The Kexi Handbook

This documentation was converted from the KDE UserBase Kexi/Handbook page at 2012-09-14.
Update to 2.4 by the KDE Documentation Team

Contents

List of Tables

Abstract

Kexi is the application for creating databases and for data management in the Calligra productivity suite.

Chapter 1

Introduction

Kexi is a database management application. It can be used for creating databases, inserting data, performing queries, and processing data. Forms can be created to provide a custom interface to your data. All database objects - tables, queries and forms - are stored in the database, making it easy to share data and design.

Kexi is part of the Calligra productivity suite by KDE.

In addition to storing your Kexi databases in files, Kexi can also store your data on a database server. Using a database server allows you to share your database with other people, and also allows more than one person to use the database at one time. The following database servers are supported by Kexi:

- MySQL
- PostgreSQL

More information about Kexi can be found at the Kexi page on the Calligra website, and on the website for Kexi itself.

If you have any questions about Kexi, there are two mailing lists you can use. The Kexi user mailing list can be used to ask questions about using Kexi or about the Kexi project. The Kexi development mailing list can be used to ask questions about the development of Kexi. Further information on how to subscribe to these lists, together with a few other ways of making contact with Kexi developers, can be found here.

> **NOTE**
> This handbook for Kexi 2.5 is meant to be based on the Kexi Handbook for Kexi 1.1. Please coordinate any effort with Jarosław Staniek, email: kexi at kde.org

Chapter 2

Kexi Basics

2.1 Kexi Databases

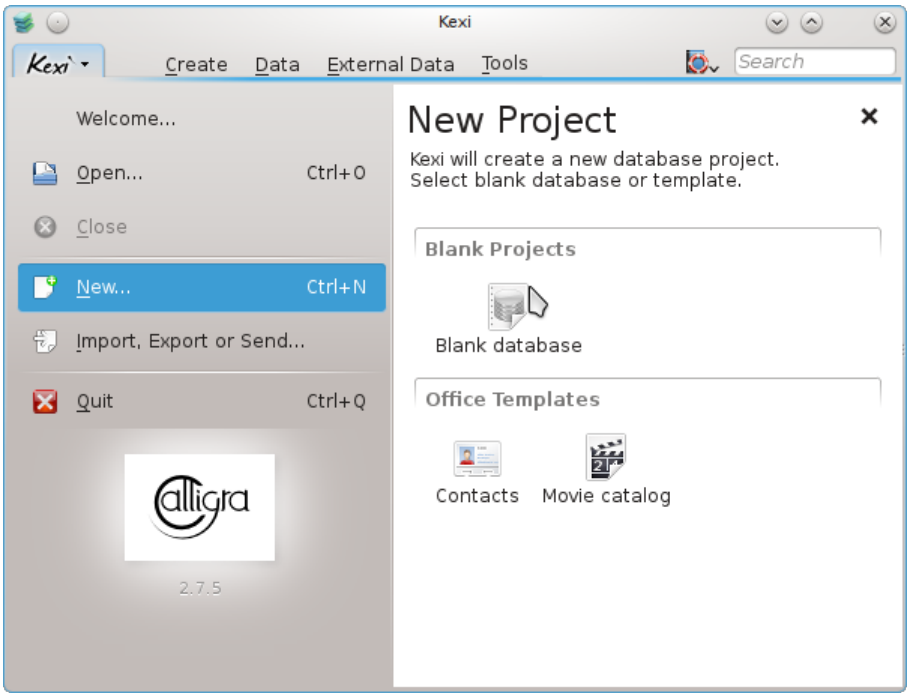
Many applications such as OpenOffice.org®, LibreOffice® or Microsoft® Excel create files which are called documents. Kexi creates files too, but we refer to them as Kexi database files, or simple database files here.

In addition to storing your databases in database files, Kexi can also use databases on database servers, which is why we refer to them as database files, and not simply as databases.

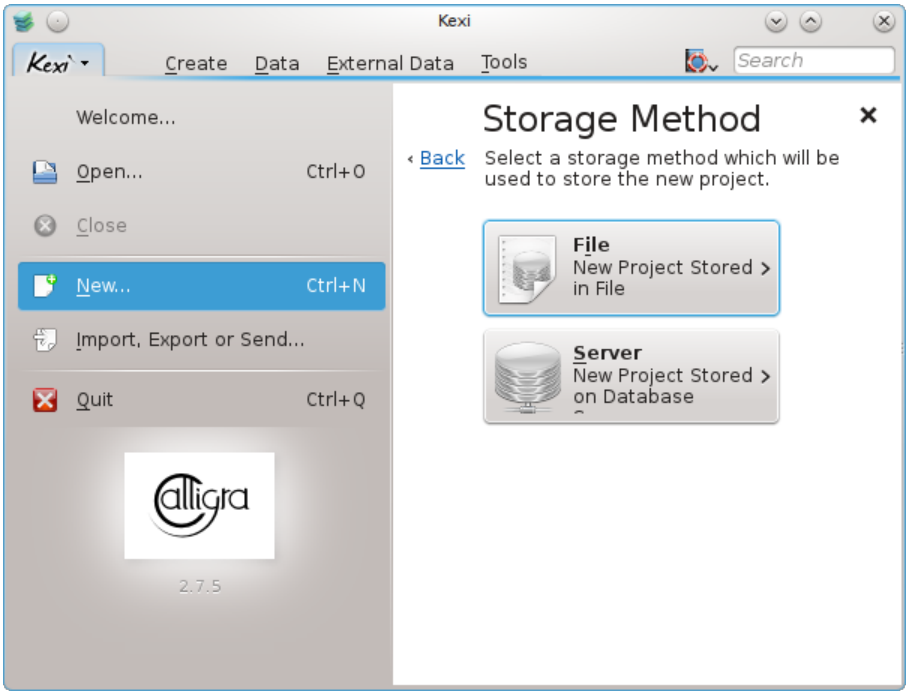
The term Kexi project, or simply project is also used to refer to a Kexi database, regardless of whether it is stored in a file or on a database server.

> NOTE
> Kexi database files usually have the extension `.kexi`

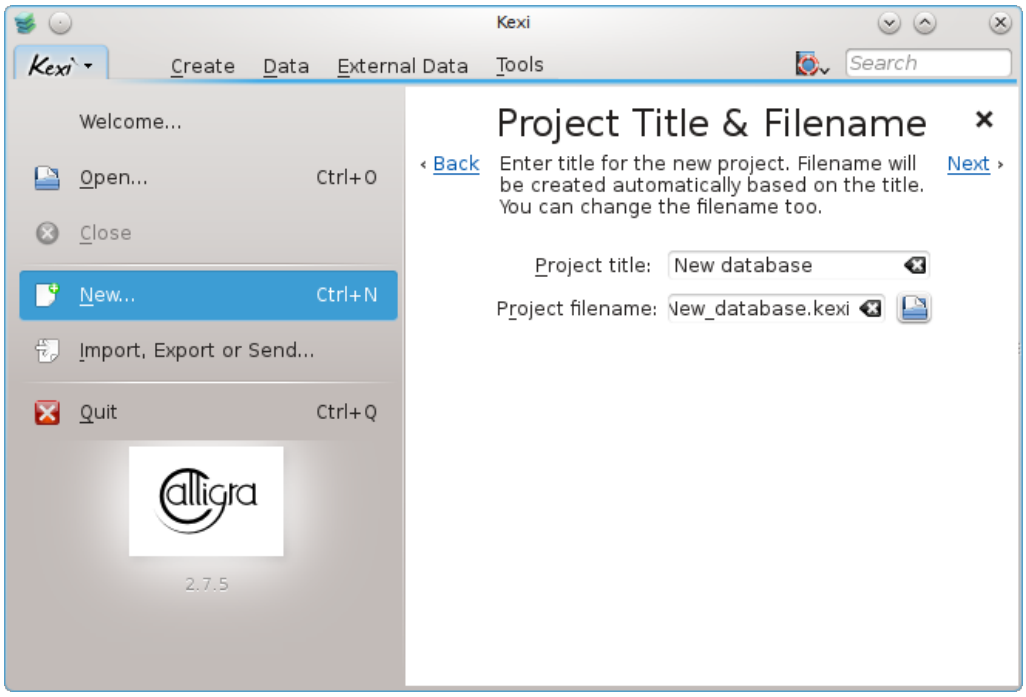
2.2 Creating a New Database File



1. Run Kexi, or if it is already running, use **Kexi** → **New... (Ctrl+N)**.
2. On the **New Project** page, under **Blank Projects** section, choose **Blank Database**.



3. On the **Storage Method** page, click **File**.

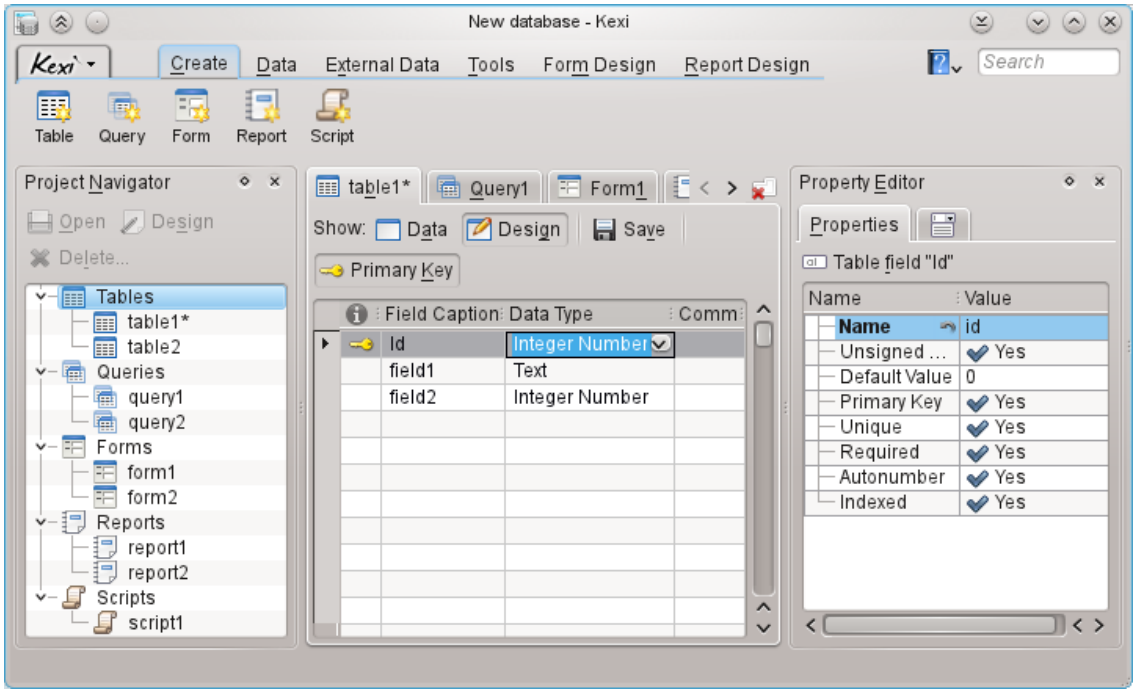


4. On the **Project Title & Filename**, define a title and the filename for the project.
5. Click **Next** to create the project.

NOTE

- When you change the project title, the proposed filename automatically changes accordingly.
- You can use the file browser to choose a folder where you would like to save your database file.

2.3 The Kexi Main Window



The **Tabbed Toolbar** on the top gives access to common actions and commands.

The **Project Navigator** and **Property Editor** are shown in panes on each side of the child window. These can be resized or hidden as required. A pane can be hidden by clicking the small cross at the top of the pane (just below the toolbar).

Database objects (tables, queries, etc.) listed in the **Project Navigator** can be opened by clicking (or double-clicking, depending upon your global KDE settings) on their names.

2.3.1 Main application elements

Main elements of Kexi application's window are:

Tabbed Toolbar

Contains available commands for the application. You will find detailed description of any of the commands in the appendix.

Project Navigator pane

Contains a list of any object (tables, queries, forms, ...) created within the currently opened database project. The navigator also contains small toolbar with most usable commands related to the database objects.

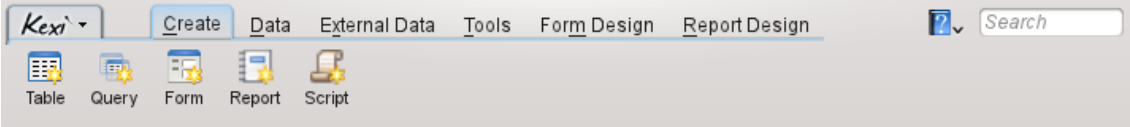
Opened Database Objects Area / Tabbed Windows

A central area of the application taking most of the screen space. The user interface contains switchable tabs with windows that are always maximized.

Property Editor pane

Contains a list of properties of currently activated database object. For certain objects (e.g. form's widgets) it can have several tabs.

2.3.2 Tabbed Toolbar



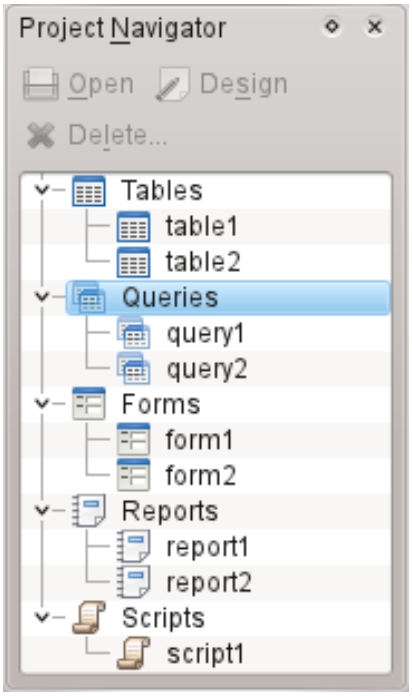
The toolbar is the place that gives you access to most Kexi commands and actions.
Using the actions found in the different tabs on the toolbar you can:

- Create / Open / Close Kexi projects
- Create Database objects
- Import / Export Data

Depending on the context, additional tabs can be visible:

- **Form Design** tab is visible if the Form Designer is actually used.
- **Report Design** tab is visible if the Report Designer is actually used.

2.3.3 Project Navigator pane



The **Project Navigator** pane is one of the most frequently used elements of the Kexi main window. The pane contains a list of all objects created within the currently opened Kexi database project. The objects are split into groups: tables, queries, forms, reports and scripts.

The Project Navigator pane also contains a small toolbar for most frequently used commands (from left to right): **Open** selected object, **Design** selected object, and **Delete** selected object.

For each object on the list a context menu is available using the right mouse button.

Double clicking with the left mouse button on the object's name on the list opens the object in **Data View**. If the object's window was already opened, the action just activates the window without switching it's view mode.

NOTE
Even though your operating system or window manager may be set up to handle single clicks instead of double clicks, Kexi uses double clicks in **Project Navigator** to avoid accidentally opening large datasets or executing queries.

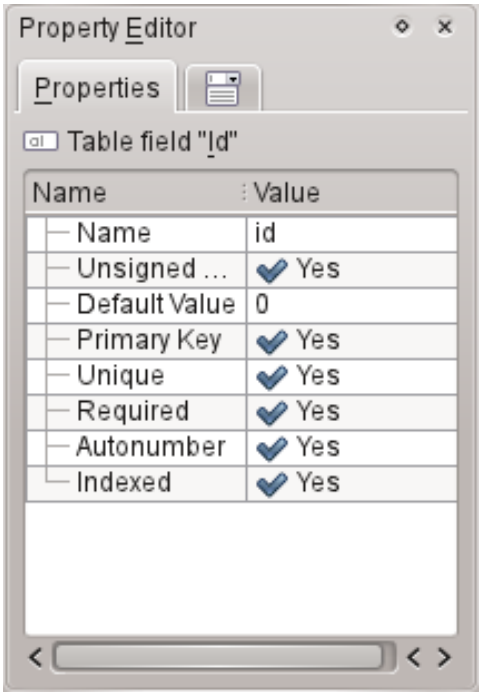
2.3.4 Opened database objects area / Tabbed Windows

Whenever you double click an object in the project navigator, it opens in the **Opened database objects area**. Each window has its own tab in Kexi.

You can rearrange the tabs by drag and drop and close them using the button located at the far right of the tab strip.

NOTE
Sometime later on there will be an option to detach tabs, creating somewhat something similar to an MDI interface. That could be useful for custom solutions or multiple displays.

2.3.5 Property Editor pane



In the **Property Editor** pane you can change properties of the object displayed in the active window. Depending on the context, the pane is consisted of one or more tabs. The first, always visible tab, **Properties**, contains the list of available properties.

Rules for using the **Property Editor**:

- Each row contains a single property.

- You can use the mouse or the keyboard to change values of particular properties.
- Most frequently used types of property values are:
  - a number; you can enter the value directly or increase or decrease its value by clicking with the left mouse button on the arrows.
  - text
  - drop down list of values
  - Yes/No; you can toggle the value by clicking on the button; Yes (true) means that the button is toggled on, No (false) means that the button is toggled off.

NOTE

- There is no need to confirm a changed value: changes are visible immediately after moving to a different row of the Property Editor's list or by pressing the **Enter** key.
- Names of the recently changed properties that not yet were stored in the database are marked with bold text.
- After changing the value of a property, a special **Undo changes** button appears on the right side of the Property Editor's list. By clicking it you can revert the value of the property to the original value that was loaded from the database upon opening the database object. The button is only visible when the property is actually highlighted.

The **Property Editor** pane is empty if:

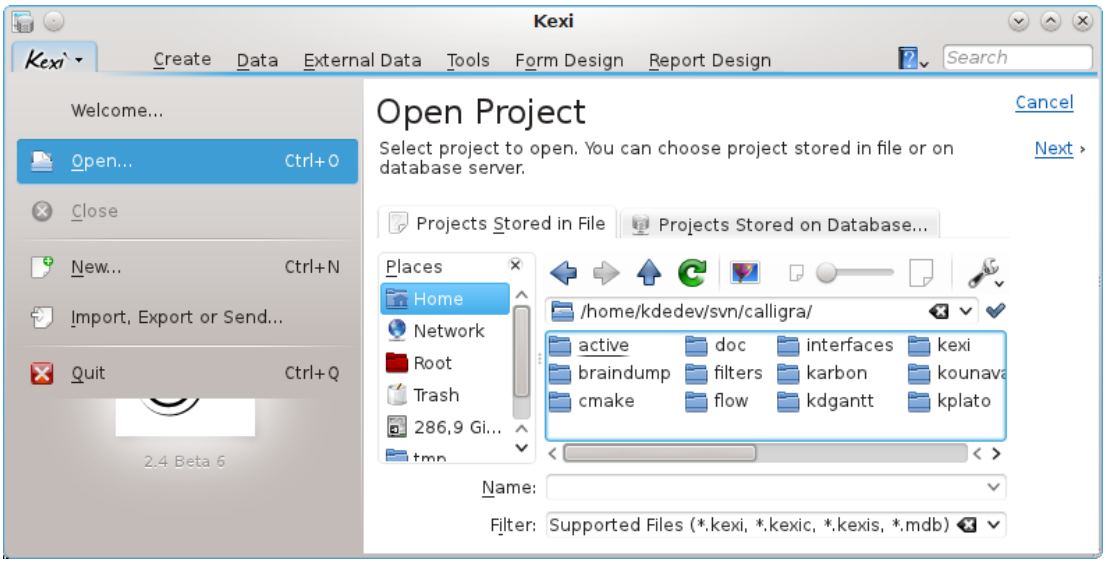
- no single database object's window is opened, or
- the active database object's window does not offer properties; it is usually the case when it is opened in **Data View** instead of **Design View**.

2.4 Opening an existing Kexi database file

There are two ways to open an existing Kexi database file:

2.4.1 Opening a database file in the Open Project dialog

- Run Kexi. You should see the **Welcome to Kexi** startup dialog, which allows you select one of the recently used projects to open.
- If the project is not on the page with recently used projects choose **Open...** (**Ctrl**+**O**). You will see the following dialog:



- From the location bar, pick a folder containing a file you are looking for.
- You can either pick a file which is immediately opened. Alternatively you enter its name in the **Name:** box and click on **Next**.

Notes

- By default the **Filter:** drop down list has **All Supported Files (*.kexi, *.kexic, *.kexis, *.mdb)** selected. In case the file you are looking for has another extension, you can change the selection of the **Filter:** drop down list to **All Files** to display all available files (regardless of an extension).
- If you have selected a file of an external type, like a **MS Access** `.mdb` file, Kexi will provide you with the option to import the file.
- If you have selected a connection data file (with `.kexic` extension) or a shortcut to a project on database server file (with `.kexis` extension), Kexi will display appropriate dialogs.

2.4.2 Opening an existing Kexi database file by clicking on .kexi file's icon

- Click file's icon using your file manager or desktop. Kexi will open this database project automatically.

Notes

Note about database files accessed remotely. You may want to open a database file that is located on a remote source (e.g. a web or FTP server or a MS Windows network share). KDE allows you to open files from remote sources directly in applications and to save changes back to the source, but this is not the case with database files. By clicking on a database file located on a remote source, a copy of the file will be downloaded to a temporary directory on your computer and all your changes will be made to this local file. The remote original of the file will remain unchanged, so it's recommended to copy (download) the file to your computer first, then open the file and copy it back to the remote source if you want to make it up to date.

2.5 Using built-in help

The following ways to get built-in help in Kexi are available:

The Handbook in form of electronic document.

The Handbook is available by pressing **F1** key or click  in the menubar and select **Help**.

***What's This?* hints.**

Select **What's This?** from the menu that pops up when you click  and then click on an area of the application to get hints about it.

Chapter 3

Building Simple Databases

3.1 Introduction

To learn the basics of Kexi, you could build a simple database utilizing most elementary Kexi's features. To make things simpler, advanced database design topics will not be covered here.

Start by creating a new empty Phone Book.

Having a new empty database project, perform the following steps:

1. Design database tables. Read the section called Designing Database Tables.
2. Enter data into tables. Read the section called Entering Data Into Tables.
3. Design database queries. Read the section called Designing Database Queries.
4. Design forms. Read the section called Designing Forms.
5. Use forms to enter data. Read the section called Entering Data Using Forms.

3.2 Designing Database Tables

First, there will be two tables added to your database: `Persons` and `Phones`. These are exactly the same tables as described in chapter Database and Spreadsheet. A layout for `Persons` can be found in section Data integrity and validity in that chapter.

1. Select **Table** from the toolbar. You can also use **Create object: table** in the context menu of the **Tables** item in the Project Navigator.
2. The Table Designer's window will appear. Looking at the top of designer's window you will notice that Kexi proposed you a generic name like **Table1** for the new table. The table design is not saved yet so you will be able to assign more proper name later. Moreover, because of the same reason, the table name is not yet visible in the **Project Navigator**.

3.2.1 The Table Designer window

3.2.1.1 Table Designer window consists of following columns:

-  - Additional Information about the field.
- **Field Caption** - caption of the field which will be displayed during data entering.
- **Data Type** - a combo box containing a list of data types, allowing to set a main rule for entered data for a given field. For example, when an integer number data type is set for a field, a database user will not able to enter letter characters into this field.
- **Comments** - you can enter here any information useful for understanding what the given field is provided for. This additional text will be saved within the table design and only visible in design mode.

In the Table designer window, every row corresponds to a single table field. You can recognize you are in design mode because the **Design** button is toggled on within the Table designer window toolbar.

3.2.1.2 Designing the Persons table

In the first row click on the cell in the **Field Caption** column and enter `Name` as field caption. Filling the **Field Caption** field automatically fills the **Name** field as seen in the **Property Editor** pane.

Notes about field names and captions

- Every table field must have a name and a caption, these cannot be empty.
- Field name is a word used by the database, usually not visible for users of the database application. The name may not contain special (national) characters (like ±, ¶, Ü) or space characters. The name must only contain roman letters, numbers and underscore sign '_'. Use the latter instead of spaces or dashes.
- Field names must be started with a letter or underscore sign '_', never with a number.
- It does not matter whether you are using small or capital letters. For Kexi the field name `Persons` is the same as `persons`.
- Field caption, on the other hand, allows you to enter any letters and special characters. It will be displayed for users of the database application.

In a similar way, enter the following fields into the table design:

- *surname*
- *street*
- *house_number*
- *city*

All the above fields, except *house_number*, are of type text. Change *house_number* field's type to integer number. To do this, click on a cell in the **Data Type** column, *house_number* row and then click on drop down list's button or press **F4**. The list of data types will appear. You can also use the arrow keys **Up** and **Down** to select another type. Select the **Integer Number** type.

From now on, the *house_number* field only accepts numbers.

Persons table design is ready. Click the **Data** button on the toolbar to finish designing and switch to **Data View** for the table. This allows you to enter data into the table.

As the design is not yet saved in the database, the **Save Object As** dialog window appears. You need to specify the name for the new table.

Kexi offers a generic name like **Table1**. To change the name, enter *Persons* into the **Caption** field and press the **Enter** key or click the **OK** button. The **Caption** field will be used to display the table to database end-users, e.g. as a form. Unlike the name, the caption can contain any characters including spaces and special characters.

Note that filling the **Caption** field automatically fills the **Name** field. For your convenience the rule for using only letters, numbers and the '_' character is kept. You can alter the contents of the **Name** field if you want to.

1. You are asked about an agreement for automatic adding of primary key to the table. Click **Add primary key** button to continue.
2. The *Persons* table has been created and opened in Data View. Its name appears in the **Project Navigator** pane.
3. Create the *Phones* table, in a similar way as *Persons* table.
4. Create a *person* field of type **Integer** number and *phone* of type **Text**. Do not use a number type here because phone numbers can have many different forms and prefixes.
5. Click the **Data** button on the toolbar and enter *Phones* caption for the table. As for your previous table, allow Kexi to automatically create a primary key.

3.3 Entering Data Into Tables

You have designed the two tables *Persons* and *Phones*. None of them contain any data yet. You can enter some, and in this chapter you will learn how to do this fast and effectively.

Start with the *Persons* table. Open it in Data View using **Open** in the **Project Navigator's** context menu or the toolbar button. The current cell is marked with (usually black) thicker border, a cell cursor. The contents of the cell, if present, are highlighted with a different color. The current row, i.e. the one you have placed your rectangular cursor in, is marked on the left hand with an arrow symbol.

You can navigate through table cells using the arrow keys, **Page Down**, **Page Down**, **Home**, **End** keys; you can also click with the mouse in a cell to select it.

Initially, after opening the table *Persons*, the cursor is placed in the *Id* column. The column has autonumber property defined, marked with blue (autonumber) text in the last row. It means you

do not have to enter values there by hand when entering data for a new row because the cell will be filled automatically with successive numbers.

Inserting new rows and entering data for them in Kexi is different from the way of doing this in spreadsheets. To enter data for a new row, you need to use the arrow keys or mouse, to move your cursor to the special empty last row marked with a plus sign. Place your cursor in the (second) *name* column and enter a person's name. Also enter surname, street, house number and city. When done, move the cell cursor to the last empty row either by using the **Down** key or by clicking in the last row with the mouse to append a new row.

3.3.1 Details About Actions Available While Entering Data Into Tables

- As soon as you enter the first character, the current row is being edited. A pencil symbol appears on the left side of the data table.
- Double clicking a cell with the left mouse button or pressing **Enter** or the **F2** key also starts editing of the current row.
- Pressing the **Esc** key when the contents of a cell is edited cancels changes made to this cell. However, the pencil symbol will not disappear because you can still move to a different cell of the edited row to change its contents. To cancel changes made to the entire edited row, press the **Esc** key again.
- Instead of pressing the **Esc** key, you can click the **Cancel Record Changes** toolbar button.
- Press the **Shift-Enter** keys to accept changes made to all cells in the currently edited row.

Fill the *Phones* table with data. In the *person* column you need to provide an *Id* number of the person existing in the *Persons* table.

3.4 Designing Database Queries

A database's primary purpose is to store and help extract information you are looking for. Unlike databases written on a paper sheets, Kexi database allows you to specify more search criteria. Results are returned faster even for large data sets. All this is a power of databases, however to be able to perform effective queries in your database, you need to learn how to tell the database what you are looking for.

With database queries you can limit data coming from a table to a predefined set of rows and columns as well as dynamically join data coming from multiple tables.

To see how queries work in practice you will create a contacts query joining data from two tables: *Persons* and *Phones* (designed here and filled with data here).

1. Create a new empty query by selecting **Query** from the toolbar. The design window will appear. The window is split into two areas: query relationships at the top and query columns below.
2. Select the table *Persons* in the drop down list **Table:** located at the top of the window and click the **Insert** button. A graphical representation of the table will appear in the relations area. Do the same for the *Phones* table to insert it too.
3. Add query relationship using mouse drag & drop technique: click the field *id* in the *persons* table, drag it and drop into the *person* field of the *Phones* table. This will join both fields by creating a new relationship.
4. Double-click the *name* field in the *Persons* table, to add the field as a query column. In a similar way, add *surname*, *street*, *house_number*, *city* fields from the *Persons* table and *phone* from the *Phones* table.

5. Query design is now ready for testing. Click the **Data** button on the toolbar, to switch from design to viewing the data provided as query results.
6. Save the query design for later use by clicking the **Save** button on the toolbar. Because the query design has not been saved yet, you will be asked to specify a name for it. Enter `Contacts` text in the caption field and click the **OK** button.

3.5 Designing Forms

3.5.1 Most important terms

Form

A window provided for easy data entry and presentation on the computer screen.

Form's data source

Database table or query providing data displayed in the form. The data source is needed because forms itself are only tools for displaying and entering data, while tables and queries are the source of data. New, empty forms have no data source assigned, so they are not displaying any data from your database unless you assign a data source to them.

Form field

Direct equivalent of a column in a table or query. Most frequently used are fields for displaying text and numbers. Entering a new value or changing the existing value of such a field causes a change in the bound table or query column (after accepting the change).

Form design

Tasks you are performing to define the appearance and functions of the form. To do this, you need to provide data source, insert form fields of various types and place them at the appropriate location.

Form widget

Form's element. Main widget types are:

- Widgets displaying information, e.g. a text box or an image box. Each widget of this type can be bound to a data source field (a table or a query column). Therefore, such widgets are called in short form fields.

- Widgets able to perform a specified action, e.g. a push button that can close the current form. Within other applications this widget type is sometimes called form control because it can perform previously defined action of controlling your database application's behavior.
- Other widgets allowing to enrich a form's appearance, e.g. a "line widget" can visually separate two form areas.

Container widget

A widget that can contain other widgets within its area. For example, frame widget or tab widget are containers. The form's surface itself is a container as well. A command button cannot be called as container because it is not possible to insert a widget inside it. In more complex cases, container widgets can be inserted inside a container, so nesting is possible.

3.5.2 Forms versus tables

In chapter Entering Data Into Tables you learned about how to enter data directly into tables using their data sheet view. However, in many cases forms are better suited for data entry:

- A table can contain too many columns to display them on your screen. A form can display such a data using multiple rows.
- A form allows to visually split data fields into logical groups, thus increasing readability. Labels with additional information can be inserted to give users more hints on how to use the form or what given data fields mean.
- Command buttons can be used within forms for commonly used commands so users can use forms in a similar way as a standalone applications they know.

In data sheet view displaying multi-row data text fields or images is as easy as within forms.

3.5.3 Working with form design

As with table or query design, you are able to use Data View and Design View. Form designing is performed in Design View. We will often refer to the form design window as to *Form Designer*.

1. To create a new empty form, select **Form** from the toolbar. Optionally, you can use the **Create Object: Form** command from drop-down button on the Project Navigator's toolbar or **Create Object: Form** command from the context menu of the Project Navigator.
2. A new frame will appear, you can resize the form by moving the borders. The form is covered with a grid which simplifies accurate positioning of the widgets.

As with table design, Form Designer provides a **Property Editor** pane. To save some space on the screen, the pane has three tabs related to the currently selected form:

The Properties tab

Contains a list of properties for the currently selected widget.

The Data source tab 

Contains properties related specifically to the data source of the currently selected widget or the form itself.

The Widgets tab

Contains a hierarchy of all widgets of the form. The list simplifies widgets lookup by name and navigation between them.

There is information about currently selected widget's name and type displayed on the first and second tab.

Additional toolbars are also available:

- The **Widgets** toolbar used for inserting new widgets into the form. Select **Form Design** to display it.

3.5.4 Using the Widgets tab

The widgets tab in the **Properties** pane provides a list of form widgets and their hierarchy. Each widget is presented within the hierarchy beside other widgets being on the same level (the same parent container). Child widgets (inside containers) are presented using indented names.

Each widget has displayed its name and type. The type has also an icon displayed - the same as the one displayed on the toolbar used while form designing is performed.

NOTE

- Changing the current selection on the list causes appropriate selection on the designed form. This allows for easier widget lookup by name and easier navigation. For example, it is possible to select a widget by name, and then switch to the **Properties** tab to change the widget's properties.
- Keeping the **Ctrl** key pressed while an item on the widgets list is being selected allows to select multiple widgets at a time. Keeping the **Shift** key pressed allows to select entire lists of widgets.

Giving widgets reasonable names can be useful but is not mandatory. Note that widget's name is a property that is not visible to the user of your form. Users will only see a widget text, provided by **Text** property or similar.

3.5.5 Inserting widgets - text fields

Let's create a form providing information about persons, i.e. a form connected it with `Persons` table.

If the form being designed should present data obtained from the database, you need to place appropriate fields on it. To do this, use the buttons on the **Widgets** toolbar. Each button corresponds to a single widget type.

1. Click **Text Box** button on the **Widgets** toolbar.
2. Click on the form surface with the left mouse button. A new text box widget will be placed in the point where you clicked. Before you release you can drag your mouse to specify a desired size for the widget.
3. If needed, move the inserted widget using drag & drop to a desired position. You can resize the widget afterwards by dragging one of the small boxes appearing near its corners. Note that the boxes are only visible when the widget is selected. If you select another widget or the form surface, the boxes disappear.

4. Click the **Text Box** toolbar button again and click on the form surface to insert another widget. Repeat this action once again until you get three more text boxes inserted in your form. For the sake of simplicity we will limit ourselves to five data fields.

NOTE

- There is a context menu available in form's design mode, activated by a right mouse button click the desired widget or the form's surface. The menu offers commands like **Cut**, **Copy**, **Paste**, **Delete** and other, more complex. Keyboard shortcuts are also available for these commands. Some of the commands are only available for certain types of widgets.
- The commands **Cut**, **Copy** and **Paste** makes it possible to move or copy widgets between forms, even between separate database projects.
- Holding the **Ctrl** key down while clicking a widget allows to select multiple widgets.
- Instead of using **Copy** and **Paste** commands, to duplicate a widget within the same form you can hold down the **Ctrl** key while moving the widget. After the mouse button is released, the dragged widget will not be moved but copied in the new location.

3.5.6 Assigning data sources

The fields you inserted have no data source assigned yet, so these are not able to display information from the database. To assign data source, use the (Data Source) tab of the **Property Editor** pane.

The very first step is to specify the form's data source, i.e. a place the displayed data will be fetched from. As mentioned above, you will use table `Persons` as a data source for your new form.

1. Click on the form's surface, as you will alter its properties.
2. Switch to the (Data Source) tab and enter `persons` table name in the **Form's data source** drop down list. Alternatively, you can select this name from the drop down list.

You have assigned form's data source. Now you need to do specify widget's data source.

1. Click the first text field widget at the top of the form.
2. In the (Data Source) tab of the property pane enter field name `name` in the **Widget's data source** drop down list. Alternatively, you can select this name from the drop down list.
3. Click on next text field widget and enter `surname` as the data source.
4. Enter data sources for street, house_number and city text fields in a similar way.

You can now save the form's design (this is not mandatory to test the form in action). To save, click the **Save** toolbar button. Upon saving you will be asked for entering the form's name. Enter `Persons` as caption and click the **OK** button. The form's name will be filled automatically.

This is the right moment for testing your form. Click the **Data** toolbar button. Unless you made a mistake while entering data sources, you should see the form's fields filled with data from the `Persons` table.

NOTE

- If you want to remove widget's data source assignment for a form widget, you can use the button in the **Widget's data source** box.
- Use the (Go to selected form's data source) button to select appropriate table or query in the **Project Navigator**, so you can quickly open a table or query being the data source of the form.

3.5.7 Inserting text labels

To make it easier for the form's user to identify the meaning of every field widget, these should have added text labels with appropriate titles. To create text labels the **Label** widget is used.

Insert three text label widgets onto the form, placing them on the left side of the text fields (or on the right hand if your operating system uses right-to-left layout). On inserting a new label, a text cursor appears at the location where you can enter the desired title. Enter consecutively: `Name`, `Surname`, `Street`, `House Number` and `City`. Additionally, on the top of the form insert another label displaying name of the form, i.e. `Persons`. Enlarge this label's size and increase the font size using **Font** in the **Properties** tab.

3.5.8 Actions

An Action is a single activity isolated in the application, available for the user to execute. It can also be executed automatically as a reaction for a given event (e.g. after opening a form).

3.5.8.1 Assigning actions to form buttons

Many actions can be assigned to form button. The assigned action is executed after button is clicked.
To assign action:

1. Switch to form's Design view if you have not done yet.
2. Select the existing button widget by clicking on it or put a new button widget onto the form. If you inserted a new button, enter its title and press **Enter** key.
3. Click the button widget with the right mouse button to display the context menu.
4. From the context menu select **Assign action...** command.
5. An **Assigning Action to Button** dialog window will appear presenting a list of available actions. One of the actions is selected if the widget already has action assigned. Otherwise the **Action category** list has the **No action** item selected.
6. From the **Action category** list select **Application actions** item. Available application-wide actions will be listed.
7. Select one of the actions on the list (e.g. **Delete Selected object**).

After switching to the form's data view you can try whether the action works.

NOTE

- To remove an action assignment, select the **No action** item from the **Action category** list of the **Assigning Action to Button** dialog window.
- Actions only work in the form's data view. Not every action's assignment is reasonable. For example, the **Font...** action is available in data view, but only if you have a widget selected in the Design view. If you make changes to the font settings the changes are applied to the text of that selected widget.

3.5.9 Widget layouts

In most cases form widgets should be conveniently arranged and aligned. Positioning, aligning and resizing widgets by hand is not easy and these parameters are not adjusted when the user resizes the form. In fact the situation is even worse because you cannot assume a given form requires a given space because users have different font sizes and display resolutions.

Using special tool called widget layouts can help to automatically lay out the form widgets. Widget layout is an action of grouping two or more widgets so these are well positioned and have appropriate sizes.

Using layout in a form improves alignment. Moreover, its space is better used. Text fields are closer to each other, spacing is constant.

3.5.9.1 Size policies for widgets within a layout

Instead of setting a fixed size for your widgets, in Kexi you can choose between various widget's size policies. A size policy is a flexible strategy for controlling how a widget is stretched (or shrunk) depending on other neighbouring widgets and space available within the form.

After putting widgets into on a line will be resized to fit their visible text.

For each widget inserted into the form, there are settings for size policy available in the **Property Editor**. The settings are presented as a group of properties called **Size Policy**.

This group of properties contains:

Horizontal Size Policy
: defining horizontal size of the widget,

Vertical Size Policy
: defining vertical size of the widget,

Horizontal Stretch
: defining strength of activity of the Horizontal Size Policy,

Vertical Stretch
: defining strength of activity of the Vertical Size Policy

3.5.9.2 Values of size policies

The following values are available in the drop down list for **Hor. Policy** and **Vert. Policy** visible in the **Property Editor**:

Fixed
: this value means that the widget cannot be automatically resized; it should maintain the constant size defined at design time (width or height),

Minimum
: this value means that the original size of the widget is set as minimal allowed, it is sufficient and there is no need for expanding the widget, but the widget will be expanded if needed. This type of policy can be used to force the widget to be expanded to the whole width or height, especially if you set a stretch value greater than 0.

Maximum
: this value means that the original size of the widget is set as maximum allowed and can be decreased without breaking the widget's usability and readability if other widgets need more space,

Preferred
: this value means that the original size of the widget is the best and preferred; the widget can be shrunk or expanded however and it will stay readable,

Expanding
: this value means that the original size of the widget is reasonable but the widget can be also shrunk; it can be expanded as well to take as much space as possible,

Minimum Expanding
: this value means that the original size of the widget is allowed; it can be expanded to take as much space as possible,

Ignored
: this value means that the original size of the widget is ignored; the widget can be expanded to take as much space as possible but other widgets usually will not allow for that

Different widget types have various default size policies; for example, button widgets have default size policy set to **Minimum** (in both directions), while text field widgets have vertical size policy set to **Fixed**.

The most frequently used size policies are **Preferred**, **Minimum** and **Maximum**.

3.5.9.3 Vertical and horizontal stretch

The **Vert. Stretch** and **Hor. Stretch** properties accept integer values greater than or equal to 0. These properties allow to fine-tune the behavior of size policies. The default value for the properties is 0. A higher value of the stretch means that the widget will be expanded more than widgets for which a lower stretch value is set.

3.6 Entering Data Using Forms

Data entering and editing is usually the task of the user of the database application. The designer of the database should check the form in terms of valid data entry, and see whether the form works as expected.

To test your form, switch to its data view. A single database row (record) of data will be displayed . You can move between fields using the left mouse button or the **Tab** and **Shift**-**Tab** keys. While editing, there will be a pencil icon visible near the record navigator. After entering the row's (record) data you can press the **Shift**-**Enter** keys or click the **Save Record** toolbar button to accept changes made to the current row. Clicking the **Cancel Record Changes** toolbar button discards changes made to the current row and restores the contents of the data fields. You can use the record navigator's button to move to a new row. All the navigator's functions are also available in a similar way as in the data table view.

Chapter 4

Configuring Kexi

4.1 Docking and undocking side panels

The **Project Navigator** and **Property Editor** side panels may be undocked by either:

- Double-clicking on the 'grip' bar at the top of the panel; or
- Clicking once on the diamond button at the top of the panel, next to the x.

Once undocked, panel windows may be docked into the main window again similarly to undocking:

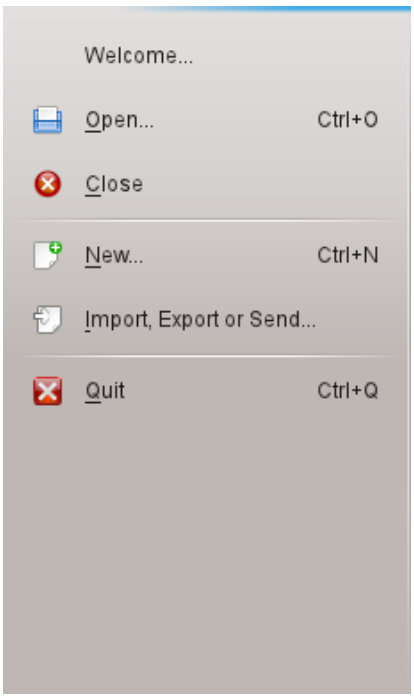
- Double-clicking on the 'grip' bar at the top of the window; or
- Clicking once on the diamond button at the top of the panel, next to the x.

> TIP
> When undocking a panel, Kexi remembers the last docked position so when you dock the panel again it will be placed at the location it was last docked.

Chapter 5

Command Reference

5.1 The Kexi Tab



The Kexi tab is the place where you interact with Kexi's projects.

Selecting an action will open a screen right next to the menu, offering options about the action you chose.

The options offered are:

Welcome
: Here you can select to open a project you recently worked on.

Open
: Here you can select to open a Kexi Project whether stored in a file or a database server.

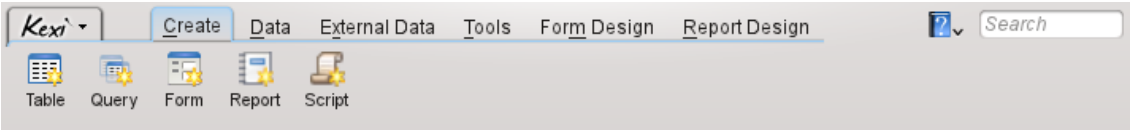
Close
: This action closes the Kexi project you have currently open.

New
: Here you can follow the wizard to create a new Kexi Project.

Import, Export or Send...
: Here you can follow the wizard to Import existing data to the currently open Kexi Project.

Quit
: Closes Kexi.

5.2 The Create Tab



From the **Create** tab you can create objects that will be added to your project.

Table
: Selecting **Table** will take you to the table designer at the design view to allow you to add field definitions to your table.

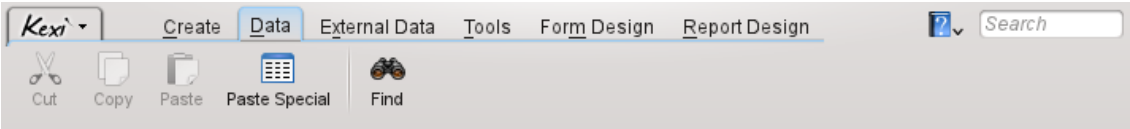
Query
: Selecting **Query** will take you to the query designer at the design view to allow you to design a query to get custom results from your tables in your project.

Form
: Selecting **Form** will take you to the form designer at the design view to allow you to add widgets to your form.

Report
: Selecting **Report** will take you to the report designer at the design view to allow you to add widgets to your report.

Script
: Selecting **Script** will take you to the script editor to allow you to add custom code for your project and / or objects.

5.3 The Data Tab



From the **Data** Tab you can manipulate data in your tables or widgets in your objects (forms, reports, etc)

Cut

Cut, places the data/widget on the clipboard and removes it from it's current position.

Copy

Copy, places the data/widget on the clipboard without removing it from it's current position.

Paste

Paste, places the data/widget found on the clipboard to the table/object accordingly.

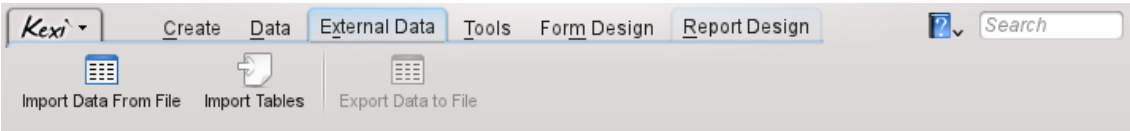
Paste Special

Paste Special, is used to place arbitrary data found on the clipboard to a table, in a way that is predictable so as to be correctly added according to the table's definition.

Find

Find, opens the **Find** dialog so as to search for specific text in the database data.

5.4 The External Data Tab



From the External Data Tab you can you can import data from other sources into your Kexi project.

Import Data From File

Import Data From File, displays a dialog that allows you to import data from CSV or plain text files to a new table in your project.

Import Tables

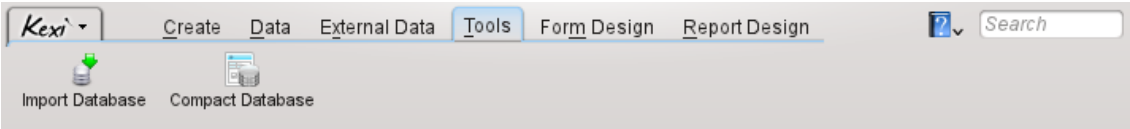
Import Tables, opens the **Table Importing Wizard** that allows you to import data either from a `fods`, `mdb`, `ods`, `tsv` file or from another database server, to a new table in your project.

Export Data to File

Export Data to File, displays a dialog that allows you to export the current tables data to a plain text or CSV file.

> NOTE
> The table has to be open, in order to be able to export data from it to a file.

5.5 The Tools Tab



From the **Tools** tab you can you can manipulate your projects.

Compact Database

Compact Database, checks for minor error in the database and reduces the database's size.

> NOTE
> You should regularly compact your database to keep it in good shape, especially after extensive records operations (mass add, delete)

5.6 The Form Design Tab

From the **Form Design** tab you can you select widgets to add to your form.

Pointer

Pointer, switches to the widget selection mode.

> NOTE
> Selecting any widget, will switch to the widget add mode. Then you can click anywhere on the form to place the widget.

Label

A **Label** widget displays predefined information on a form. Usually it is used as a caption next to other data-aware widgets.

Text Box

A **Text Box** is a single line container for data contained in your table.

Text Editor

A **Text Editor** is a multiline container for data contained in your table.

Combo Box

A **Combo Box** displays a list of options to choose from.

Check Box

A **Check Box** holds two or three states of data (e.g. On/Off)

Image Box

An **Image Box** holds an image, bound to a field in a table.

Button

A **Button** allows you to define actions to be executed upon clicking on it.

Frame

A **Frame** is used as a container for other widgets.

Group Box

A **Group Box** is used to group other widgets and control their state.

Tab Widget

A **Tab Widget** is used as a container for other widgets and can have many pages that contain different widgets.

Line

A **Line** is used as a logical separator between different parts of a form.

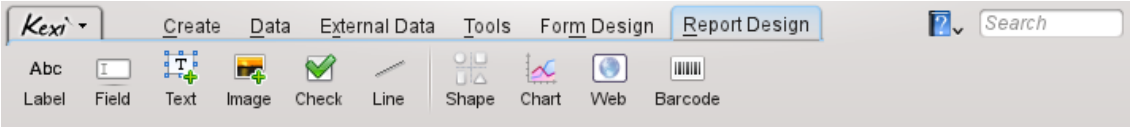
Web Browser

A **Web Browser** is a widget that allows to display a web page inside the form.

Assign Action

Assign Action is used to assign an action to be executed when an event occurs (e.g. clicking on a button).

5.7 The Report Design Tab



From the **Report Design** tab you can you select widgets to add to your report.

Label

A **Label** widget displays predefined information on a report. Usually it is used as a caption next to other data-aware widgets.

Field

A **Field** widget is a single line container for data contained in your table.

Text

A **Text** widget is a multi line container for data contained in your table.

Image

An **Image** widget holds an image, bound to a field in a table.

Check

A **Check** widget holds two or three states of data (e.g. On/Off)

Line

A **Line** is used as a logical separator between different parts of a form.

Chart

A **Chart** widget is used to add a visual representation of your data presented on a graph.

Web

A **Web** widget is used to provide a minimal web browser component and print information from a local or web site on a report.

Barcode

A **Barcode** widget is used to create a barcode to be printed on a report, from data held in a field.

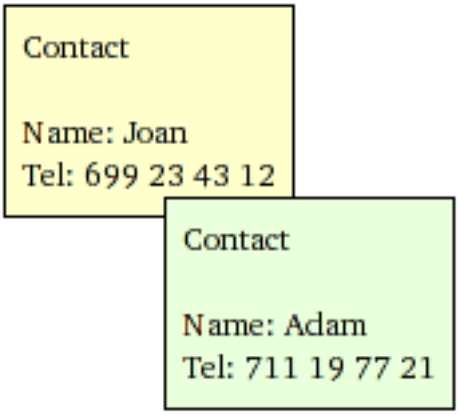
Chapter 6

Appendix A. Introduction to Databases

6.1 What Is a Database?

You can define a database as a collection of data on one topic. It is organised in a way allowing to easily browse the information, make changes or add new items.

Look at this diagram for one of the above examples: a simple phone book.



The above picture shows a set of two contacts each of which is presented on a separate card. It appears that such a card can constitute a single row in a table:

Name	**Tel No.**
Joan	699 23 43 12
Adam	711 19 77 21

Table 6.1: Contacts table

Terms and definitions: A single data which constitutes a part of a greater collection can be called a *row* or more professionally a *record*. The collection is normally called a *table*. Moreover, the most natural name for the table is one describing the data it offers/stores which is Contacts. Furthermore, each row in the table consists of columns often also called *fields*. In the table `Contacts` there are two columns (fields): **Name** and **Tel No.**.

For simple uses a single table can make up a database. Many people consider these two equivalent. As you will see, for real databases we usually need more than one table.

To sum up, you have already got a simple database with one table `Contacts`.

NOTE
Check content at https://www.zoho.com/creator/database-software-vs-spreadsheet.html

6.2 Database and Spreadsheet

It is very likely that you have already used spreadsheet applications like **Calligra Sheets**, **LibreOffice Calc** or Microsoft® Excel. If so, you will probably wonder: since both spreadsheets and databases have tables, why should I use the latter?

While comparing spreadsheets and databases you may encounter the following issues which you will later see in greater detail.

6.2.1 How Is a Database Different From a Spreadsheet?

Gradually exceeding the capacity of a mobile phone, expand your table `Contacts` adding a column (field) `Address`. Add more telephone numbers (office, home) for each person and add surnames to names. To make it simpler we assume the following:

- The table is limited to two people (obviously, there could be hundreds and thousands of them in a real database)
- There are no two persons with the same name and surname

Name and surname	Tel	Address
Joan Smith	699 23 43 12	Western Gate 1, Warsaw
Adam Willson	711 19 77 21	London, Frogs Drive 5
Joan Smith	110 98 98 00	Western Gate 1
Smith Joan	312 43 42 22	Warsaw, Western Gate 1
ADAM Willson	231 83 02 04	Frogs Drive 5, London

Table 6.2: Contacts table

Such a table can be made both in a spreadsheet and in a database. Using a spreadsheet is very easy, of course. What problems do we encounter at this stage?

6.2.2 Referential data integrity

Suppose you are using a spreadsheet and you need to change the address of at least one person. You have a small problem: you often have to change the address in many rows. For example, Joan takes three rows. A real problem will arise if you forget to change one of the rows - the address assigned to this person will be ambiguous, hence your data loses integrity.

Moreover there is no simple way of deleting a chosen person from the table since you have to remember about deleting all rows related to him or her.

6.2.3 Data redundancy

This is directly connected to the previous problem. In fields Name and surname and Address the same data is entered many times. This is typical of a spreadsheets' ineffective way of storing data because the database grows unnecessarily, thus requiring more computer resources (larger size of data and slower access).

How can you solve these problems with a database? You can split information into smaller chunks by creating an additional table *Persons* with only two columns: *Name and surname* and *Address*:

Name and surname	**Address**
Joan Smith	Western Gate 1, Warsaw
Adam Willson	Frogs Drive 5, London

Table 6.3: Persons table

Each row in the table *Persons* corresponds to a single person. Table *Contacts* is from now on a relation to the table *Persons*.

6.2.4 Data integrity and validity

Note the way data is entered in the fields *Name and surname* and *Address*. People entering data can be fallible, sometimes even negligent. In our sample data we have both different sequence of entering name and surname (Joan Smith and Smith Joan; Adam and ADAM) and many more ways of entering the same address. Surely you can think of many other ways.

The above problem shows that e.g. when searching the telephone number of a person whose address is 'Western Gate 1, Warsaw' you will not get a full result. You will get only one row instead of three. Moreover You will also not find all the telephone numbers searching for the value 'Joan Smith' in the field *Name and surname*, because 'Smith Joan' will not fit to 'Joan Smith'.

How can you solve these problems using a database? You can do this by changing the design of the table *Persons* by:

1. Dividing data in the field **Name and surname** into two separate fields: *Name* and *Surname*.
2. Dividing data in the field *Address* into three separate fields: *Street*, *House number* and *City*.
3. Guaranteeing data correctness: by ensuring that no fields are empty, e.g. you must always enter house number.

A modified table looks something like this:

Name	Surname	Street	House number	City
Joan	Smith	Western Gate	1	Warsaw
Adam	Willson	Frogs Drive	5	London
Conditions				
required field	required field	required field	required field	required field

Table 6.4: Persons table

Thanks to introducing the condition required field we can be sure that the entered data is complete. In case of other tables you may of course allow omitting certain fields while entering data.

6.2.5 Limiting data view

A spreadsheet displays all rows and columns of the table which is bothersome in case of very large data sheets. You may of course filter and sort rows in spreadsheets, however you must be extra careful while doing so. Spreadsheet users are in risk of forgetting that their data view has been filtered what can lead to mistakes. For example, while calculating sums you may think you have 100 rows of data while in fact there are 20 rows more hidden.

If you want to work on a small subset of data, e.g. to send it for others to edit, you can copy and paste it to another spreadsheet and after editing paste the changed data back to the main spreadsheet. Such 'manual' editing may cause data loss or incorrect calculations.

To limit the data view database applications offer queries, forms and reports.

A very practical way of limiting is the following extended version of the previously described table `Persons`:

Name	Surname	Street	House number	City	Income
Joan	Smith	Western Gate	1	Warsaw	2300
Adam	Willson	Frogs Drive	5	London	1900

Table 6.5: Persons table

Let's assume that the newly introduced column Income contains confidential data. How can you share e.g. contact details of the persons with your coworkers but without revealing their income? It is possible if you share only a query and not the whole table. The query could select all columns except for the column Income. In database world such a query is often known as a view.

6.2.6 Performance and capacity

Your computer is probably quite fast, however you will easily see that it doesn't help with slow, large spreadsheets. Their low efficiency is first of all due to lack of indexes accelerating the process of data search (databases do offer them). Moreover if you use things like system clipboard, even copying data may become troublesome with time.

Spreadsheets containing large data sets may take ages to open. A spreadsheet loads lots of data to the computer's memory while opening. Most of the data loaded are probably useless/unnecessary for you at the moment. Databases unlike spreadsheets load data from computer storage only when needed.

In most cases you will not have to worry how the database stores its data. This means that unlike spreadsheets, databases do not care about:

- The sequence of rows since you can order the rows according to your needs. Moreover, you can view the same data in many views with different orders.
- The same goes for columns (fields) of the table.

Together with Limiting data view described in the previous paragraph these qualities constitute the advantage of databases.

6.2.7 Data entry

The latest editions of applications for creating spreadsheets enable you to design data-entry forms. Such forms are most useful if your data cannot be conveniently displayed in tabular view, e.g. if the text occupies too many rows or if all the columns do not fit on the screen.

In this case the very way the spreadsheet works is problematic. Fields for data entry are placed loosely within the spreadsheet and very often are not secure against the user's (intentional or accidental) intervention.

6.2.8 Reports

Databases enable grouping, limiting and summing up data in a form of a report. Spreadsheets are usually printed in a form of small tables without fully automatic control over page divisions and the layout of fields.

6.2.9 Programming

Applications for creating databases often contain full programming languages. Newer spreadsheets have this capability too, however calculations come down to modifying the spreadsheet's fields and simple data copying, regardless of the relevance and integrity rules mentioned in previous paragraphs.

Data processing within a spreadsheet is usually done via a graphical user's interface which may slow down the data processing speed. Databases are capable of working in background, outside of graphical interfaces.

6.2.10 Multiuse

It is hard to imagine a multiuse of one spreadsheet. Even if it is technically possible in the case of the latest applications, it requires a lot of discipline, attention and knowledge from the users, and these cannot be guaranteed.

A classical way to sharing data saved in a spreadsheet with other person is to send a file as a whole (usually using e-mail) or providing a spreadsheet file in a computer network. This way of work is ineffective for larger groups of people - data that could be needed in a particular time may be currently locked by another person.

On the other hand, databases have been designed mainly with multiuser access in mind. Even for the simplest version locking at a particular table row's level is possible, which enables easy sharing of table data.

6.2.11 Security

Securing a spreadsheet or its particular sections with a password is only symbolic activity. After providing a spreadsheet file in a computer network, every person being able to copy the file can try to break the password. It is sometimes not so hard as the password is stored in the same file as the spreadsheet.

Features for edit locking or copy locking of a spreadsheet (or its part) is equally easy to break.

Databases (except these saved in a file instead of a server) do not need to be available in a single file. You're accessing them using a computer network, usually by providing a user name and a password. You are gaining access only to these areas (tables, forms or even selected rows and columns) which were assigned to you by setting appropriate access rights.

Access rights can affect ability of data editing or only data reading. If any data is not available to you, it will not be even sent to your computer, so there is no possibility of making a copy of the data in such easy way as in case of spreadsheet files.

6.3 Database Design

Database design needs careful planning. Note that `Persons` table redesign proposed in section Data integrity and validity can generate problems when the table is filled with data. For example, renaming a field is a simple task, but splitting the `Address` field into separate fields requires careful and tedious work.

To avoid such situations, rethink your database project before you create it in your computer, and before you and others will start to use it. Thus, by investing some time initially, you will most probably save your time on everyday use.

6.4 Who Needs a Database?

6.4.1 Stick to spreadsheets if:

- Your needs are limited and your data will never grow to large volumes (can you actually forecast that now?)
- You are unable to acquire the methodology of database construction. You may however consider either outsourcing this task to someone else or using simpler tools.
- You use complicated spreadsheets and you lack time or money to switch to databases. Think or ask someone whether this does not lead down a blind alley. Don't count on magical tools that would change your spreadsheet (regardless how well made) into a database.

6.4.2 Consider using databases if:

- Your data collection expands every week.
- You often create new spreadsheets, copy within these and you feel that this work is getting more and more tedious. In this case the effort of switching to databases easily pays off.
- You create reports and statements for which the table view of a spreadsheet is not suitable. You can then consider switch to using a database with form views.

6.5 Database Creation Software

So far you have learnt the general characteristics of databases without going into much detail about specific applications for designing them.

The first databases were built together with large mainframe computers in the 60s, e.g. IBM System/360. Those were not the days of PCs, therefore these databases required a highly specialized personnel. Although the old computers' hardware was unreliable, they were immeasurably slower and had less storage capacity, one feature of databases still remains most attractive: the data access by many users through a network.

In the 70s scientists formed the theory of relational databases (terms like: table, record, column (field) and relationality and many others). On the basis of this theory IBM DB2 and Oracle databases were created, which have been developed and used till today. In the late 70s the first PCs were constructed. Their users could (gradually) utilize many types of applications, including those for database construction.

When it comes to large databases in companies, the situation hasn't changed: they still require powerful computers or computer complexes called clusters. This goes, however, beyond the topic of this manual.

In the area of 'accessible' databases with graphic user interface for PCs you can choose from the following:

- DBase - a tool for databases operation for DOS popular in the 80s. Files in DBase format are still used in some specific cases due to their simplicity.
- FoxPro - an application similar to DBase (early 90s). After being taken over by Microsoft, graphic user interfaces were introduced and therefore it is used for creating databases on PCs. This product is still offered, though seems a bit obsolete.
- Microsoft Access - an application for databases (data and graphic interface design) with many simplifications, therefore suitable for beginners, designed in the late 80s, based on 16-Bit Architecture. This product is offered and widely used till today, especially by small companies, where efficiency and multiuser requirements are not very demanding.
- FileMaker - popular application similar to MS Access in simplicity, operating on Windows and Macintosh platforms, offered since 1985.
- Kexi - a multiplatform application (UNIX® /Linux®, Windows, Mac® OS X) designed in 2003, developed according to OpenSource principles, part of the global KDE community, that among other things provide a graphic environment for UNIX® /Linux® systems. A significant contributor to Kexi's development is the OpenOffice Poland company.

Chapter 7

Appendix B. Comparing Kexi to other database applications

7.1 Data types

Although different database applications tend to provide similar functionality, they often use different terminology. For your convenience, this appendix shows how the terminology used in Kexi corresponds to that used by other database applications. Thus, this chapter may be useful when migrating databases from one application to another.

The table below shows how the data types in Kexi correspond to data types in other database applications.

Some of the data types listed here are sub-types of other types. For example, the **Long text** type is a sub-type of the **Text** type. To use a sub-type in Kexi, you should select the corresponding basic type (in this case, **Text**) in the table designer, and then select the sub-type using the **Subtype** setting in the **Property Editor**.

Kexi	**MS Access**	**dBase/FoxPro**	**Paradox**
Text (Text)	Text	Character	Alphanumeric
Long text (Long text)	Memo	Memo	Memo
Date/Time (Date/Time)	Date, Time	Date	DateTime
Integer Number (Integer Number)	Number (Integer)	Numeric	Integer
Big Integer Number (Big Integer Number)	Long Integer	Numeric	Long Integer
Floating Point Number (Floating Point Number)	Single/Double precision number	Float	Number

Table 7.1: Comparison of data types used in Kexi and other database applications

Chapter 8

Appendix C. Reserved words for SQL

The following lists contain words that are used internally by Kexi when dealing with data sources.

> WARNING
> When designing your database you should do your best to avoid using these reserved words because otherwise you might end up having problems with your database file or even corrupt it.

> TIP
> If you still want to use reserved words enclose them with double quotation marks ‘’.

8.1 Kexi SQL Reserved words

This list contains keywords that are reserved for use in Kexi SQL:

- AFTER
- ALL
- ASC
- BEFORE
- BEGIN
- BETWEEN
- BY
- CASCADE
- CASE
- CHECK
- COLLATE
- COMMIT
- CONSTRAINT

- CROSS
- DATABASE
- DEFAULT
- DELETE
- DESC
- DISTINCT
- DROP
- END
- ELSE
- EXPLAIN
- FOR
- FOREIGN
- FULL
- GROUP
- HAVING
- IGNORE
- INDEX
- INNER
- INSERT
- INTO
- KEY
- LIMIT
- MATCH
- NATURAL
- OFFSET
- ORDER
- OUTER
- PRIMARY
- REFERENCES
- REPLACE
- RESTRICT
- ROLLBACK
- ROW
- SET
- TEMPORARY
- THEN
- TRANSACTION
- UNION
- UNIQUE
- UPDATE
- USING
- VALUES
- WHEN

8.2 Kexi SQLite Driver Reserved words

This list contains keywords that are reserved for use by Kexi SQLite Driver:

- ABORT
- ATTACH
- CLUSTER
- CONFLICT
- DEFERRED
- DEFERRABLE
- DETACH
- EACH
- EXCEPT
- FAIL
- GLOB
- IMMEDIATE
- INITIALLY
- INSTEAD
- INTERSECT
- ISNULL
- NOTNULL
- OF
- PRAGMA
- RAISE
- STATEMENT
- TEMP
- TRIGGER
- VACUUM
- VIEW

8.3 Kexi MySQL Driver Reserved words

This list contains keywords that are reserved for use by Kexi MySQL Driver:

- ACTION
- ADD
- AGAINST
- AGGREGATE
- ALTER
- ANALYZE
- ANY
- ASCII

- AUTO_INCREMENT
- AVG
- AVG_ROW_LENGTH
- BACKUP
- BDB
- BERKELEYDB
- BIGINT
- BINARY
- BINLOG
- BIT
- BLOB
- BOOL
- BOOLEAN
- BOTH
- BTREE
- BYTE
- CACHE
- CHANGE
- CHANGED
- CHAR
- CHARACTER
- CHARSET
- CHECKSUM
- CIPHER
- CLIENT
- CLOSE
- COLLATION
- COLUMN
- COLUMNS
- COMMENT
- COMMITTED
- COMPRESSED
- CONCURRENT
- CONVERT
- CUBE
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_USER
- DATA

- DATABASES
- DATE
- DATETIME
- DAY
- DAY_HOUR
- DAY_MICROSECOND
- DAY_MINUTE
- DAY_SECOND
- DEALLOCATE
- DEC
- DECIMAL
- DELAYED
- DELAY_KEY_WRITE
- DESCRIBE
- DES_KEY_FILE
- DIRECTORY
- DISABLE
- DISCARD
- DISTINCTROW
- DIV
- DO
- DOUBLE
- DUAL
- DUMPFILE
- DUPLICATE
- DYNAMIC
- ENABLE
- ENCLOSED
- ENGINE
- ENGINES
- ENUM
- ERRORS
- ESCAPE
- ESCAPED
- EVENTS
- EXECUTE
- EXISTS
- EXPANSION
- EXTENDED
- FALSE

- FAST
- FIELDS
- FILE
- FIRST
- FIXED
- FLOAT
- FLOAT4
- FLOAT8
- FLUSH
- FORCE
- FULLTEXT
- FUNCTION
- GEOMETRY
- GEOMETRYCOLLECTION
- GET_FORMAT
- GLOBAL
- GRANT
- GRANTS
- HANDLER
- HASH
- HELP
- HIGH_PRIORITY
- HOSTS
- HOUR
- HOUR_MICROSECOND
- HOUR_MINUTE
- HOUR_SECOND
- IDENTIFIED
- IF
- IMPORT
- INDEXES
- INFILE
- INNOBASE
- INNODB
- INSERT_METHOD
- INT
- INT1
- INT2
- INT3
- INT4

- INT8
- INTERVAL
- IO_THREAD
- ISOLATION
- ISSUER
- KEYS
- KILL
- LAST
- LEADING
- LEAVES
- LEVEL
- LINES
- LINESTRING
- LOAD
- LOCAL
- LOCALTIME
- LOCALTIMESTAMP
- LOCK
- LOCKS
- LOGS
- LONG
- LONGBLOB
- LONGTEXT
- LOW_PRIORITY
- MASTER
- MASTER_CONNECT_RETRY
- MASTER_HOST
- MASTER_LOG_FILE
- MASTER_LOG_POS
- MASTER_PASSWORD
- MASTER_PORT
- MASTER_SERVER_ID
- MASTER_SSL
- MASTER_SSL_CA
- MASTER_SSL_CAPATH
- MASTER_SSL_CERT
- MASTER_SSL_CIPHER
- MASTER_SSL_KEY
- MASTER_USER
- MAX_CONNECTIONS_PER_HOUR

- MAX_QUERIES_PER_HOUR
- MAX_ROWS
- MAX_UPDATES_PER_HOUR
- MEDIUM
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- MICROSECOND
- MIDDLEINT
- MINUTE
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MIN_ROWS
- MOD
- MODE
- MODIFY
- MONTH
- MULTILINESTRING
- MULTIPOINT
- MULTIPOLYGON
- NAMES
- NATIONAL
- NDB
- NDBCLUSTER
- NCHAR
- NEW
- NEXT
- NO
- NONE
- NO_WRITE_TO_BINLOG
- NUMERIC
- NVARCHAR
- OLD_PASSWORD
- ONE_SHOT
- OPEN
- OPTIMIZE
- OPTION
- OPTIONALLY
- OUTFILE
- PACK_KEYS

- PARTIAL
- PASSWORD
- POINT
- POLYGON
- PRECISION
- PREPARE
- PREV
- PRIVILEGES
- PROCEDURE
- PROCESS
- PROCESSLIST
- PURGE
- QUERY
- QUICK
- RAID0
- RAID_CHUNKS
- RAID_CHUNKSIZE
- RAID_TYPE
- READ
- REAL
- REGEXP
- RELAY_LOG_FILE
- RELAY_LOG_POS
- RELAY_THREAD
- RELOAD
- RENAME
- REPAIR
- REPEATABLE
- REPLICATION
- REQUIRE
- RESET
- RESTORE
- RETURNS
- REVOKE
- RLIKE
- ROLLUP
- ROWS
- ROW_FORMAT
- RTREE
- SAVEPOINT

- SECOND
- SECOND_MICROSECOND
- SEPARATOR
- SERIAL
- SERIALIZABLE
- SESSION
- SHARE
- SHOW
- SHUTDOWN
- SIGNED
- SIMPLE
- SLAVE
- SMALLINT
- SOME
- SONAME
- SOUNDS
- SPATIAL
- SQL_BIG_RESULT
- SQL_BUFFER_RESULT
- SQL_CACHE
- SQL_CALC_FOUND_ROWS
- SQL_NO_CACHE
- SQL_SMALL_RESULT
- SQL_THREAD
- SSL
- START
- STARTING
- STATUS
- STOP
- STORAGE
- STRAIGHT_JOIN
- STRING
- STRIPED
- SUBJECT
- SUPER
- TABLES
- TABLESPACE
- TERMINATED
- TEXT
- TIME

- TIMESTAMP
- TINYBLOB
- TINYINT
- TINYTEXT
- TRAILING
- TRUE
- TRUNCATE
- TYPE
- TYPES
- UNCOMMITTED
- UNICODE
- UNLOCK
- UNSIGNED
- UNTIL
- USAGE
- USE
- USER
- USER_RESOURCES
- USE_FRM
- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP
- VALUE
- VARBINARY
- VARCHAR
- VARCHARACTER
- VARIABLES
- VARYING
- WARNINGS
- WITH
- WORK
- WRITE
- X509
- YEAR
- YEAR_MONTH
- ZEROFILL

8.4 Kexi PostgreSQL Driver Reserved words

This list contains keywords that are reserved for use by Kexi pqxx Driver:

- ABORT
- ABSOLUTE
- ACCESS
- ACTION
- ADD
- AGGREGATE
- ALTER
- ANALYSE
- ANALYZE
- ANY
- ARRAY
- ASSERTION
- ASSIGNMENT
- AT
- AUTHORIZATION
- BACKWARD
- BIGINT
- BINARY
- BIT
- BOOLEAN
- BOTH
- CACHE
- CALLED
- CAST
- CHAIN
- CHAR
- CHARACTER
- CHARACTERISTICS
- CHECKPOINT
- CLASS
- CLOSE
- CLUSTER
- COALESCE
- COLUMN
- COMMENT
- COMMITTED
- CONSTRAINTS

- CONVERSION
- CONVERT
- COPY
- CREATEDB
- CREATEUSER
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_USER
- CURSOR
- CYCLE
- DAY
- DEALLOCATE
- DEC
- DECIMAL
- DECLARE
- DEFAULTS
- DEFERRABLE
- DEFERRED
- DEFINER
- DELIMITER
- DELIMITERS
- DO
- DOMAIN
- DOUBLE
- EACH
- ENCODING
- ENCRYPTED
- ESCAPE
- EXCEPT
- EXCLUDING
- EXCLUSIVE
- EXECUTE
- EXISTS
- EXTERNAL
- EXTRACT
- FALSE
- FETCH
- FIRST
- FLOAT
- FORCE

- FORWARD
- FREEZE
- FUNCTION
- GLOBAL
- GRANT
- HANDLER
- HOLD
- HOUR
- ILIKE
- IMMEDIATE
- IMMUTABLE
- IMPLICIT
- INCLUDING
- INCREMENT
- INHERITS
- INITIALLY
- INOUT
- INPUT
- INSENSITIVE
- INSTEAD
- INT
- INTERSECT
- INTERVAL
- INVOKER
- ISNULL
- ISOLATION
- LANCOMPILER
- LANGUAGE
- LAST
- LEADING
- LEVEL
- LISTEN
- LOAD
- LOCAL
- LOCALTIME
- LOCALTIMESTAMP
- LOCATION
- LOCK
- MAXVALUE
- MINUTE
- MINVALUE

- MODE
- MONTH
- MOVE
- NAMES
- NATIONAL
- NCHAR
- NEW
- NEXT
- NO
- NOCREATEDB
- NOCREATEUSER
- NONE
- NOTHING
- NOTIFY
- NOTNULL
- NULLIF
- NUMERIC
- OF
- OFF
- OIDS
- OLD
- ONLY
- OPERATOR
- OPTION
- OUT
- OVERLAPS
- OVERLAY
- OWNER
- PARTIAL
- PASSWORD
- PATH
- PENDANT
- PLACING
- POSITION
- PRECISION
- PREPARE
- PRESERVE
- PRIOR
- PRIVILEGES
- PROCEDURAL
- PROCEDURE

- READ
- REAL
- RECHECK
- REINDEX
- RELATIVE
- RENAME
- RESET
- RESTART
- RETURNS
- REVOKE
- ROWS
- RULE
- SCHEMA
- SCROLL
- SECOND
- SECURITY
- SEQUENCE
- SERIALIZABLE
- SESSION
- SESSION_USER
- SETOF
- SHARE
- SHOW
- SIMPLE
- SMALLINT
- SOME
- STABLE
- START
- STATEMENT
- STATISTICS
- STDIN
- STDOUT
- STORAGE
- STRICT
- SUBSTRING
- SYSID
- TEMP
- TEMPLATE
- TIME
- TIMESTAMP
- TOAST

- TRAILING
- TREAT
- TRIGGER
- TRIM
- TRUE
- TRUNCATE
- TRUSTED
- TYPE
- UNENCRYPTED
- UNKNOWN
- UNLISTEN
- UNTIL
- USAGE
- USER
- VACUUM
- VALID
- VALIDATOR
- VARCHAR
- VARYING
- VERBOSE
- VERSION
- VIEW
- VOLATILE
- WITH
- WITHOUT
- WORK
- WRITE
- YEAR
- ZONE

8.5 Kexi Oracle Driver Reserved words

This list contains keywords that are reserved for use by Kexi Oracle Driver:

- ADMIN
- AFTER
- ALLOCATE
- ANALYZE
- ARCHIVE
- ARCHIVELOG
- AUTHORIZATION

- AVG
- BACKUP
- BECOME
- BEFORE
- BEGIN
- BLOCK
- BODY
- CACHE
- CANCEL
- CASCADE
- CHANGE
- CHARACTER
- CHECKPOINT
- CLOSE
- COBOL
- COMMIT
- COMPILE
- CONSTRAINT
- CONSTRAINTS
- CONTENTS
- CONTINUE
- CONTROLFILE
- COUNT
- CURSOR
- CYCLE
- DATABASE
- DATAFILE
- DATE
- DBA
- DEC
- DECLARE
- DISABLE
- DISMOUNT
- DOUBLE
- DUMP
- EACH
- ENABLE
- END
- ESCAPE
- EVENTS
- EXCEPT

- EXCEPTIONS
- EXEC
- EXECUTE
- EXPLAIN
- EXTENT
- EXTERNALLY
- FETCH
- FLUSH
- FORCE
- FOREIGN
- FORTRAN
- FOUND
- FREELIST
- FREELISTS
- FUNCTION
- GO
- GOTO
- GROUPS
- INCLUDING
- INDICATOR
- INITRANS
- INSTANCE
- INT
- KEY
- LANGUAGE
- LAYER
- LINK
- LISTS
- LOGFILE
- MANAGE
- MANUAL
- MAX
- MAXDATAFILES
- MAXINSTANCES
- MAXLOGFILES
- MAXLOGHISTORY
- MAXLOGMEMBERS
- MAXTRANS
- MAXVALUE
- MIN
- MINEXTENTS

- MINVALUE
- MODULE
- MOUNT
- NEW
- NEXT
- NOARCHIVELOG
- NOCACHE
- NOCYCLE
- NOMAXVALUE
- NOMINVALUE
- NONE
- NOORDER
- NORESETLOGS
- NORMAL
- NOSORT
- NUMERIC
- OFF
- OLD
- ONLY
- OPEN
- OPTIMAL
- OWN
- PACKAGE
- PARALLEL
- PCTINCREASE
- PCTUSED
- PLAN
- PLI
- PRECISION
- PRIMARY
- PRIVATE
- PROCEDURE
- PROFILE
- QUOTA
- READ
- REAL
- RECOVER
- REFERENCES
- REFERENCING
- RESETLOGS
- RESTRICTED

- REUSE
- ROLE
- ROLES
- ROLLBACK
- SAVEPOINT
- SCHEMA
- SCN
- SECTION
- SEGMENT
- SEQUENCE
- SHARED
- SNAPSHOT
- SOME
- SORT
- SQL
- SQLCODE
- SQLERROR
- SQLSTATE
- STATEMENT_ID
- STATISTICS
- STOP
- STORAGE
- SUM
- SWITCH
- SYSTEM
- TABLES
- TABLESPACE
- TEMPORARY
- THREAD
- TIME
- TRACING
- TRANSACTION
- TRIGGERS
- TRUNCATE
- UNDER
- UNLIMITED
- UNTIL
- USE
- USING
- WHEN
- WORK
- WRITE

8.6 Kexi Sybase Driver Reserved words

This list contains keywords that are reserved for use by Kexi Sybase Driver:

- ACTION
- ADD
- AGAINST
- AGGREGATE
- ALTER
- ANALYZE
- ANY
- ASCII
- AUTOINCREMENT
- AVG
- AVG_ROW_LENGTH
- BACKUP
- BDB
- BERKELEYDB
- BIGINT
- BINARY
- BINLOG
- BIT
- BLOB
- BOOL
- BOOLEAN
- BOTH
- BTREE
- BYTE
- CACHE
- CHANGE
- CHANGED
- CHAR
- CHARACTER
- CHARSET
- CHECKSUM
- CIPHER
- CLIENT
- CLOSE
- COLLATION
- COLUMN
- COLUMNS

- COMMENT
- COMMITTED
- COMPRESSED
- CONCURRENT
- CONVERT
- CUBE
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_USER
- DATA
- DATABASES
- DATE
- DATETIME
- DAY
- DAY_HOUR
- DAY_MICROSECOND
- DAY_MINUTE
- DAY_SECOND
- DEALLOCATE
- DEC
- DECIMAL
- DELAYED
- DELAY_KEY_WRITE
- DESCRIBE
- DES_KEY_FILE
- DIRECTORY
- DISABLE
- DISCARD
- DISTINCTROW
- DIV
- DO
- DOUBLE
- DUAL
- DUMPFILE
- DUPLICATE
- DYNAMIC
- ENABLE
- ENCLOSED
- ENGINE

- ENGINES
- ENUM
- ERRORS
- ESCAPE
- ESCAPED
- EVENTS
- EXECUTE
- EXISTS
- EXPANSION
- EXTENDED
- FALSE
- FAST
- FIELDS
- FILE
- FIRST
- FIXED
- FLOAT
- FLOAT4
- FLOAT8
- FLUSH
- FORCE
- FULLTEXT
- FUNCTION
- GEOMETRY
- GEOMETRYCOLLECTION
- GET_FORMAT
- GLOBAL
- GRANT
- GRANTS
- HANDLER
- HASH
- HELP
- HIGH_PRIORITY
- HOSTS
- HOUR
- HOUR_MICROSECOND
- HOUR_MINUTE
- HOUR_SECOND
- IDENTIFIED
- IF

- IMPORT
- INDEXES
- INFILE
- INNOBASE
- INNODB
- INSERT_METHOD
- INT
- INT1
- INT2
- INT3
- INT4
- INT8
- INTERVAL
- IO_THREAD
- ISOLATION
- ISSUER
- KEYS
- KILL
- LAST
- LEADING
- LEAVES
- LEVEL
- LINES
- LINESTRING
- LOAD
- LOCAL
- LOCALTIME
- LOCALTIMESTAMP
- LOCK
- LOCKS
- LOGS
- LONG
- LONGBLOB
- LONGTEXT
- LOW_PRIORITY
- MASTER
- MASTER_CONNECT_RETRY
- MASTER_HOST
- MASTER_LOG_FILE
- MASTER_LOG_POS

- MASTER_PASSWORD
- MASTER_PORT
- MASTER_SERVER_ID
- MASTER_SSL
- MASTER_SSL_CA
- MASTER_SSL_CAPATH
- MASTER_SSL_CERT
- MASTER_SSL_CIPHER
- MASTER_SSL_KEY
- MASTER_USER
- MAX_CONNECTIONS_PER_HOUR
- MAX_QUERIES_PER_HOUR
- MAX_ROWS
- MAX_UPDATES_PER_HOUR
- MEDIUM
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- MICROSECOND
- MIDDLEINT
- MINUTE
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MIN_ROWS
- MOD
- MODE
- MODIFY
- MONTH
- MULTILINESTRING
- MULTIPOINT
- MULTIPOLYGON
- NAMES
- NATIONAL
- NDB
- NDBCLUSTER
- NCHAR
- NEW
- NEXT
- NO
- NONE

- NO_WRITE_TO_BINLOG
- NUMERIC
- NVARCHAR
- OLD_PASSWORD
- ONE_SHOT
- OPEN
- OPTIMIZE
- OPTION
- OPTIONALLY
- OUTFILE
- PACK_KEYS
- PARTIAL
- PASSWORD
- POINT
- POLYGON
- PRECISION
- PREPARE
- PREV
- PRIVILEGES
- PROCEDURE
- PROCESS
- PROCESSLIST
- PURGE
- QUERY
- QUICK
- RAID0
- RAID_CHUNKS
- RAID_CHUNKSIZE
- RAID_TYPE
- READ
- REAL
- REGEXP
- RELAY_LOG_FILE
- RELAY_LOG_POS
- RELAY_THREAD
- RELOAD
- RENAME
- REPAIR
- REPEATABLE
- REPLICATION

- REQUIRE
- RESET
- RESTORE
- RETURNS
- REVOKE
- RLIKE
- ROLLUP
- ROWS
- ROW_FORMAT
- RTREE
- SAVEPOINT
- SECOND
- SECOND_MICROSECOND
- SEPARATOR
- SERIAL
- SERIALIZABLE
- SESSION
- SHARE
- SHOW
- SHUTDOWN
- SIGNED
- SIMPLE
- SLAVE
- SMALLINT
- SOME
- SONAME
- SOUNDS
- SPATIAL
- SQL_BIG_RESULT
- SQL_BUFFER_RESULT
- SQL_CACHE
- SQL_CALC_FOUND_ROWS
- SQL_NO_CACHE
- SQL_SMALL_RESULT
- SQL_THREAD
- SSL
- START
- STARTING
- STATUS
- STOP

- STORAGE
- STRAIGHT_JOIN
- STRING
- STRIPED
- SUBJECT
- SUPER
- TABLES
- TABLESPACE
- TERMINATED
- TEXT
- TIME
- TIMESTAMP
- TINYBLOB
- TINYINT
- TINYTEXT
- TRAILING
- TRUE
- TRUNCATE
- TYPE
- TYPES
- UNCOMMITTED
- UNICODE
- UNLOCK
- UNSIGNED
- UNTIL
- USAGE
- USE
- USER
- USER_RESOURCES
- USE_FRM
- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP
- VALUE
- VARBINARY
- VARCHAR
- VARCHARACTER
- VARIABLES
- VARYING
- WARNINGS

- WITH
- WORK
- WRITE
- X509
- YEAR
- YEAR_MONTH
- ZEROFILL

8.7 Kexi xBase Driver Reserved words

This list contains keywords that are reserved for use by Kexi xBase Driver:

- ABORT
- ATTACH
- CLUSTER
- CONFLICT
- DEFERRED
- DEFERRABLE
- DETACH
- EACH
- EXCEPT
- FAIL
- GLOB
- IMMEDIATE
- INITIALLY
- INSTEAD
- INTERSECT
- ISNULL
- NOTNULL
- OF
- PRAGMA
- RAISE
- STATEMENT
- TEMP
- TRIGGER
- VACUUM
- VIEW

> NOTE
> Reserved words are kept separate for each driver so that they can also be used as reference.

Chapter 9

Appendix D. Supported File Formats

9.1 Comma-separated values format (CSV)

Kexi is capable of importing and exporting data from/to a variety of Comma-separated values formats (CSV files). Most spreadsheet and database applications can import and export in this format, making this an appropriate format to migrate textual data between different applications.

Kexi also supports a number of options that can be configured before performing import:

- data type for each column,
- other field delimiters such as tabs,
- different text quotes,
- given number of records can be skipped if needed,
- duplicated delimiters can be skipped,
- values from the first row can be used to set column names,
- text encoding (UTF-8 is the default),
- date format (defined by the operating system is the default),
- stripping leading and trailing blanks off of the text values.

On importing Kexi shows preview of imported data. Most suitable set of options is auto-detected by Kexi based on provided CSV file.

9.2 Microsoft Access (MDB) file format

9.2.1 Overview

Support for importing **Microsoft Access** databases (2003 or older) is built into Kexi. **Access** databases can be *imported* into a Kexi database only. It cannot be used to edit the database, nor to export to an **Access** database. However, it should also be able to import databases created by other applications that use the JET database engine. Importing `.accdb` databases introduced in MS **Access** 2007 are not currently supported.

To import a database:

1. In Kexi menu select **Import, Export or Send..** command.
2. Click **Import Database** button.
3. Use the import wizard by selecting desired .mdb file.

9.2.2 Capabilities

The import function has successfully been used to import *Northwind* database well known to MS **Access** users. Many other **Access** template databases can be also imported.

9.2.3 Supported features

Importing the following field types has been tested, and generally works well:

- Text fields
- Memo fields
- Date fields
- Numeric values

9.2.4 Unsupported features

Currently, only tables are imported. During import Kexi skips the following objects:

- Queries
- Forms
- Scripts
- Reports

The following are untested:

- Importing binary objects

Chapter 10

Credits and License

Kexi Developers:

- Jaroslaw Staniek (staniek kde.org)
- OpenOffice Polska, LLC (info openoffice.com.pl)
- Lucijan Busch (lucijan kde.org)
- Cedric Pasteur (cedric.pasteur free.fr)
- Adam Pigg (adam piggz.fsnet.co.uk)
- Martin Ellis (martin.ellis kdemail.net)
- Sebastian Sauer (mail dipe.org)
- Christian Nitschkowski (segfault_ii web.de)
- Peter Simonsson (psn linux.se)
- Joseph Wenninger (jowenn kde.org)
- Seth Kurzenberg (seth cql.com)
- Laurent Montel (montel kde.org)
- Till Busch (till bux.at)

Documentation by Martin A. Ellis (martin.ellis kdemail.net), Jaroslaw Staniek (staniek kde.org) with contributions from Anne-Marie Mahfouf, Raphael Langerhorst, Michal Kubicki and Aron Stansvik.