number. The currency symbol is ignored (see Table 3.2). If the type of the second column is changed from currency to string, and the sort is redone, the order is now based on a string comparison that includes the currency symbol (see Table 3.3).

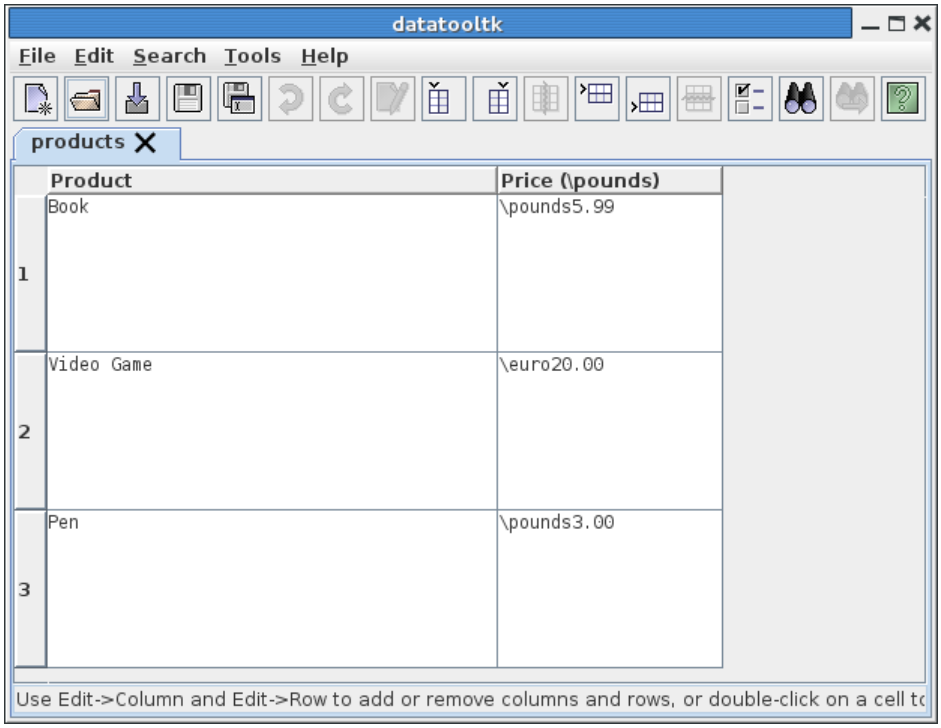

Figure 3.2: Original Data

Table 3.2: Data Sorted on Second Column (Currency Comparison)

Pen	\pounds3.00
Book	\pounds5.99
Video Game	\euro20.00

Table 3.3: Data Sorted on Second Column (String Comparison)

Video Game	\euro20.00
Pen	\pounds3.00
Book	\pounds5.99

3.2 Shuffling the Data

Data can be reordered by randomly swapping pairs of rows. By default, this random row swapping is done 100 times, but this number can be changed via the `--shuffle-iterations` command line option or the **Shuffle Iterations** field in the Preferences dialog box. Data shuffling can be performed either by the `--shuffle` command line option or the Tools→Shuffle menu item.

Example 3.
Consider the database shown in Figure 3.3. This database has three columns. The first is a question, the second is the corresponding answer (optional) and the third is a number indicating the question level. For example, 1 could correspond to easy and 2 could correspond to medium difficulty.

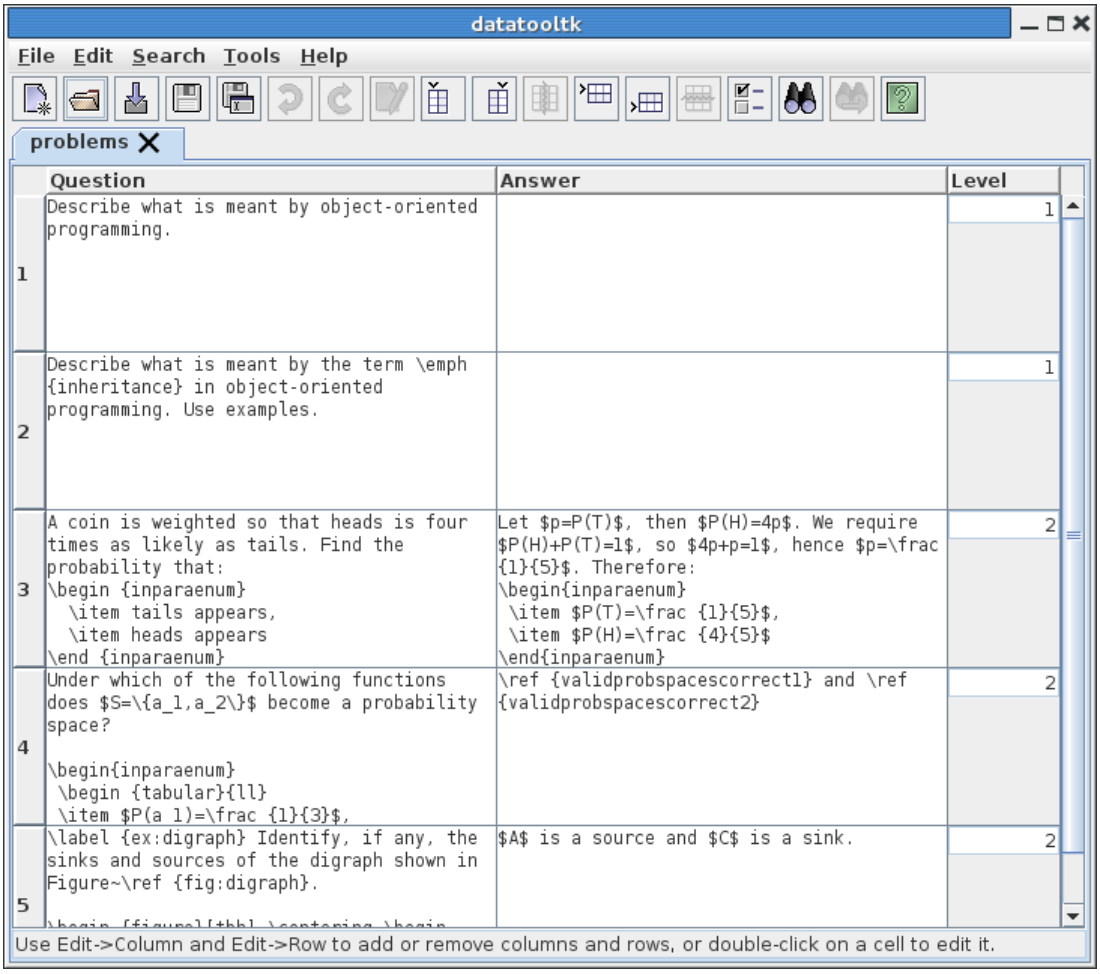

Figure 3.3: Shuffle Example

Now suppose I want to write an assignment sheet that has one randomly selected question of level 1 and two randomly selected questions of level 2. Let's suppose the file name is `data.dbtex` and the database label is "`problems`". Then I can run `datatooltk` in batch mode using:

```
datatooltk --shuffle --in data.dbtex --out data-shuffled.dbtex
```

Remember to use `--seed` if you don't want a different ordering every time you run that command. For example:

```
datatooltk --seed 2013 --shuffle --in data.dbtex --out data-shuffled.dbtex
```

This shuffled database can now be loaded in my document:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

% Used by some of the questions:
\usepackage{paralist}
\usepackage{tikz}

\newtoggle{showanswers}
\toggletrue{showanswers}

\input{data-shuffled.dbtex}

% Number to select from level 1
\newcounter{maxleveli}
\setcounter{maxleveli}{1}

% Number to select from level 2
\newcounter{maxlevelii}
\setcounter{maxlevelii}{2}

% Counter to keep track of level 1 questions
\newcounter{leveli}

% Counter to keep track of level 2 questions
\newcounter{levelii}

\begin{document}
```

```
\begin{enumerate}
 \DTLforeach*{problems}%
  {\Question=Question,\Answer=Answer,\Level=Level}%
  {%
    % Increment counter for this level
    \stepcounter{level\romannumeral\Level}%
    % Have we reached the maximum for this level?
    \ifnumgreater
      {\value{level\romannumeral\Level}}%
      {\value{maxlevel\romannumeral\Level}}%
    {}% reached maximum, do nothing
    {\item \Question

     \ifdefempty\Answer
     {}% no answer
     {% do answer if this is the solution sheet
       \iftoggle{showanswers}{Answer: \Answer}{}%
     }%
    }%
    % do we need to continue or have we got everything?
    \ifboolexpr
    {%
      test{\ifnumgreater{\value{leveli}}{\value{maxleveli}}}
      and
      test{\ifnumgreater{\value{levelii}}{\value{maxlevelii}}}
    }%
    {\dtlbreak}{}%
  }
\end{enumerate}

\end{document}
```

What if I want all the easy questions listed first? This requires some modifications to the code as shown below:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

% Used by some of the questions:
\usepackage{paralist}
\usepackage{tikz}
```

```
\newtoggle{showanswers}
\toggletrue{showanswers}

\input{data-shuffled.dbtex}

% Number to select from level 1
\newcounter{maxleveli}
\setcounter{maxleveli}{1}

% Number to select from level 2
\newcounter{maxlevelii}
\setcounter{maxlevelii}{2}

% Counter to keep track of level 1 questions
\newcounter{leveli}

% Counter to keep track of level 2 questions
\newcounter{levelii}

% List of level 1 questions
\newcommand*{\listleveli}{}

% List of level 2 questions
\newcommand*{\listlevelii}{}

\begin{document}

 \DTLforeach*{problems}%
  {\Question=Question,\Answer=Answer,\Level=Level}%
  {%
    % Increment counter for this level
    \stepcounter{level\romannumeral\Level}%
    % Have we reached the maximum for this level?
    \ifnumgreater
      {\value{level\romannumeral\Level}}%
      {\value{maxlevel\romannumeral\Level}}%
    {}% reached maximum, do nothing
    {% Add row number to the appropriate list
      \listcsxadd{listlevel\romannumeral\Level}{\DTLcurrentindex}%
    }%
    % do we need to continue or have we got everything?
    \ifboolexpr
    {%
      test{\ifnumgreater{\value{leveli}}{\value{maxleveli}}}
```

```
      and
      test{\ifnumgreater{\value{levelii}}{\value{maxlevelii}}}
    }%
    {\dtlbreak}{}%
  }

\renewcommand{\do}[1]{%
  \dtlgetrow{problems}{#1}%
  \dtlgetentryfromcurrentrow{\Question}{\dtlcolumnindex{problems}{Question}}%
  \dtlgetentryfromcurrentrow{\Answer}{\dtlcolumnindex{problems}{Answer}}%
  \item \Question

   \ifdefempty\Answer
   {}% no answer
   {% do answer if this is the solution sheet
     \iftoggle{showanswers}{Answer: \Answer}{}%
   }%
}

\begin{enumerate}

% do easy questions
\dolistloop{\listleveli}

% do medium level questions
\dolistloop{\listlevelii}

\end{enumerate}
\end{document}
```

Now, the `\DTLforeach` loop just stores the row numbers of the required questions in two lists, corresponding to the two different levels. Then each list is iterated through and the corresponding row is fetched using `\dtlgetrow`. Extending this example to accommodate an arbitrary number of levels is left as an exercise for the reader.

Remember that if you have shell escape enabled when you run LaTeX you can invoke `datatooltk` in your document *before* you load the database:

```
\immediate\write18{datatooltk --in data.dbtex --seed 2013 --shuffle
--out data-shuffled.dbtex}

\input{data-shuffled.dbtex}
```

3.3 Sorting and Shuffling

As mentioned earlier, if you specify both `--sort` and `--shuffle`, the sorting will always be performed first, regardless of the option order, but why would you want to sort the data if you're going to shuffle it? Consider the command invocation:

`datatooltk --shuffle --in` ⟨*in-file*⟩ `--out` ⟨*out-file*⟩

Every time you run this command, you will get a different ordering. If, however, you set a seed for the random generator, for example:

`datatooltk --seed 2013 --shuffle --in` ⟨*in-file*⟩ `--out` ⟨*out-file*⟩

You will always get the same random ordering *provided the original data in* ⟨*in-file*⟩ *has remained unchanged*. If you want to modify the shuffled data in your document and save it to the original file ⟨*in-file*⟩ using `\DTLsaverawdb`, the ordering in that file will change, so the next time you shuffle it, you'll get a different ordering, even if you use the same seed. If you sort first on a unique label, that will ensure the shuffle has the same starting point (unless you add or remove rows).

Example 4.
Suppose you have a database of exam questions and you want to keep track of the year in which each question was last used. (To make life easier, let's identify the academic year "2012/13" as 2013, the academic year "2013/14" as 2014, etc.) Let's further suppose the database of questions is in a file called `mth-101.dbtex` and the database label is "`problems`" (see Figure 3.4). The database contains a column with the label "`Label`", which uniquely identifies an exam question, a column with the label "`Question`" that contains the exam question, a column with the label "`Answer`" that contains the answer and an integer column with the label "`Year`" that contains the exam year in which that question was last used. (A zero entry means the question hasn't been used.)

Now suppose the exam requires five questions to be randomly selected from this database, but must not include any question used in the past three years. So the exam LaTeX document needs to load in a shuffled version of `mth-101.dbtex`, use the first five questions that don't have a year set in the past three year range, set the year for the selected questions to the current exam year, display the questions (and optionally the answers for the solution sheet), and at the end of the document, overwrite `mth-101.dbtex` so that it now has a record of this year's exam questions.

There are two problems. Firstly, if the process is to be automated with a call to `datatooltk --shuffle` followed by a LaTeX call, a different set of problems will be selected on each run, even with the same seed. To overcome this, a sort on the `Label` column needs to be done before the shuffle:

```
datatooltk --sort Label --seed 2013 --shuffle --in mth-101.dbtex
--out mth-101-shuffled.dbtex
```

(The symbol ↩ above indicates a line wrap. Don't insert a line break at that point.) This way the shuffle always starts from the same ordering.

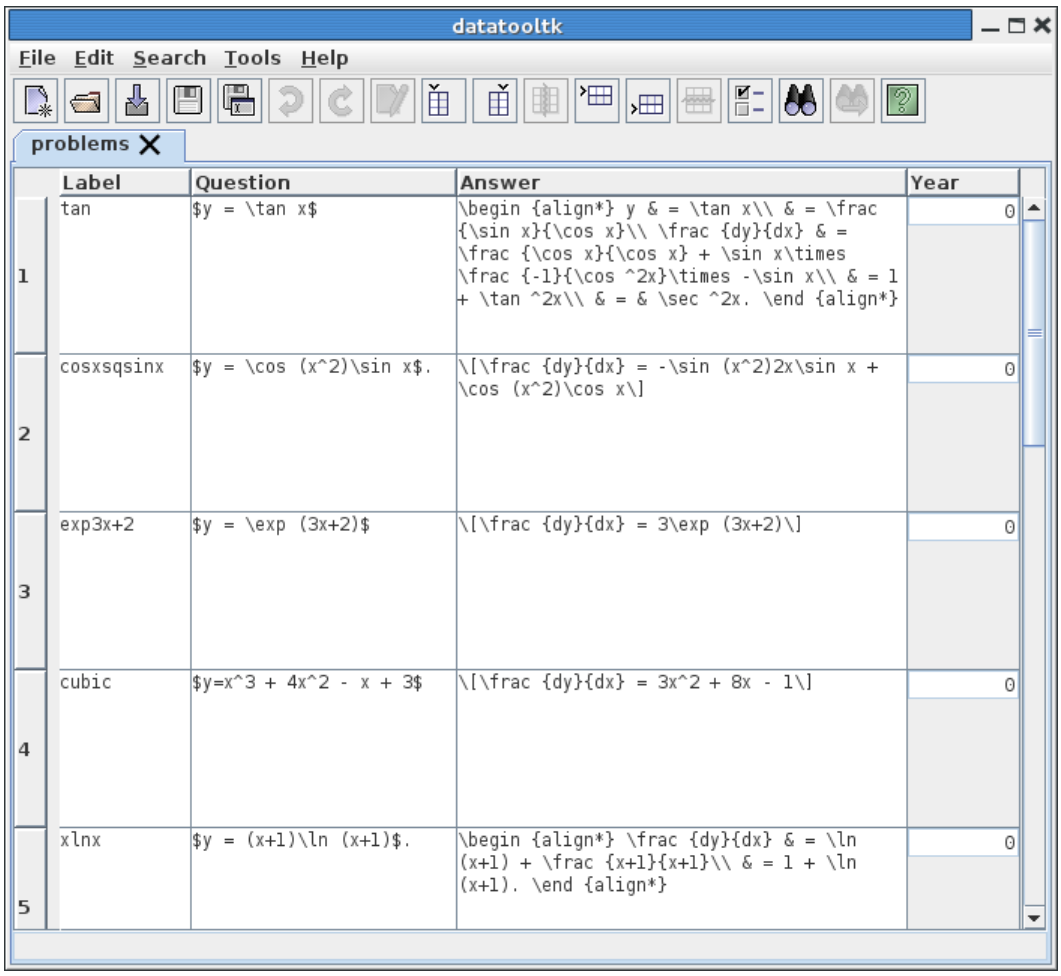

Figure 3.4: Sort and Shuffle Example

The second problem occurs if you edit the database such that you add or remove rows. This will change the initial conditions, even with the sort. If you add or remove rows, you need to accept that the document may well end up with a different selection of questions, which is okay if you haven't finalised the exam, but it means that some of the questions will be identified as having been used in that exam year from a previous run but are now no longer selected. In order to make them available for the next year, if they haven't been selected but have had the year set to this year, the year needs to be cleared.

To solve this, once you have selected the maximum required number of questions, don't break out of the loop, as was done earlier (see section 3.2). Instead, for the rest of the loop, if the exam year is set to the current year, clear it.

```
% arara: pdflatex: {shell: on}
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}
\usepackage{listings}
%
\newtoggle{showanswers}
\togglefalse{showanswers}

\newcommand{\examyear}{2013}
\newcommand{\maxquestions}{5}
\newcounter{question}

\immediate\write18{datatooltk --sort Label --seed \examyear\space
--shuffle --in mth-101.dbtex --out mth-101-shuffled.dbtex}

\input{mth-101-shuffled.dbtex}

\begin{document}

\begin{enumerate}
  \DTLforeach{problems}{\Question=Question,\Answer=Answer,\Year=Year}%
  {%
     % If year hasn't been specified, set it to 0 to
     % allow numeric comparisons
     \ifdefempty{\Year}{\def\Year{0}}%
     {% check for null as well, just in case
        \DTLifnull{\Year}{\def\Year{0}}{}
        \DTLappendtorow{Year}{0}%
     }%
     \ifnumgreater{\value{question}}{\maxquestions}
     {%
```

```
        % Finished selecting questions, unset any year
        % equal to this exam year
        \ifnumequal{\Year}{\examyear}
        {%
           % unset year
           \DTLreplaceentryforrow{Year}{0}%
        }%
        {}%
     }%
     {%
        % Still selecting questions.
        % Check the year
        \ifboolexpr
        {%
           test{\ifnumequal{\Year}{\examyear}}
           or
           test{\ifnumless{\Year}{\examyear-3}}
        }
        {% select this question
           \stepcounter{question}%
           \item \Question

           \iftoggle{showanswers}{Answer: \Answer}{}%
           % update year
           \DTLreplaceentryforrow{Year}{\examyear}%
        }%
        {% skip this question, it was used in the past 3 years
        }%
     }%
  }
\end{enumerate}

% update database file
\DTLprotectedsaverawdb{problems}{mth-101.dbtex}
\end{document}
```

Note: since this overwrites the datatool file, you will lose any pretty-printing spaces or comments you may have done in `datatooltk`'s cell editor dialog.

3.4 Plugins

Plugins are usually associated with a particular template (see chapter 5) and provide a convenient way of adding a row of data to the currently selected database. Typically when a plugin is run it will add a new row of data if no row is selected, otherwise it will

allow you to edit the selected row. **Note:** you must have Perl installed to use the plugins (see chapter 6). **Really Important Note:** the plugin is sent the database information when you start each instance of the plugin, so if you change the database in `datatooltk` while a plugin is running there may be unexpected results. Wait until the plugin has finished (usually by clicking on **Okay** or **Cancel**) before you make any further edits to the database.

3.4.1 The people Plugin

The `people` plugin is designed for use with databases created using the `people` template.

Example 5.
Suppose you create a new database using the `people` template. This creates a database with the following fields: ID, Title, Surname, Forename, Address, Telephone and Email, as illustrated in Figure 3.5.

Having created this database, I can just use the Edit→Row menu to insert rows and then edit each entry, but suppose I want to automatically increment the associated ID for each person. I can do this using the `people` plugin that corresponds to this template via the Tools→Plugins menu.

If a row is currently selected, this plugin will allow you to edit the data for that row. Otherwise, it will allow you to insert a new row. For a new row of data, the `people` plugin will open the dialog box shown in Figure 3.6.

After entering the data, I can click on **Okay** and a new row of data is added to the database (see Figure 3.7). Note that the plugin has converted newline characters in the address into \\. The ID has automatically been inserted.

Since the `people` plugin only adds or modifies a single row at a time, if you no longer require an entry, you can delete the unwanted row using Edit→Row→Delete Row.

3.4.2 The datagidx Plugin

The datagidx package creates its own custom database to store terms, symbols and acronyms. The `datagidx` template will create a database that contains datagidx's required fields. There are a lot of fields, some of which are reserved for datagidx's private use. The `datagidx` plugin, available via the Tools→Plugins menu, provides a convenient interface to add or edit entries. If no row is selected, the plugin will create a new row. If a row is selected, the plugin will allow you to edit or remove the row. Since the `datagidx` plugin can modify other rows at the same time (for example, if you set a parent entry or cross-reference) it's recommended that you use the `datagidx` plugin to remove an entry (via the **Remove Entry** button) rather than using Edit→Row→Delete Row.

Example 6.
The new database (created via File→New From Template) is shown in Figure 3.8. The default name of the database is "index". You can change it as required, but don't call it "datagidx" as the datagidx package creates a database with that name for its private

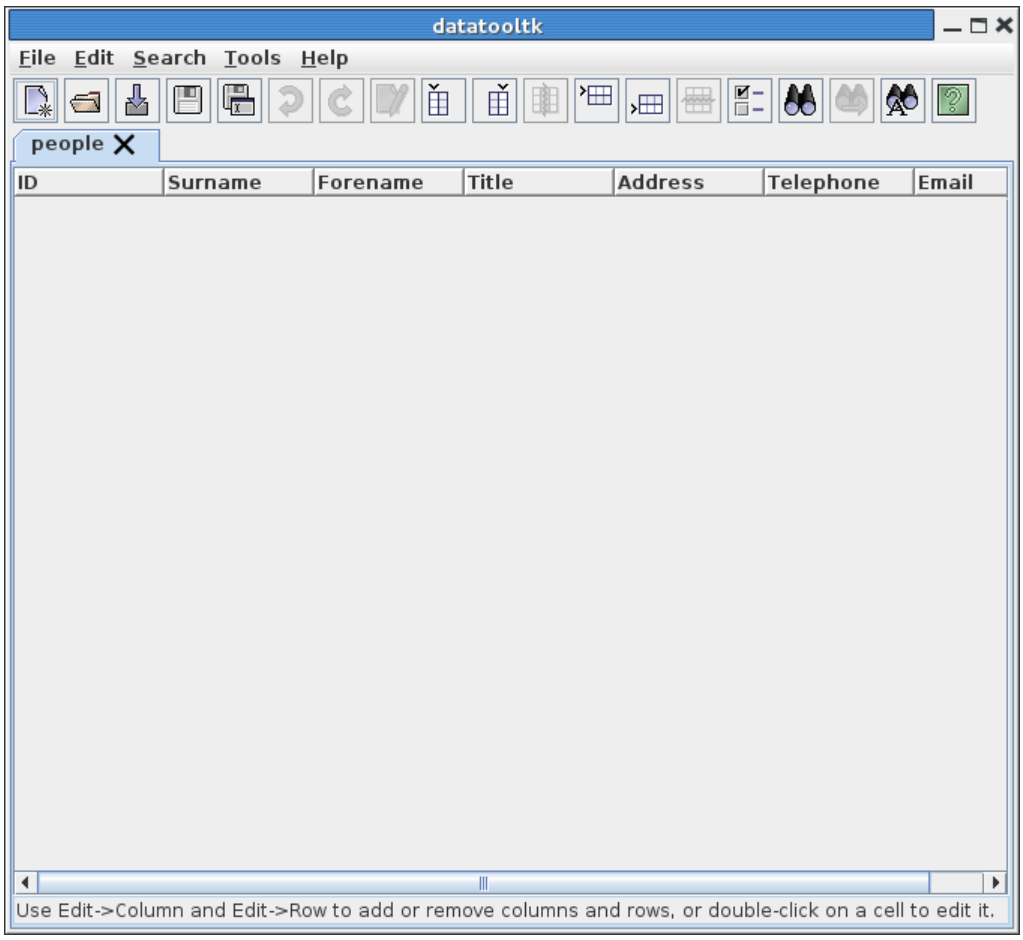

Figure 3.5: Database Created From people Template

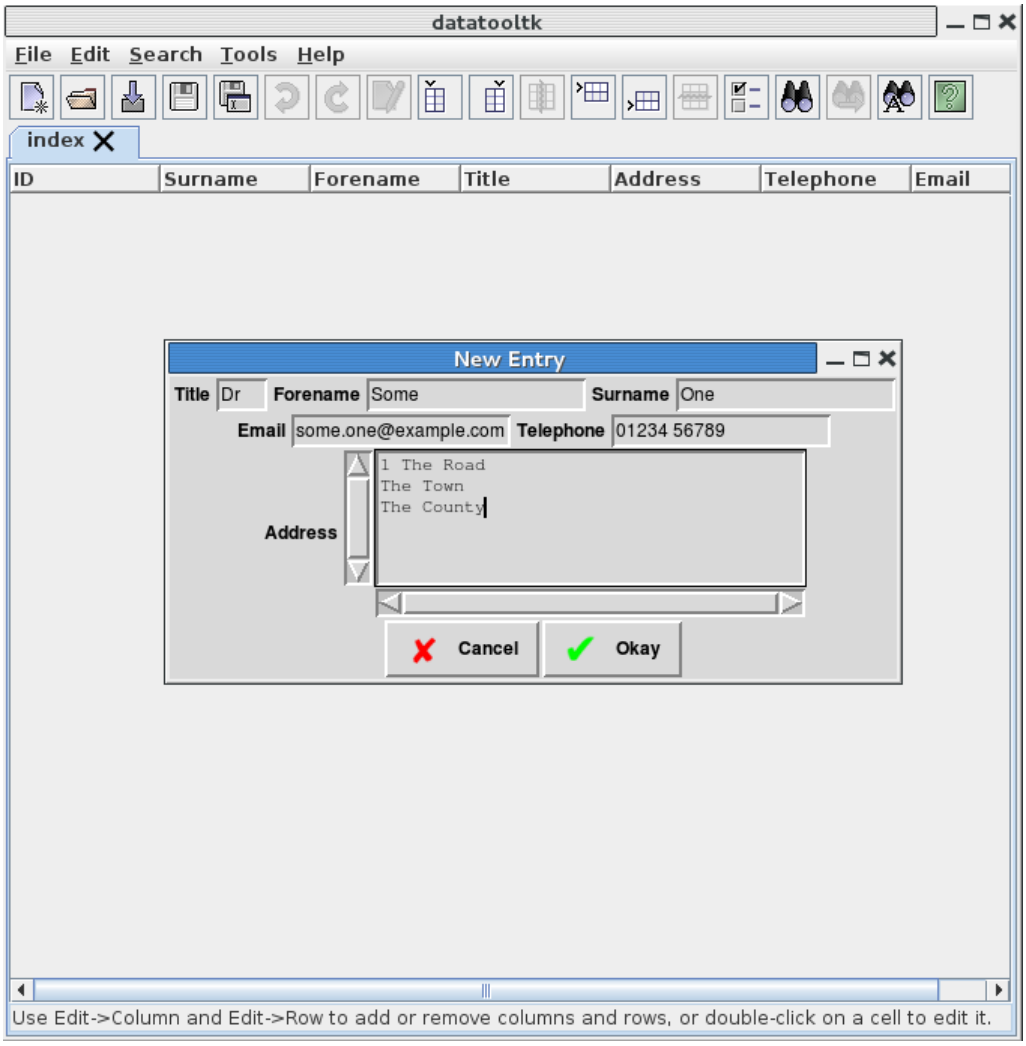

Figure 3.6: The people Plugin Dialog

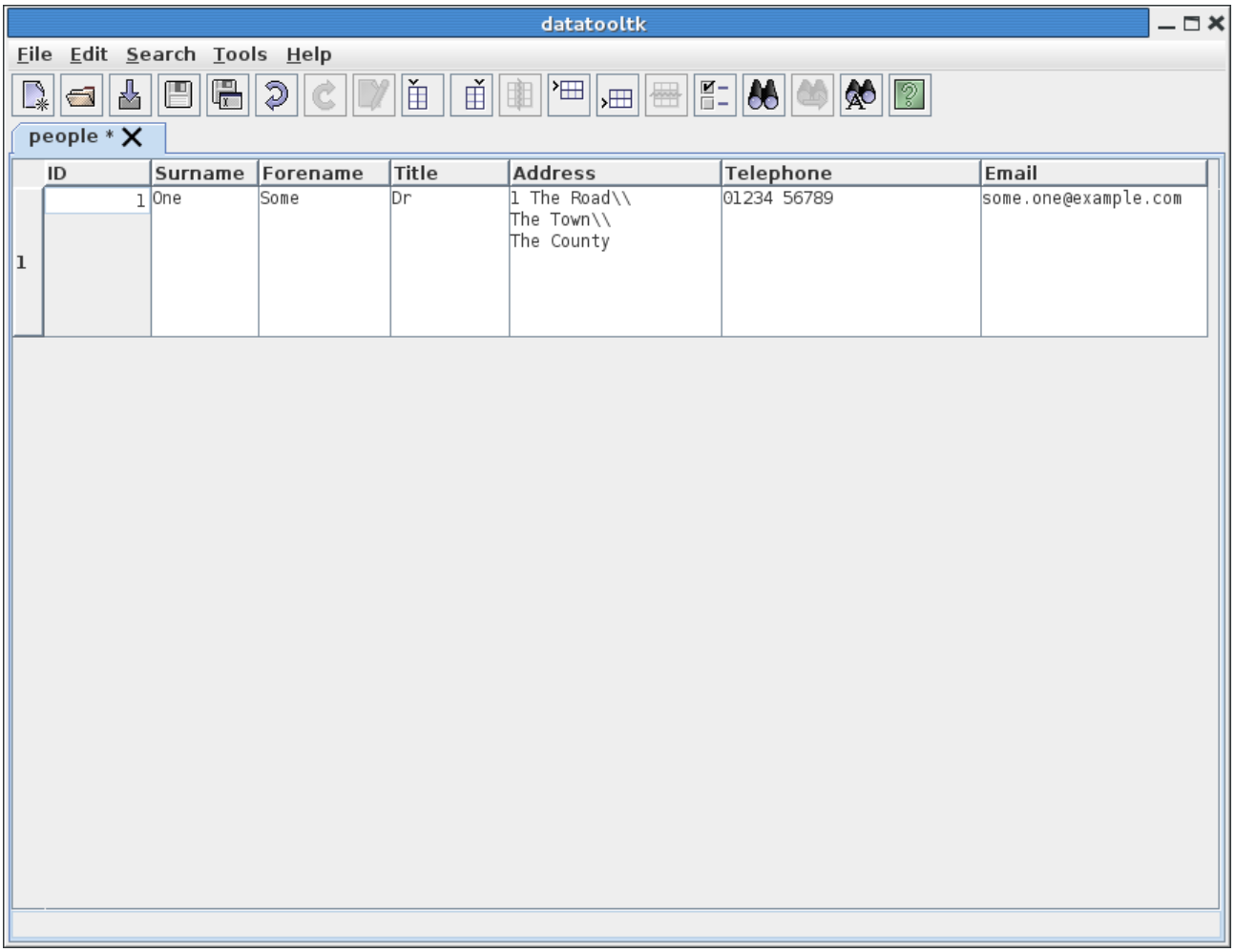

Figure 3.7: A New Row of Data

use. Once this database has been created, the `datagidx` plugin will open the dialog box shown in Figure 3.9.

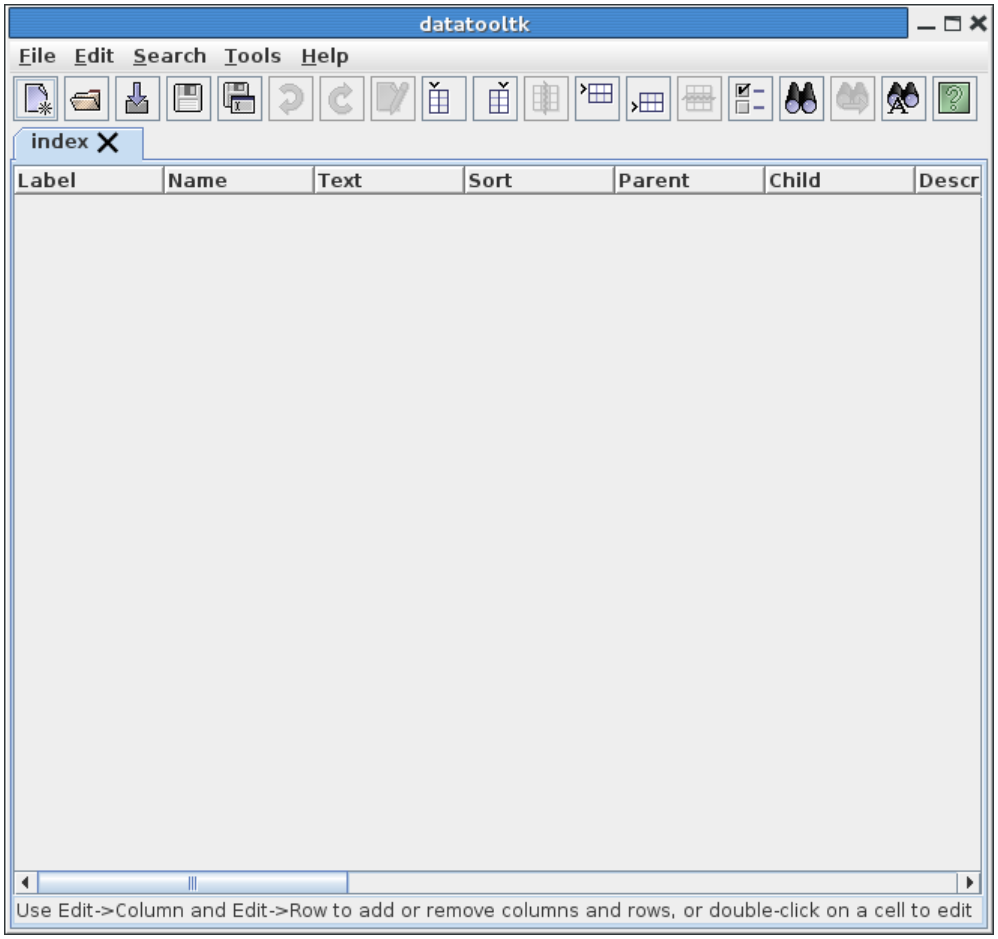

Figure 3.8: A Database Created From the datagidx Template

Since many of the fields are often duplicated (for example, the `Name` field is often the same as the `Text` field) if you first enter the name in the **Name** field, when you move the focus to another field, default entries will be added to most of the empty fields. For example, in Figure 3.10 I typed "bird" in the **Name** field and then moved the cursor to the **Description** field. This automatically filled in default values for the **Label**, **Sort**, **Text**, **Short**, **Long**, **Plural**, **Short Plural** and **Long Plural** fields. Since this is the first entry, there are no options for the **Parent**, **See** and **See Also** fields. (The last two are hidden in Figure 3.10 as the **Cross-Reference** button is unchecked.)

When I click on **Okay**, a new row is added to the database (see Figure 3.11). Note that I didn't specify a parent for this entry so the parent has been given the value `\@dtlnovalue`, which ensures it will work correctly when the datagidx package tests if

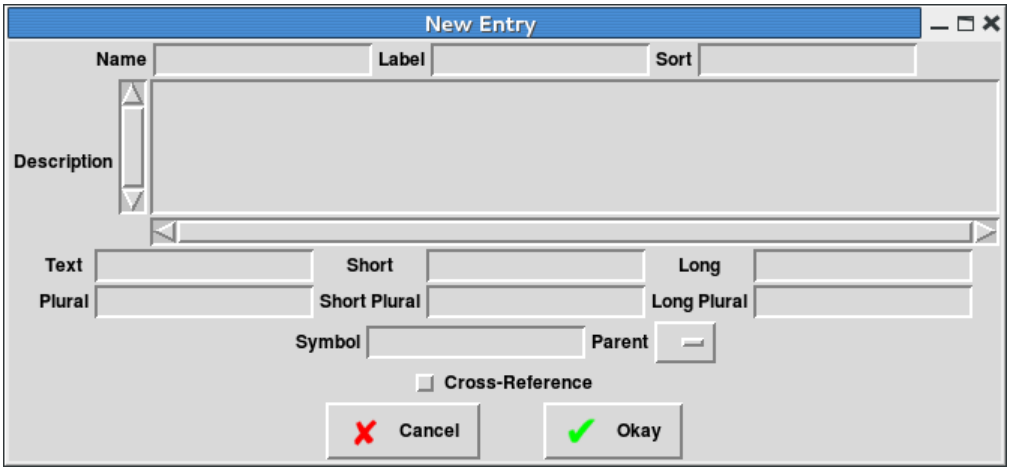

Figure 3.9: The datagidx Plugin Dialog

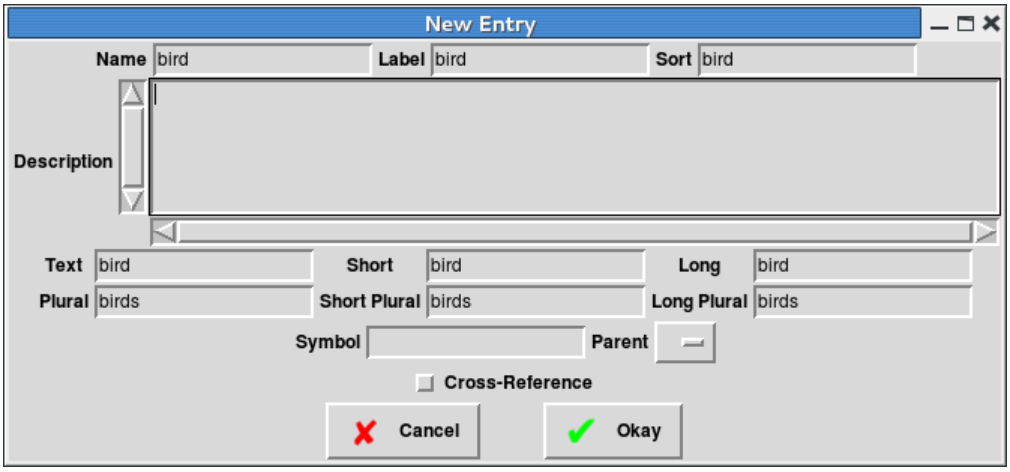

Figure 3.10: Most Fields Are Auto-Filled From the Name Field

the parent entry is null.

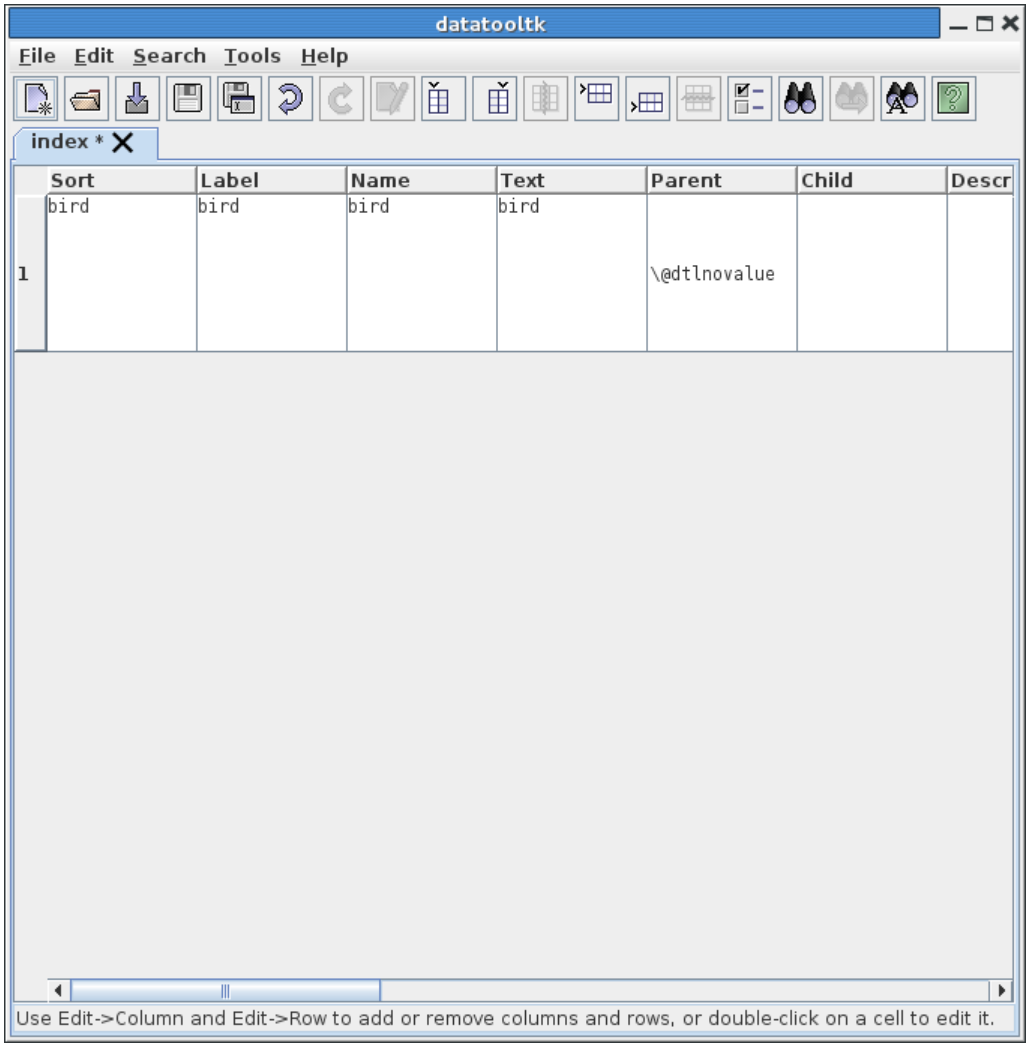

Figure 3.11: New Row Added to the Database

If I use the `datagidx` plugin to create a new row, there are now options available in the **Parent**, **See** and **See Also** fields (see Figure 3.12).

In Figure 3.12 I have set the parent to `bird`. When the new row is added, the plugin automatically adjusts the `bird` entry to include the new `duck` label as one of its children (see Figure 3.13).

It's also possible to cross-reference another entry. There are two ways of cross-referencing an entry: (1) using **See** which redirects the reader to a synonym that has the location list; (2) using **See Also** which in addition to the location list refers the reader to one or more related topics. (See the datagidx section of the datatool user manual for further de-

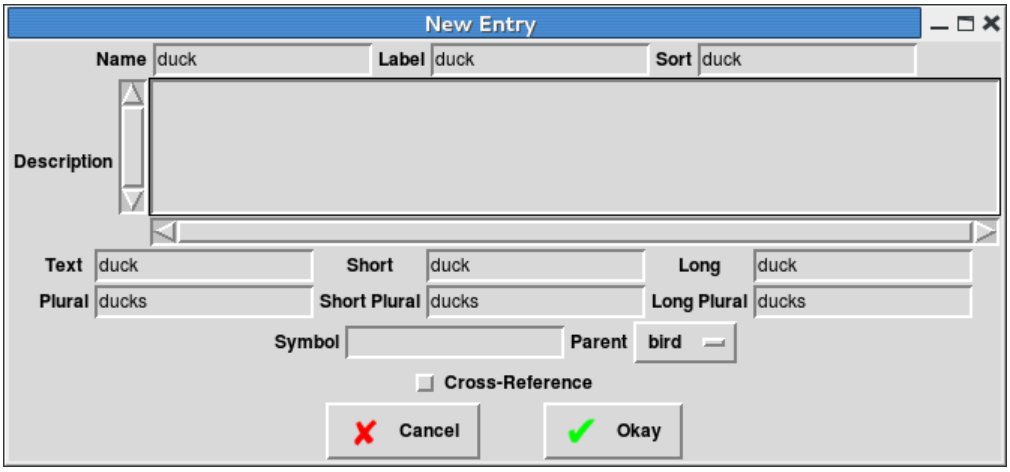

Figure 3.12: Parent Field Lists Other Entry Labels

tails.) To enable either form of cross-referencing, make sure the **Cross-Reference** button is selected. This will display extra options, shown in Figure 3.14.

Either select the **See** button and choose the synonym from the drop-down box next to it, or select the **See Also** button and select the related cross-reference from the drop-down box to the right and either click on **Add "See Also" Entry** to append it to the **See Also** list or click on **Remove "See Also" Entry** to remove it from the list. For example, in Figure 3.14 I've added the `chicken` and `turkey` entries to the **See Also** list. (Assuming I've already added the `chicken` and `turkey` entries before defining this new entry.)

Once I've enter all my terms, I can sort the data according to the **Sort** column. (Recall section 3.1.) Now let's suppose I save this sorted database to a file called `datagidx-test.dbtex`. I can now load it into a LaTeX document as follows[2]:

```
\documentclass{article}

\usepackage{datagidx}

\loadgidx{datagidx-test.dbtex}{Index}

\begin{document}

Reference some terms: \gls{duck}, \gls{bird}, \gls{parrot},
\gls{crocodile}, \gls{caiman}, \gls{alligator}.

\printterms[columns=1]
```

[2]Ensure you have at least version 2.15 of the datatool bundle.

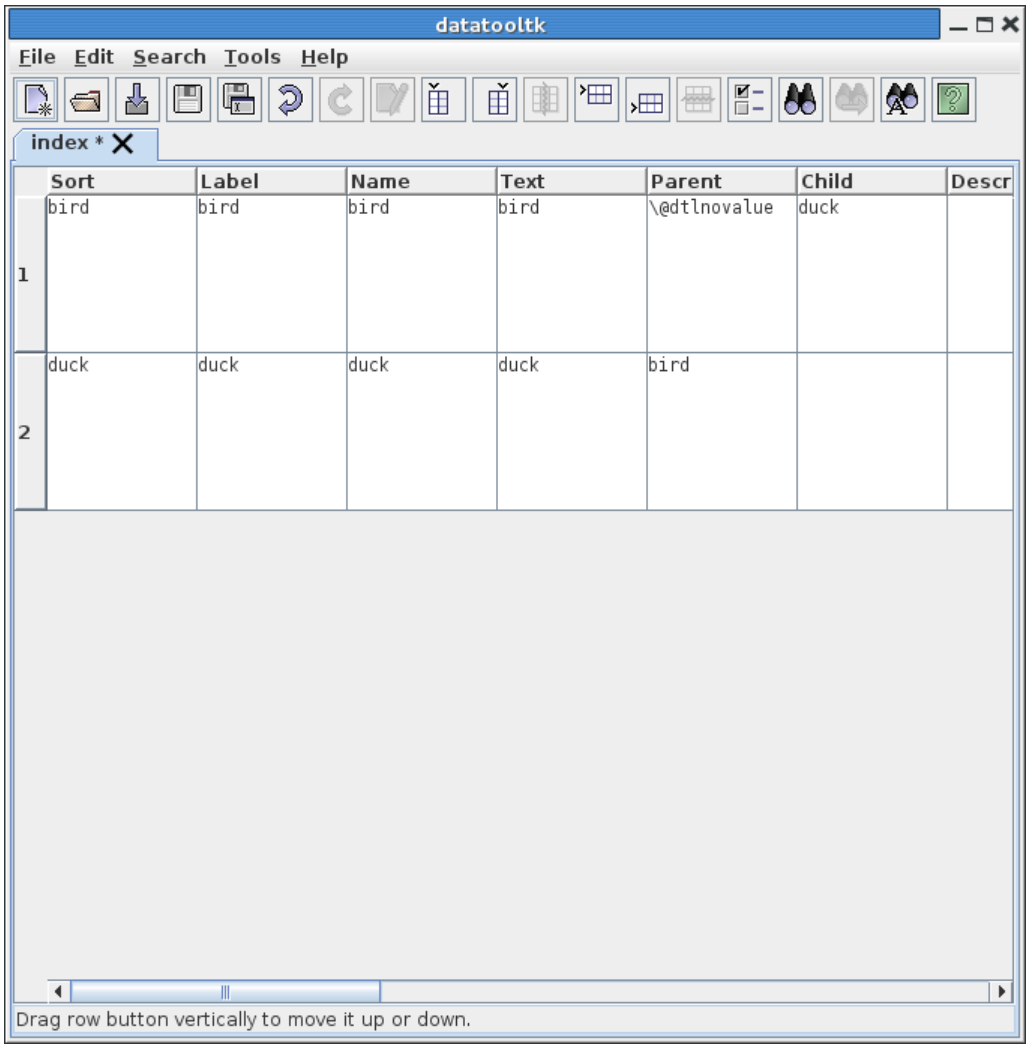

Figure 3.13: Child Entry Automatically Adjusted For Parent Entry

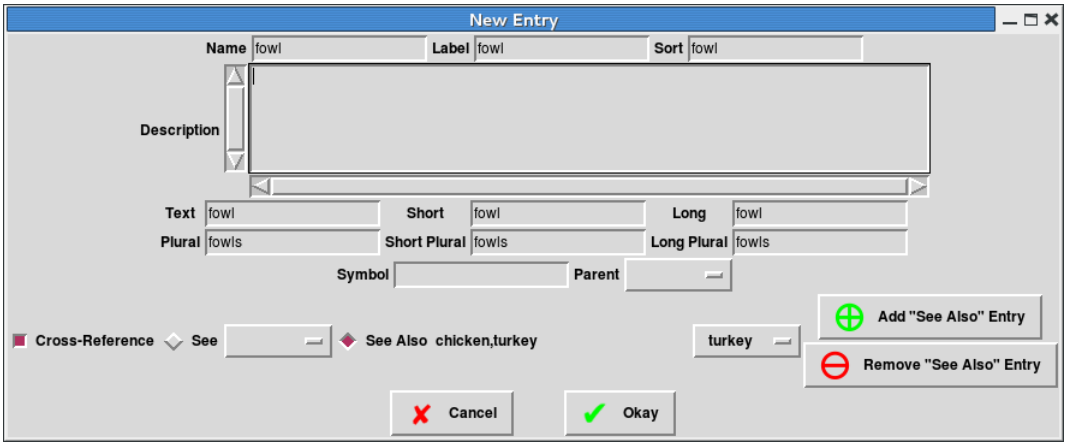

Figure 3.14: Cross-Referencing Entries

```
\end{document}
```

3.4.3 Comparison of glossaries and datagidx

If you're interested in the comparitive efficiency between using glossaries and datagidx, I performed a test with 100 entries randomly selected from a dictionary. The entries were listed in a file called `entries` in the form:

```
\newterm{minnow}
\newterm{running board}
\newterm{diamant\'e}
```

First, let's look at a document that uses datagidx with `\newgidx`:

```
% arara: clean
% arara: pdflatex
% arara: pdflatex
\documentclass{report}

\usepackage{datagidx}

\newgidx{index}{Index}

\input{entries}

\begin{document}

\tableofcontents
```

```
\chapter{Sample}
\glsaddall{index}

\printterms[postheading={\addcontentsline{toc}{chapter}{Index}}]

\end{document}
```

In general you need three LaTeX runs to compile a datagidx document. In this case, you actually only need to do it twice since there are no location lists.

Now let's test a datagidx document where `datatooltk` does the sorting. First, we need to generate a `.dbtex` file that corresponds the same set of entries. This can be done with the following document:

```
\documentclass{article}

\usepackage{datagidx}

\newgidx{index}{Index}

\input{entries}

\begin{document}

\DTLprotectedsaverawdb{index}{index.dbtex}

\mbox{}\newpage

\end{document}
```

This just converts the entries listed in `entries.tex` into the appropriate database file, simulating having entered the terms using `datatooltk`'s datagidx plugin. The file is saved as `index.dbtex`. Remember that this data only needs to be sorted when you add a term. This can either be done in `datatooltk`'s GUI mode or it can be done in batch mode:

```
datatooltk --in index.dbtex --sort Sort --out index-sorted.dbtex
```

This creates a file called `index-sorted.dbtex`. This file can be loaded into a document as follows (NB there's a bug in `\glsaddall`, which is patched using the `\setkeys` line. This will be fixed in the next version of datagidx):

```
% arara: clean
% arara: pdflatex
% arara: pdflatex
```

```
\documentclass{report}

\usepackage{datagidx}

\loadgidx{index-sorted.dbtex}{Index}

\begin{document}

\tableofcontents

\chapter{Sample}

\setkeys{newterm}{database=index}% patch for \glsaddall bug
\glsaddall{index}

\printterms[postheading={\addcontentsline{toc}{chapter}{Index}}]

\end{document}
```

Now let's look at the glossaries package. Since the terms have been defined using `\newterm`, I've defined a command that will convert this into an equivalent `\newglossaryentry`. Some of the entries have accents in their name, which datagidx automatically strips when generating the default label, so I've added a quick way of generating an analogous accent-free label and sort key that can be used with `\newglossaryentry`. Here's the document:

```
% arara: clean
% arara: pdflatex
% arara: makeglossaries
% arara: pdflatex
\documentclass{report}

\usepackage[nonumberlist,nogroupskip,toc]{glossaries}
\usepackage{glossary-mcols}

\makeglossaries
\renewcommand{\glossaryname}{Index}
\renewcommand{\glsnamefont}[1]{\textmd{#1}}

\newcommand{\newterm}[1]{%
  \bgroup
    \def\c##1{##1}%
    \let\'\c
    \xdef\thislabel{#1}%
  \egroup
```

```
  \def\thisname{#1}%
  \edef\donewgloss{%
    \noexpand\newglossaryentry{\thislabel}%
    {name={\expandonce\thisname},%
     sort={\thislabel},%
     description={\noexpand\nopostdesc}}%
  }%
  \donewgloss
}

\input{entries}

\begin{document}
\tableofcontents

\chapter{Sample}
\glsaddall

\printglossary[style=mcolindex]

\end{document}
```

In order to compare them, I used `arara` with the Linux `time` command. In each case, the `clean` directive is used at the start to ensure the tests start without an auxiliary files. Since there are no location lists, only two LaTeX calls are used on each example. If there were location lists, the datagidx examples would both need a third LaTeX call.[3] Remember that with the example that uses `index-sorted.dbtex`, `datatooltk` needs to sort the database whenever a new entry is added to the database. Assuming that all possible required entries have been added to the database, we just need one sort operation:

```
datatooltk --in index.dbtex --sort Sort --out index-sorted.dbtex
```

Invoking this with the Linux `time` command gives:

```
real 0m0.233s
user 0m0.403s
sys 0m0.014s
```

Now `arara` can be run on each of the three test documents (via the `time` command). The result from the first test that uses datagidx and `\newgidx`. The results are:

[3] The datagidx package doesn't generate a location with `\glsadd` or `\glsaddall`, whereas glossaries does. I've suppressed the location list in the glossaries example to produce an equivalent document.

```
real 0m12.170s
user 0m12.199s
sys 0m0.038s
```

The next test uses datagidx and `\loadgidx`. The result is:

```
real 0m2.315s
user 0m2.343s
sys 0m0.037s
```

The third test uses glossaries. The result is:

```
real 0m0.778s
user 0m0.801s
sys 0m0.057s
```

Using glossaries is clearly faster than using datagidx. In the case of `\loadgidx`, glossaries is approximately three times faster. In the case of `\newgidx`, glossaries is approximately 15 times faster. If a third LaTeX run was required for the location lists with `\newgidx`, using glossaries would be approximately 24 times faster (with only two LaTeX runs and one `makeglossaries` run).

4 Importing Data

Data can be imported from CSV files (see section 4.1), SQL databases (see section 4.2) or from files that can be imported with the probsoln package's `\loadallproblems` command (see section 4.3). In the case of the first two, `datatooltk` can automatically convert TEX's special characters if the `--map-tex-specials` command line option is used or the **Map TeX characters when importing data from CSV or SQL** option has been selected in the Preferences dialog box (see chapter 6).

4.1 Import CSV Data

Data can be imported from a CSV file using the `--csv` command line option or (in GUI mode) using the File→Import→Import CSV menu item. The default separator is a comma and the default delimiter is the double-quote character. These can be changed using the `--sep` and `--delim` command line options or in the Preferences dialog box (see chapter 6). Unlike datatool's `\DTLloaddb` command, `datatooltk` can import data with multilined entries (via the Open CSV library http://opencsv.sourceforge.net/). Multiple blank lines within entries are automatically converted to `\DTLpar` (although you won't see this in GUI mode).

If the CSV file has a header row, you must make sure the `--csvheader` option is used or the **Has Header Row** option is checked in the Preferences dialog box. If the CSV file has no header row, you must make sure the `--nocsvheader` option is used or the **Has Header Row** option is unchecked in the Preferences dialog box.

Example 7.
Consider the CSV file shown below:

```
Number,Notes
1,"A sample entry with several lines of text and here's some more
text.


This is supposed to be the start of a new paragraph. Here's the
next sentence."
2,A short note.
```

This has a cell with multiple lines. When it's imported into `datatooltk`, the paragraph break is converted to `\DTLpar`. However, this isn't visible when you look at the file in GUI mode (see Figure 4.1).

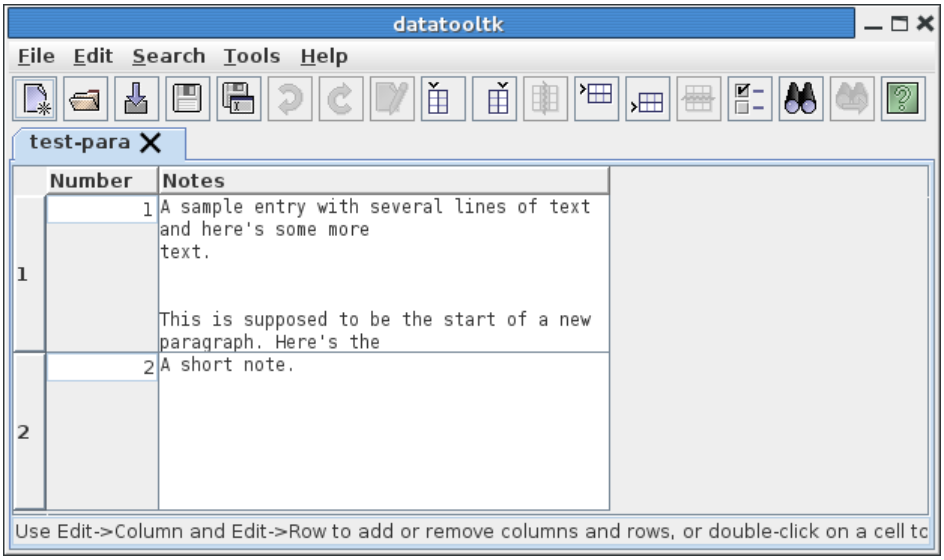

Figure 4.1: Paragraph Breaks Appear as a Single Blank Line

Note that the redundant second blank line in the CSV file has gone as multiple blank lines are replaced by a single `\DTLpar`.

4.2 Import SQL Data

Data can be imported from an SQL database using the `--sql` command line option or the File→Import→Import SQL menu item. You additionally need to supply the database, port, prefix, host, user name and password. In batch mode, you can use the command line options `--sqldb`, `--sqlport`, `--sqlprefix`, `--sqlhost` and `--sqluser`. You can specify the password with `--sqlpassword`, but that isn't secure. If you don't use that, you will be prompted for the password, where the text you enter won't be visible. See chapter 1 for more details about command line options.

In GUI mode, when you use File→Import→Import SQL the dialog box shown in Figure 4.2 will be displayed, where you can enter the settings. In addition to the above named settings, you must also specify the SQL SELECT statement that identifies the required data to import. (This manual assumes that if you have data in an SQL database, then you have a basic knowledge of SQL syntax.)

For example, in Figure 4.2 I want to import all data from the table called `customers` in the MySQL database called `myshop`. (I've created a user called `shopadmin` with SELECT privileges for this database.) Once I've entered this information, I then click on **Okay** and the password dialog box will appear (see Figure 4.3).

Alternatively, I can use batch mode to import and save the data from the command prompt:

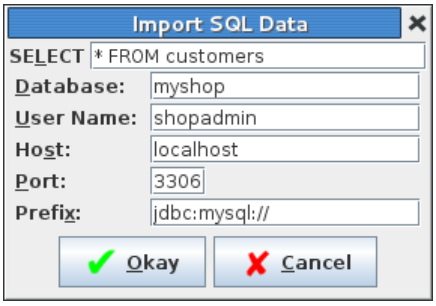

Figure 4.2: SQL Import Dialog Box

Figure 4.3: Password Dialog Box

```
datatooltk --out customers.dbtex --sql "SELECT * FROM customers"
--sqldb myshop --sqluser shopadmin
Password:
```

(The symbol ↩ above indicates a line wrap. Don't insert a line break at that point.) The password should be entered at the **Password** prompt. Remember that it's more efficient to get the SQL database to do any sorting. For example (assuming the table has a column called `Surname`):

```
datatooltk --out customers.dbtex --sql "SELECT * FROM customers ORDER BY
Surname" --sqldb myshop --sqluser shopadmin
Password:
```

4.3 Import probsoln Data

The probsoln package allows you to define problems (and optionally their solutions) using `\newproblem` or the defproblem environment. `datatooltk` can load a file containing these definitions and convert the probsoln data into a datatool database containing three columns with keys: `Label`, `Question` and `Answer`. You can import one of these files using the `--probsoln` command line option or (in GUI mode) using the File→Import→Import probsoln File menu item.

TEX is a difficult language to parse, so `datatooltk` uses LATEX to help gather the data from the imported file. The `datatooltk` application creates a temporary LATEX file and runs LATEX on it in the background. It assumes that the `latex` application is on your path. If this isn't the case, you will have to specify the location of the `latex` executable in the Preferences dialog box (see chapter 6). The temporary files are deleted when you quit `datatooltk` unless you have used the `--nodelete-tmp-files` option.

Note: `datatooltk` doesn't support problems that require arguments. Any instance of `#`⟨n⟩ will be replaced with `##`⟨n⟩, but you will have to replace those with something else. Also, recall from section 1.3 that you can't have verbatim text in a datatool database, but you can use `\verbatiminput` (from the verbatim package) or `\lstinputlisting` (from the listings package). Since LATEX is used to gather the data, pretty-printing spaces and comments won't be imported.

Example 8.
Consider the file called `prob-mixed.tex` that contains the following:

```
\newproblem*{oop}{%
  % This is an essay style question.
  Describe what is meant by object-oriented programming.%
}

\begin{defproblem}{inheritance}
  % This is an essay style question.
 Describe what is meant by the term \emph{inheritance} in
 object-oriented programming. Use examples.
\end{defproblem}

\begin{defproblem}{weightedcoin}%
  \begin{onlyproblem}
    A coin is weighted so that heads is four times as likely
    as tails. Find the probability that:
    \begin{textenum}
      \item tails appears,
      \item heads appears
    \end{textenum}%
  \end{onlyproblem}%
  \begin{onlysolution}
    Let $p=P(T)$, then $P(H)=4p$. We require $P(H)+P(T)=1$,
    so $4p+p=1$, hence $p=\frac{1}{5}$. Therefore:
    \begin{textenum}
      \item $P(T)=\frac{1}{5}$,
      \item $P(H)=\frac{4}{5}$
    \end{textenum}
  \end{onlysolution}
```

```
\end{defproblem}

\begin{defproblem}{validprobspaces}
\begin{onlyproblem}%
Under which of the following functions does
$S=\{a_1,a_2\}$ become a probability space?
\par
\begin{textenum}
\begin{tabular}{ll}
\item $P(a_1)=\frac{1}{3}$, $P(a_2)=\frac{1}{2}$
&
\item\label{validprobspacescorrect1} $P(a_1)=\frac{3}{4}$,
$P(a_2)=\frac{1}{4}$
\\
\item\label{validprobspacescorrect2} $P(a_1)=1$, $P(a_2)=0$
&
\item $P(a_1)=\frac{5}{4}$, $P(a_2)=-\frac{1}{4}$
\end{tabular}
\end{textenum}
\end{onlyproblem}%
\begin{onlysolution}%
\ref{validprobspacescorrect1} and \ref{validprobspacescorrect2}%
\end{onlysolution}
\end{defproblem}

\begin{defproblem}{digraph}
  % This problem requires the tikz package
  \begin{onlyproblem}\label{ex:digraph}
  Identify, if any, the sinks and sources of the digraph shown
  in Figure~\ref{fig:digraph}.

  \begin{figure}[tbh]
    \centering
      \begin{tikzpicture}[every node/.style={draw,circle}]
         \path (0,0) node (A) {$A$}
               (1,0) node (B) {$B$}
               (0,1) node (C) {$C$};
         \draw[->] (A) -- (B);
         \draw[->] (B) -- (C);
         \draw[->] (A) -- (C);
      \end{tikzpicture}
    \par
    \caption{Digraph for Question~\ref{ex:digraph}}
    \label{fig:digraph}
```

```
  \end{figure}
  \end{onlyproblem}
  \begin{onlysolution}
  $A$ is a source and $C$ is a sink.
  \end{onlysolution}
\end{defproblem}
```

This contains a mixture of `\newproblem` and defproblem. It also has comments and spaces to make the code more readable. As can be seen in Figure 4.4 these have gone in the import.

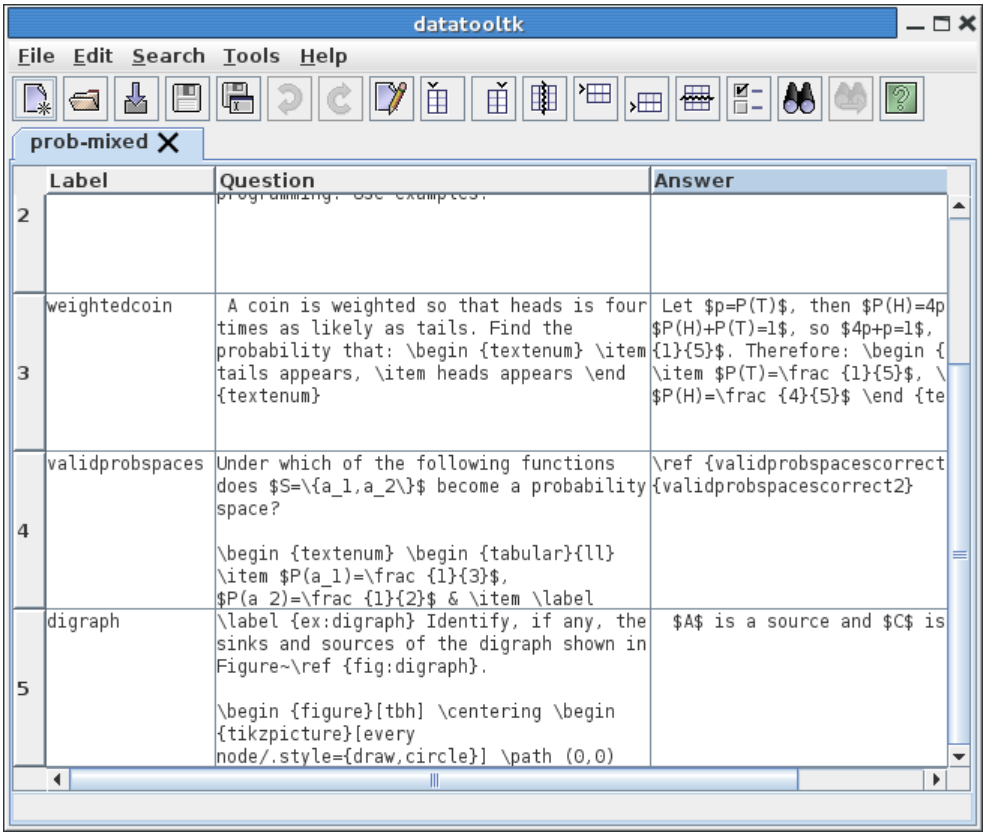

Figure 4.4: Pretty Printing and Comments are Lost When Importing Data from probsoln

See also:

- Shuffling the Data
- Sorting and Shuffling

5 Templates

Templates that come with `datatooltk` are located in the `resources/templates` subdirectory of the `datatooltk` installation directory. You can also write your own templates and store them in the user templates directory (see section 5.1). Each template defines a set of column headers. To create a new database with a particular set of column headers, use the File→New From Template menu item, which opens the dialog box shown in Figure 5.1.

The `datatooltk` application comes with the following templates: `datagidx` (creates a database with the same structure as used by the datagidx package) and `people` (creates a database suitable for storing records about people, including columns for forenames, a surname, title and address.) For example, Figure 5.2 shows a database created from the `people` template.

Rows can now be added to this database using the Edit→Row menu or via corresponding plugins (see section 3.4).

5.1 Writing a Template File

If you want to write your own template, you need to create an XML file and store it in a subdirectory of the `datatooltk` user properties directory (see chapter 6) called `templates`. You will need to create this directory, if it doesn't already exist. For example, on a UNIX-like system, the user template directory will be `~/.datatooltk/templates/`. The template file must have the extension `.xml` for it to be listed in the "New From Template" dialog box. (The base name of the file is used in the list.)

The template file must have one `<datatooltktemplate>` element. This element may contain one or more `<header>` elements. Each `<header>` element must contain one `<label>` element and optionally one `<title>` and/or one `<type>` element.

The `<label>` element contains the uniquely identifying header label. The `<title>` element contains the header title. If omitted, the title is set to the label, unless there

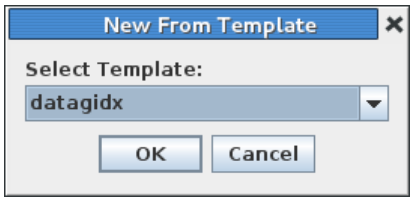

Figure 5.1: New From Template Dialog

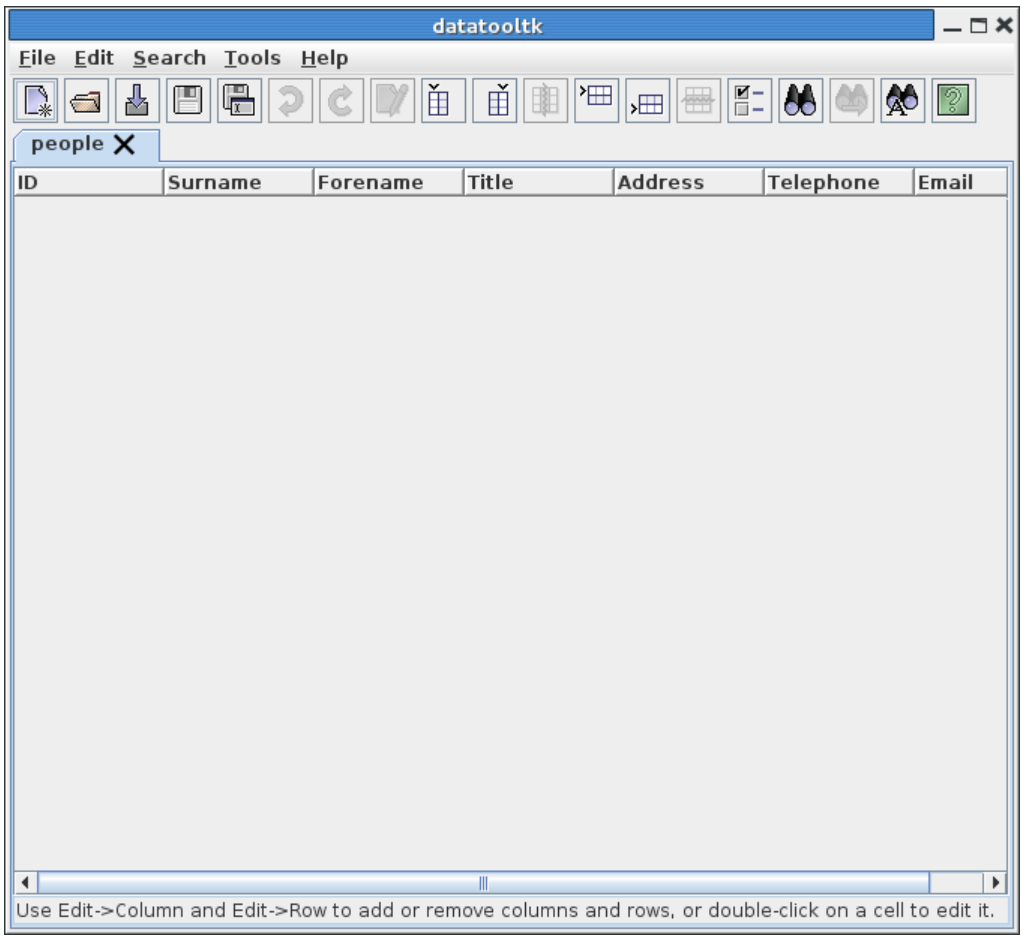

Figure 5.2: New Database Created from people Template

is an entry in the resource dictionary file that matches `plugin.`⟨*template name*⟩`.`⟨*label*⟩, in which case that property is used. The `<type>` element must be one of: `-1` (unknown type), `0` (string type), `1` (integer type), `2` (real type) or `3` (currency type). If omitted the type is set to `-1`.

Example 9.
Suppose I want to write a template to create a database for a list of products. The database needs three columns: one for the product name, one for the product code and one for the product price. The name should be a string, the price column could either be set to "real" if you don't need to worry about the currency unit or "currency" if you need a currency unit for each product. Let's suppose that the code must be an integer. Here's a template file (the price column is set to "real" rather than "currency"):

```
<datatooltktemplate>
  <header>
    <label>Name</label>
    <type>0</type>
  </header>
  <header>
    <label>Code</label>
    <type>1</type>
  </header>
  <header>
    <label>Price</label>
    <type>2</type>
  </header>
</datatooltktemplate>
```

6 Application Properties

When `datatooltk` is run, either in batch or GUI mode, the application settings are read in from the user properties file, if it exists. Any command line options override those settings. If `datatooltk` is run in GUI mode, the application properties are saved on exit. They are not saved in batch mode.

The user properties directory depends on the operating system. On Windows, it is a folder called `datatooltk-settings` in the folder given by the Java system property `user.home`. This is usually the user's home folder but in some versions of Java this can be `%userprofile%`. On other operating systems, the user properties directory is called `.datatooltk` and is in the user's home directory.

In GUI mode, the settings can be changed using Edit→Edit Preferences. This opens the **Preferences** dialog box, which has the following tabs:

General (Figure 6.1)

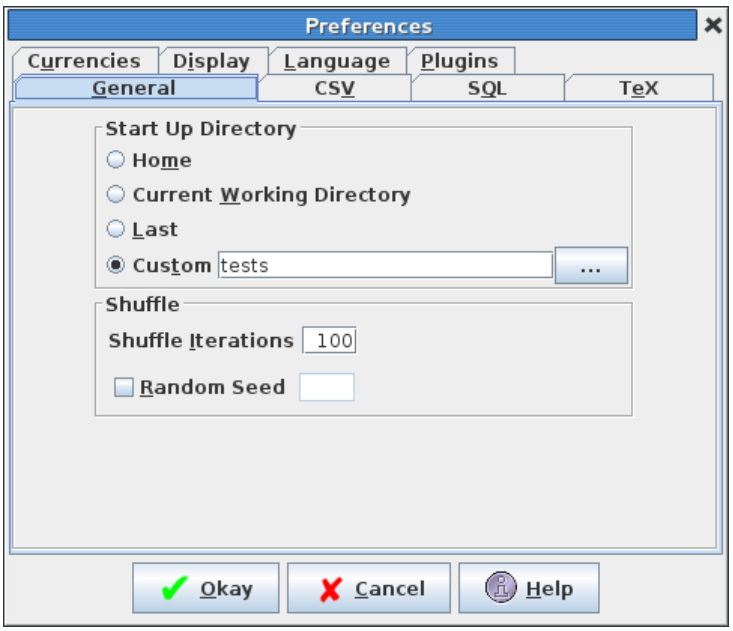

Figure 6.1: General Tab

In this tab you can specify the start up directory. (The default directory when you first load, save or import data via the File menu.) You can set this to your

home directory, the current working directory, the directory you last used on the previous run of `datatooltk` or you can specify a directory of your choice.

In this tab you can also specify the number of iterations to use in a shuffle operation (equivalent to `--shuffle-iterations`) and, optionally, a seed for the random number generator (equivalent to `--seed`).

CSV (Figure 6.2)

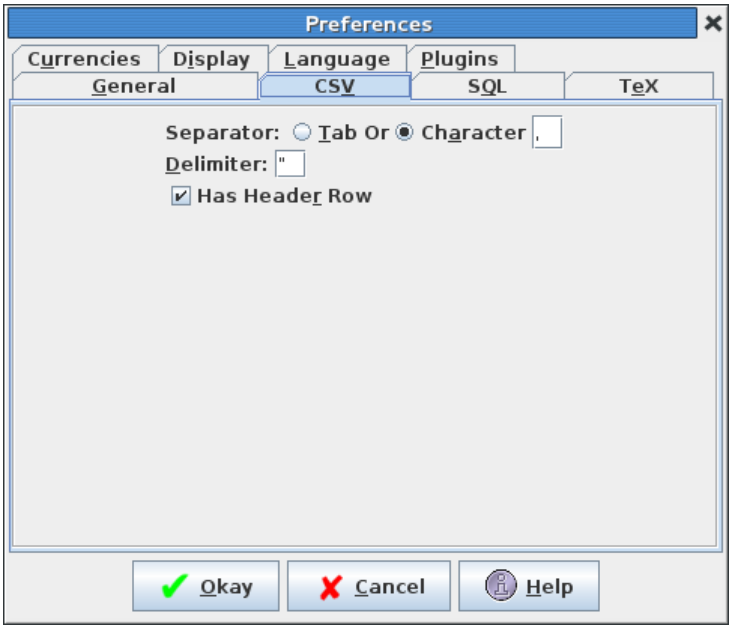

Figure 6.2: CSV Tab

In this tab you can specify the separator character. If the separator is a tab character, select the **Tab** radio button. Otherwise select the **Character** radio button and enter the character in the neighbouring text box. Set the delimiter in the **Delimiter** field. Check the **Has Header Row** button if your CSV files have a header row otherwise uncheck it.

SQL (Figure 6.3)

In this tab, you can specify the SQL connection information. Enter the host name and port number the SQL server is running on in the **Host** and **Port** fields. Currently, the only available prefix is "jdbc:mysql://", which is the JDBC driver for MySQL. If you are using another driver or SQL database, you'll have to add the relevant library to the `lib` directory and add it to the class path used by `datatooltk.jar`. Enter the name of the database you want to connect to in the **Database** field and the associated user name in the **User Name** field. If you want

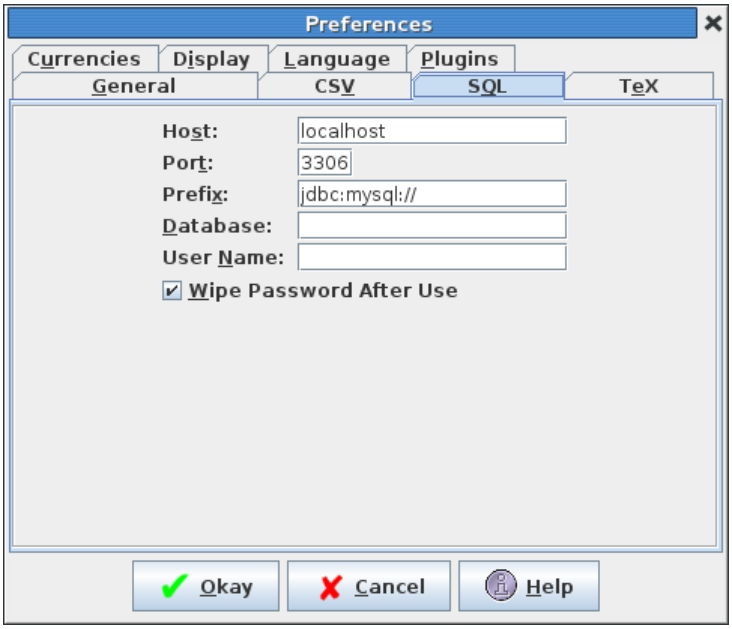

Figure 6.3: SQL Tab

the password wiped from memory as soon as a connection has been made, make sure the **Wipe Password After Use** box has been selected.

TeX (Figure 6.4)

In this tab you can specify whether or not to map TEX special characters when you import data from CSV or SQL. If you want the mapping, make sure the **Map TeX characters when importing data from CSV or SQL** box is checked. If it is checked, the performed mappings are listed in the table in the tab. To add another mapping, click on the **Add** button, which opens the dialog box shown in Figure 6.5.

To remove a mapping, select the unwanted mapping and click on **Remove**. To edit a mapping, select the mapping and click on **Edit**.

LATEX is used to help `datatooltk` import data from a probsoln dataset. If the `latex` executable isn't on the system path, you will have to specify its full location in the **LaTeX Executable** field. You can use the ellipsis button next to the field to browse your filing system.

Currencies (Figure 6.6)

If you want to identify a column as a currency type, you must make sure that `datatooltk` recognises the LATEX command to typeset your currency. Known currency commands are listed in the **Currencies** tab. If you add any currencies to the list, remember to add them in your document as well with `\DTLnewcurrencysymbol`.

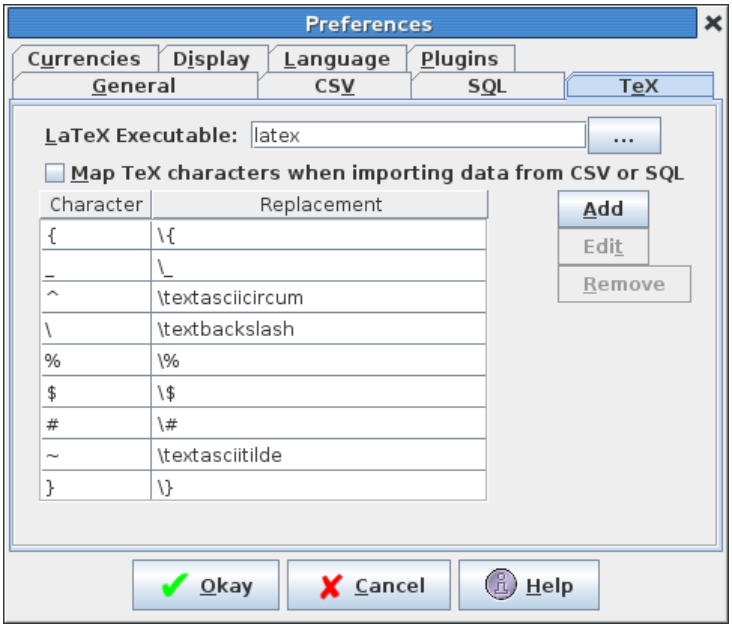

Figure 6.4: TeX Tab

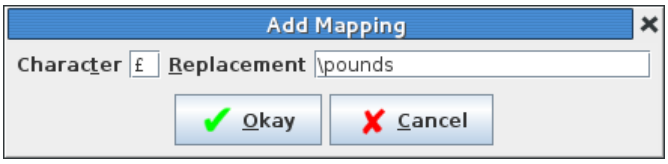

Figure 6.5: Add Mapping Dialog

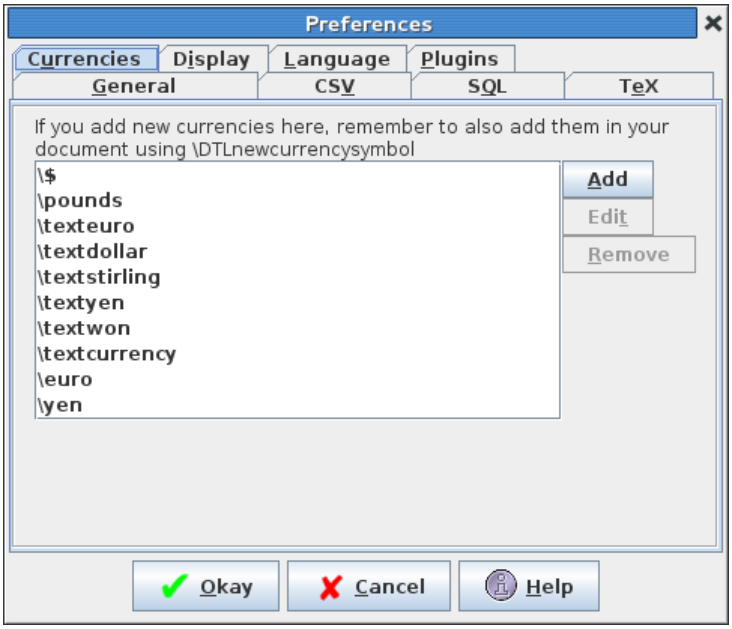

Figure 6.6: Currencies Tab

Display (Figure 6.7)

The default font used in cell entries is a monospaced font. This can be changed using the **Font** drop-down menu. You can also set the font size in the **Font Size** field. By default, each string cell has a maximum of four lines visible in the main window. (Real and integer columns only have a single line visible.) This number can be changed in the **Cell Height** field. Each column has a default width that depends on the data type for that column. The values are listed in the **Cell Widths** area. These can be changed as required.

Language (Figure 6.8)

The language used by the manual accessed via Help→Manual can be set from the **Manual Language** drop-down list. The language used in the messages, menu items, buttons and GUI labels can be set from the **GUI Language** drop-down list. Note that you have to restart `datatooltk` for these changes to take effect.

Plugins (Figure 6.9)

In order to use `datatooltk` plugins, you must have Perl installed (and the Perl Tk module). If the Perl executable is on your path, you can just specify it as `perl` in the **Perl** field of the Plugins tab. If it's not on your path, you will have to specify the full path name in this tab. You can use the ellipsis button to browse your filing system.

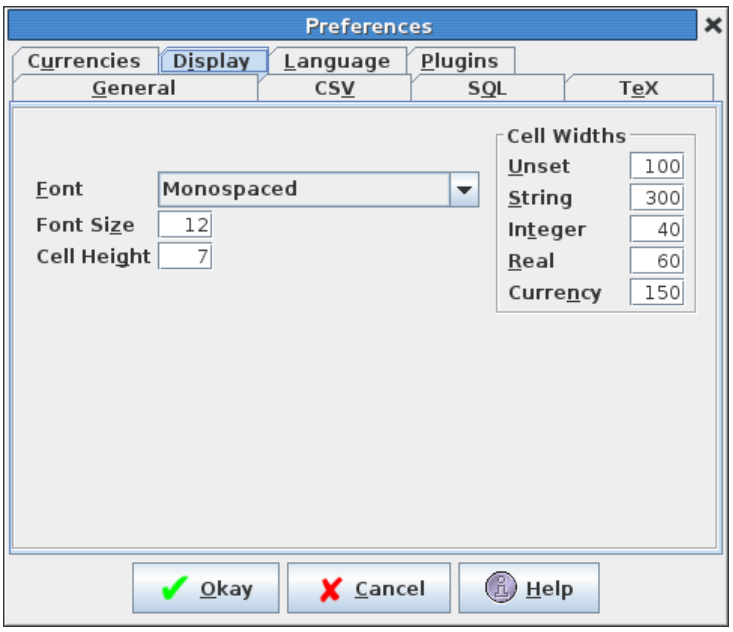

Figure 6.7: Display Tab

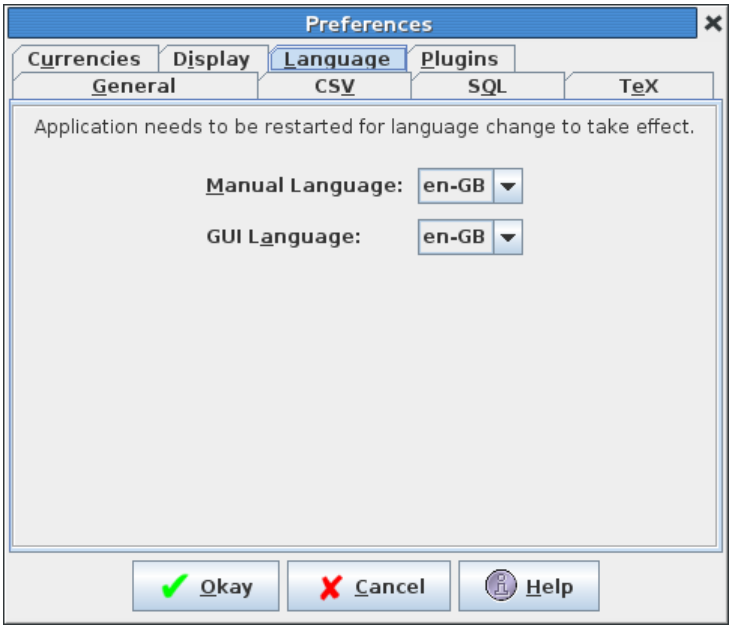

Figure 6.8: Language Tab

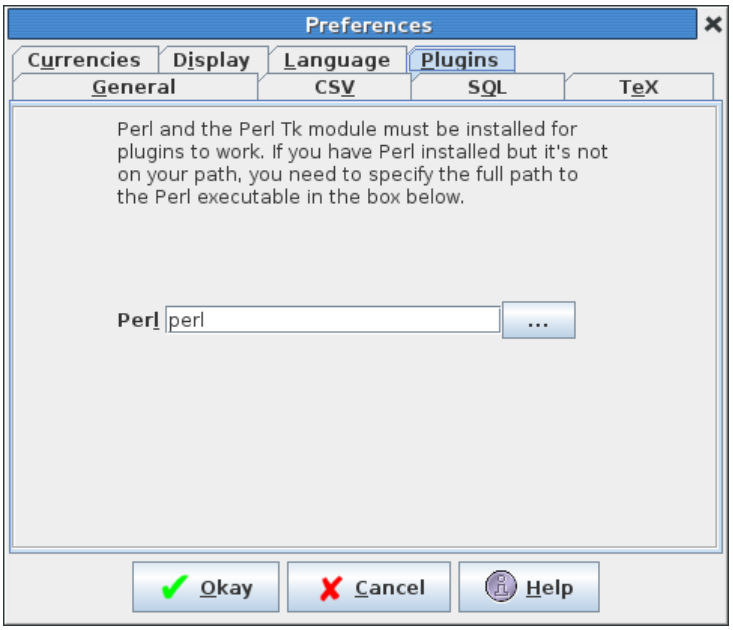

Figure 6.9: Plugins Tab

7 Licence

datatooltk is licensed under the terms of the GNU General Public License. datatooltk depends on the following third party libraries whose jar files are in the lib directory: Java Help (https://javahelp.java.net/), Open CSV (http://opencsv.sourceforge.net/), MySQL connector (http://dev.mysql.com/downloads/connector/j/) and the Java Look and Feel Graphics Repository (http://www.oracle.com/technetwork/java/index-138612.html).

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Lesser General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
```

distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program

is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License. (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you

distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the

operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then

the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free

programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Glossary

current working directory The directory in which the application was started. 50

MySQL An open source SQL database. 5, 41, 50

Acronyms

CSV comma-separated values. 3–5, 7, 40, 41, 50, 51

GUI graphical user interface. 3, 5, 12, 15, 36, 40–42, 49, 53

SQL structured query language. 3–5, 7, 14, 40–42, 50, 51

Index