OpenOffice.org Base: Comparison To MS Access, Scripting, Flexibility Of Backend

Andreas Beck
Vienna University of Economics and Business Administration
Reg.No. 9410124

Seminar Paper
Department of Business Informatics
Prof. Dr. Rony G. Flatscher

Abstract

Open Office is an open source alternative to commercial office packages and offers all components other packages have as well. Open Office is able to open and save Microsoft documents as well as it is able to save files in pdf format. A lot of people hesitate to switch from Microsoft Office to Open Office as on one hand Microsoft Office is very well known and used all over. On the other hand many people think that Open Office lacks functionality Microsoft Office offers. Basically, Open Office is an office package with all important funcionality by now, although it might lack some functionality concerning deep level functions most of the ordinary user will not miss.
This paper takes a look at the database application within Open Office. It compares Microsoft Access and Open Office Base on one hand. On the other hand it looks at the possibilities of automatisation and scripting and gives some examples.

Keywords: OpenOffice.org, OpenOffice.org Base, MS Access, Microsoft Access, UNO, Object Rexx

Table Of Contents

Table of Figures

Used Applications

Java(TM) SE Runtime Environment (build 1.6.0_03-b05): The Java Standard Edition
http://java.sun.com

bsf4rexx: The bean scripting framework for rexx
http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/

Microsoft Office Access 2003: Part of the Microsoft office package
http://office.microsoft.com

MySql 5.0.45: MySql database engine
http://www.mysql.com/

ooRexx 3.2.0: ooRexx programming language
http://www.oorexx.org/

OpenOffice.org 2.3.0: Open Office office package
http://www.openoffice.org/

Vim
Editor with syntax highlighting for many scripting and programming languages like ooRexx, Java and HTML export
http://www.vim.org

XAMPP: Development environment including MySql
http://www.apachefriends.org/de/xampp.html

1 Comparison OOo Base – Microsoft Access

In this chapter the focus is on the common aspects as well as on the differences between the open source database software OOo base and the commercial database software Microsoft Access. Open Office in general is often taken as a free alternative to Microsoft Offce, OOo base and MS Access are one part of each packages providing database functionality.

1.1 Comparison Front End OOo Base - Microsoft Office Access

The front end is understood as the interface for the user for any change within the application. Front ends usually provide a gui.

1.1.1 Front End OOo Base

1.1.1.1 Creating A New Database In OOo Base

When starting the OpenOffice.org database application OOo Base the main choices are:

- Create a new database
  Creates a new database in OOo Base format.
- Open an existing database file
  Opens an existing OOo database file.
- Connect to an existing database
  Connects to an existing database of various types.

After creating or opening a database file the possibility of registering the database within the application is offered. To register means to store where the data is located and how it is organized. By doing that, the data records can be accessed from within your OOo applications like text documents and spreadsheets.

Fig. 1: OOo database wizard

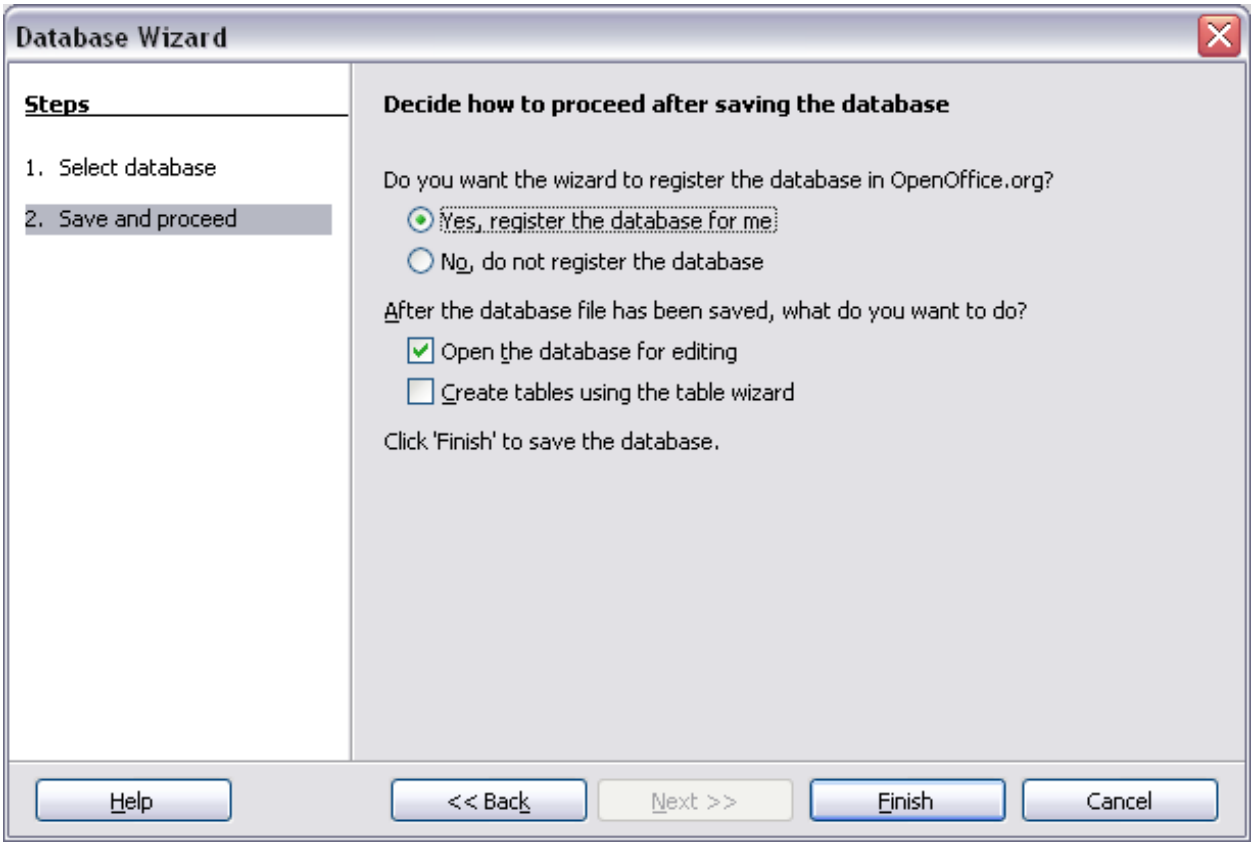

Fig. 2: Register database

1.1.1.2 GUI Elements In OOo Base

The main screen is divided into three areas:

- The database area on the left
- The tasks area at the top
- The elements area at the bottom

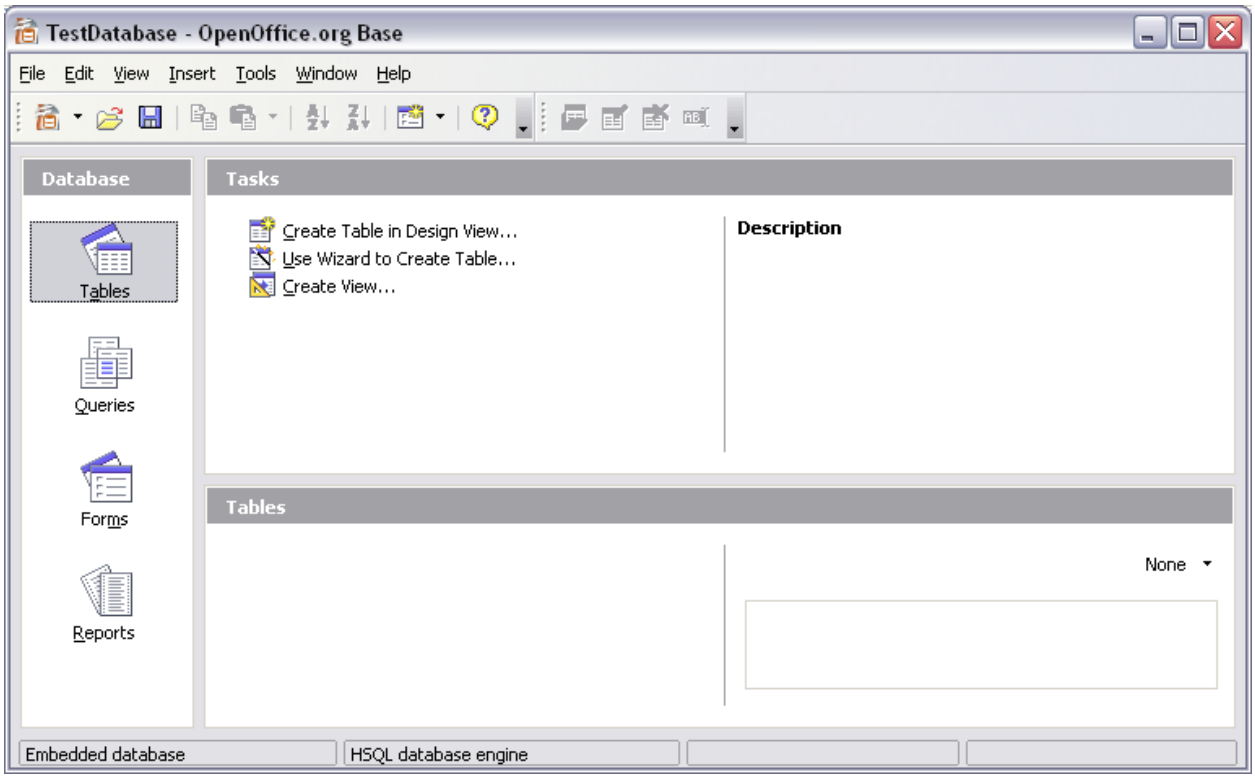

Fig. 3: GUI elements OOo Base

The database area holds the four main elements of the database: tables, queries, forms and reports. Dependent on the database element you can select different tasks in the tasks area. The elements area at the bottom lists the different saved instances of the different database elements.

Tables

Tables form the physical structure to hold the data. Possible tasks are creating a table either in design view or by using a wizard and creating a view. Wizards offer the advantage of not having to

create an own data model but to use already existing ones for frequent types of usage like CD/DVD collections, a library or an address database. Many different types of databases are available here. By creating a table in the design view it is necessary to have a good database knowledge in order to set up a relational database model and to avoid mistakes to keep your database consistent and free of errors and lost data.

Tables can hold different kind of data. Common types are integer, boolean, text and blobs. These data types can be implemented differently in the single database engines. Therefore it's very difficult to write multi database SQL code as these data types differ so much.

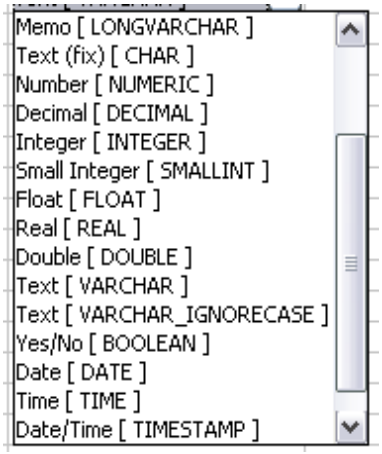

Fig. 4: Datatypes OOo Base

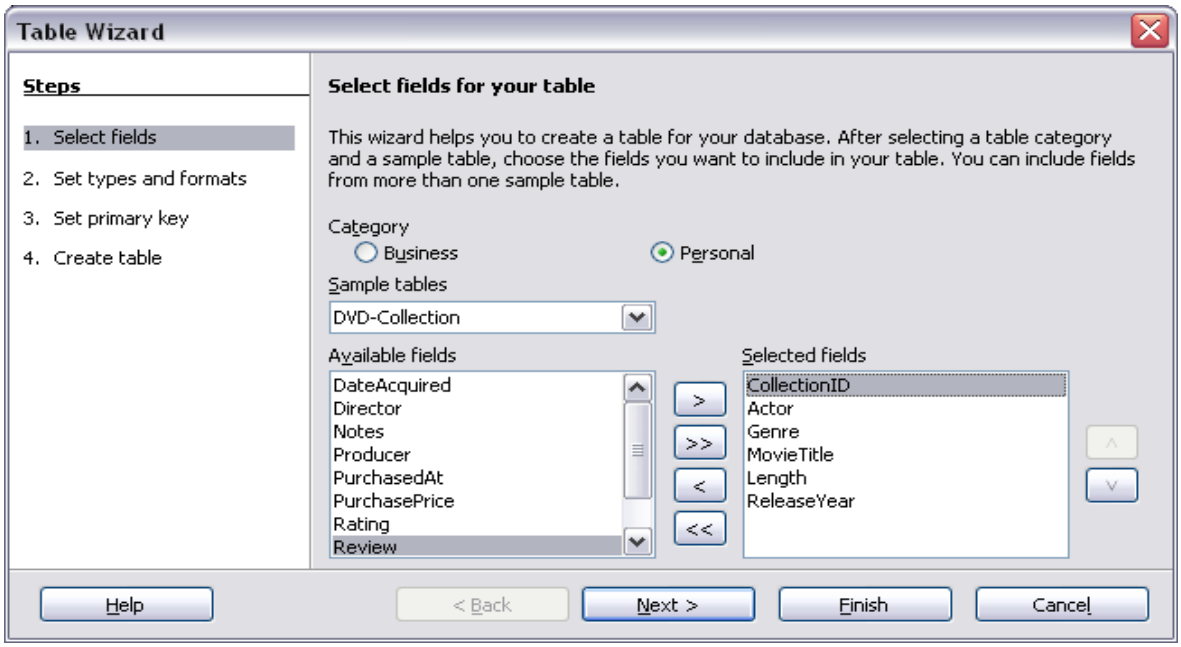

Fig. 5: Table wizard OOo Base – select fields

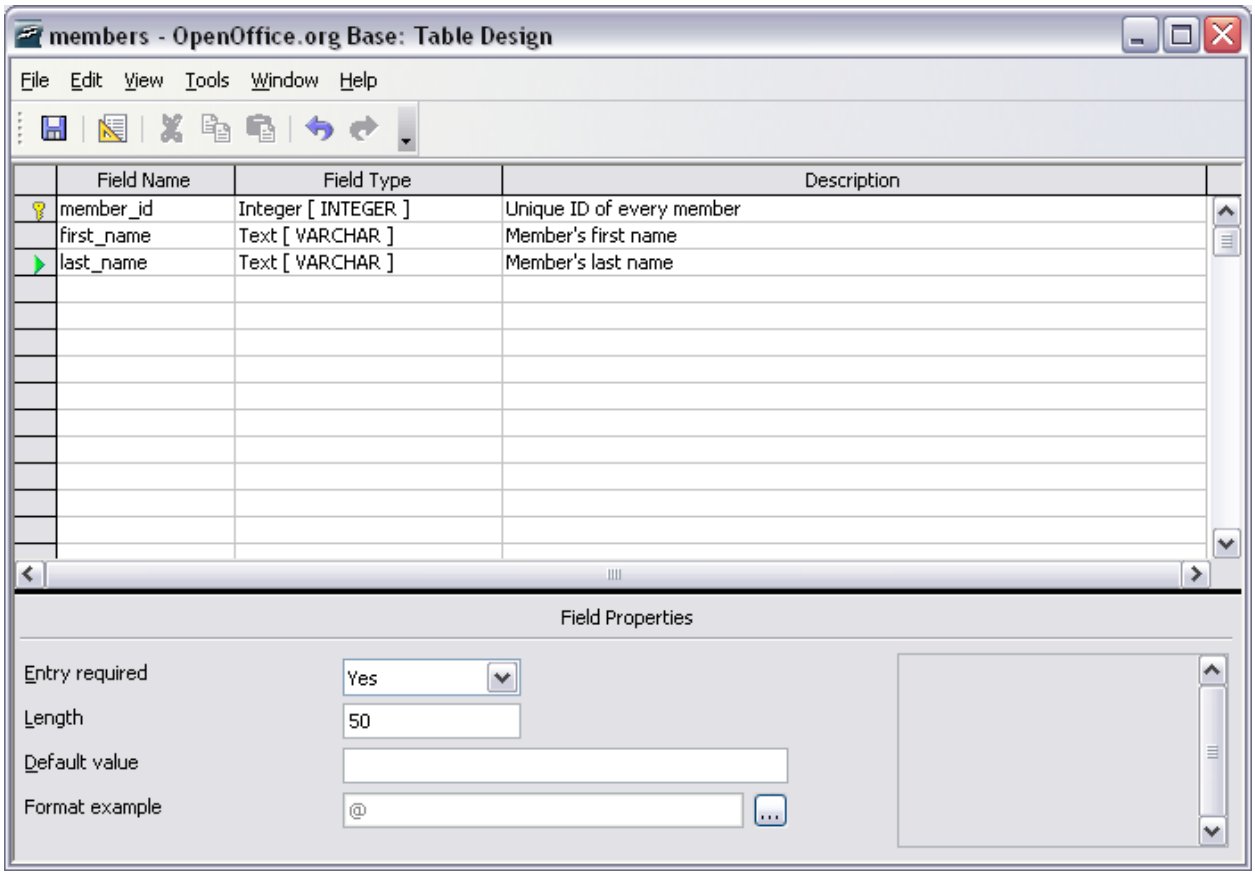

Fig. 6: Table design OOo Base

Different from tables, views don't have physically stored data but they form a table in memory by using SQL as a database language. Views can basically be used like tables for querying for data but cannot store data themselves. But you need to be careful. After saving a view it cannot be modified any more, just used within other queries or views. When you try to edit the view, all that is shown is a window similar to the table edit window except that you cannot modify the single fields any more.

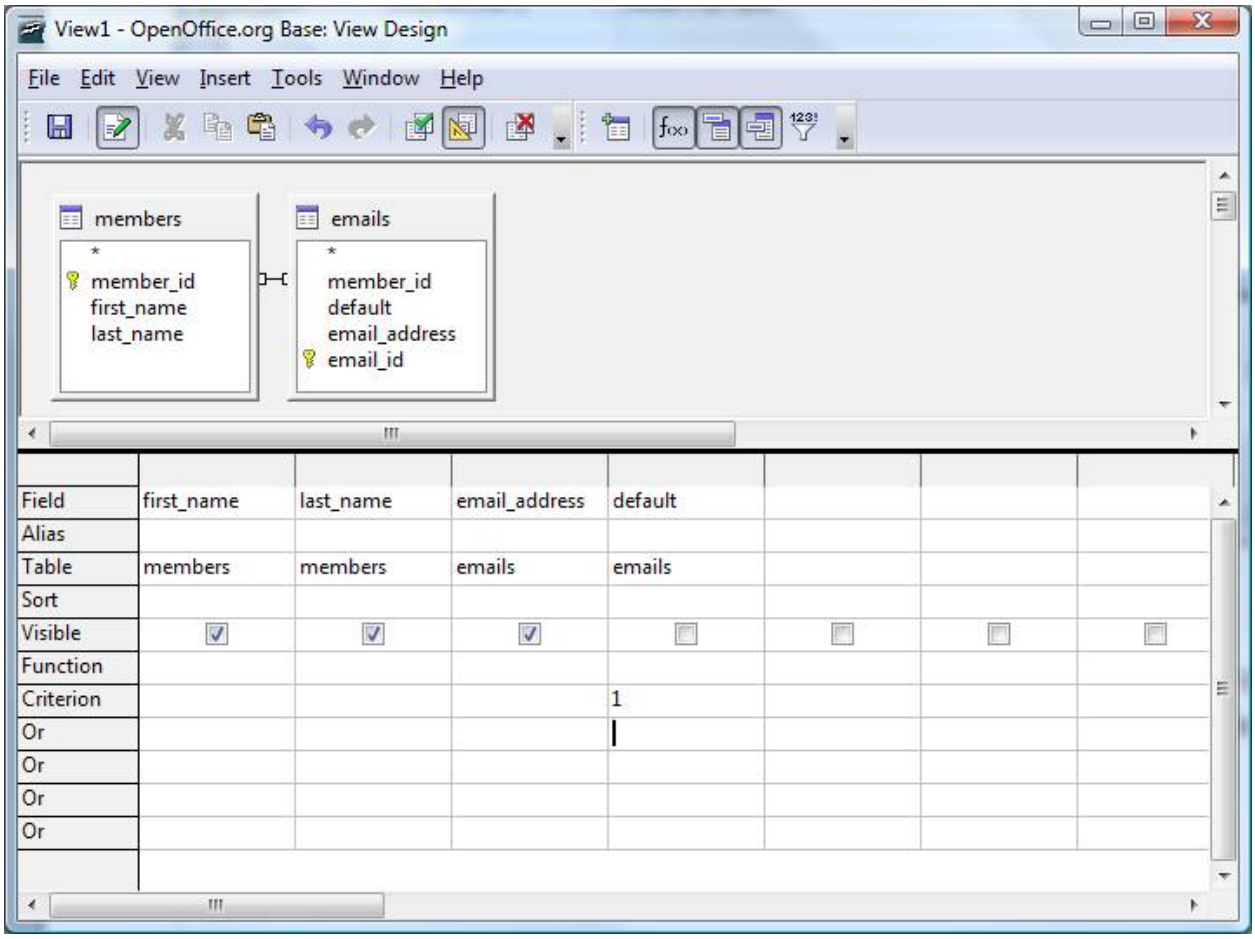

Fig. 7: View design OOo Base

Queries

Basically, there are 3 ways for creating queries. The first way is creating a query in the so called Design View. There you get a window where you can include different tables, views and other queries and where you can set criteria and functions, sortings and aliases. The capabilities of this screen however are limited. It's only possible to send SQL SELECT statements to the database using this screen, there are of course other possibilities to do so. Furthermore, only if you use a single table within this query designer you're able to add data when you open the query. If you use more than one table or even only one view, no additional data can be entered.

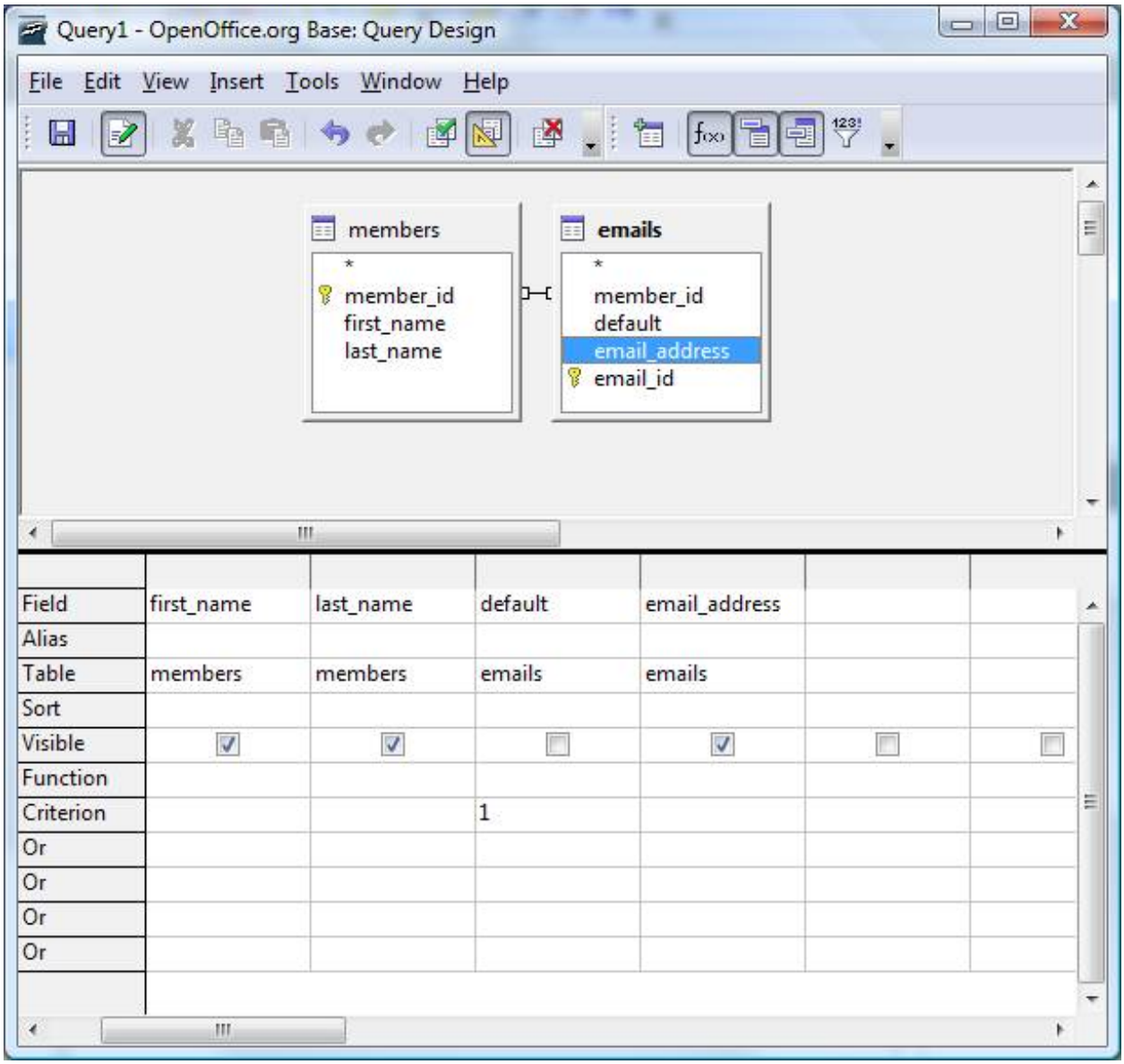

Fig. 8: Query design OOo Base

The second possibility is to use the wizard to create a query. In the query wizard you can select different tables, views and queries similar to the design view. In this screen the user can go through the different steps and select fields, add a sorting and search conditions, he can use functions for getting a sum, maximum/minimum etc. and grouping and aliases can be applied. Functionality is limited like in the design view.

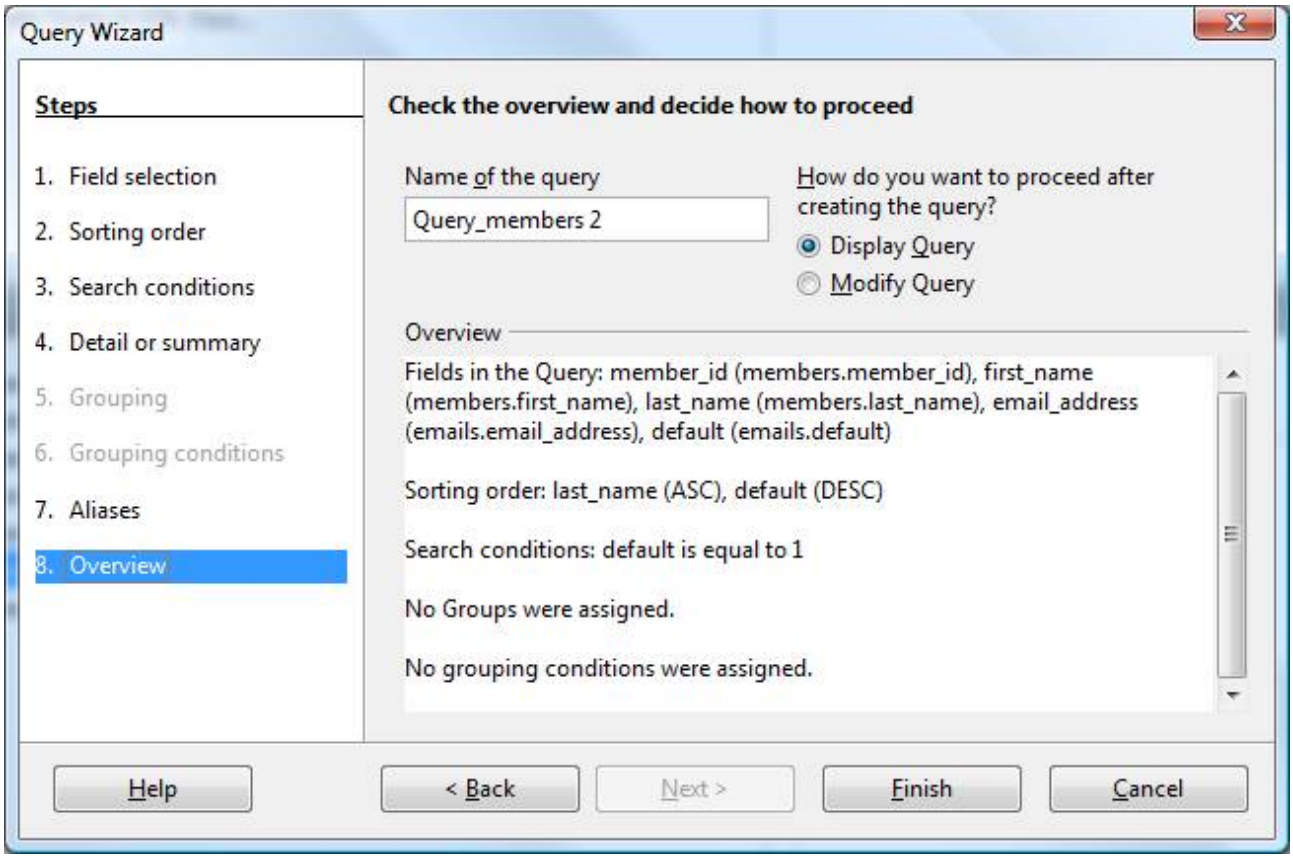

Fig. 9: Query wizard OOo Base

The third possibility is to create a query in the SQL view. This SQL view is intended to be the place for more sophisticated select statements by OpenOffice.org. It is not possible to send other commands than SELECT to the database. The right place for the larger use of different kinds of SQL statements are macros.

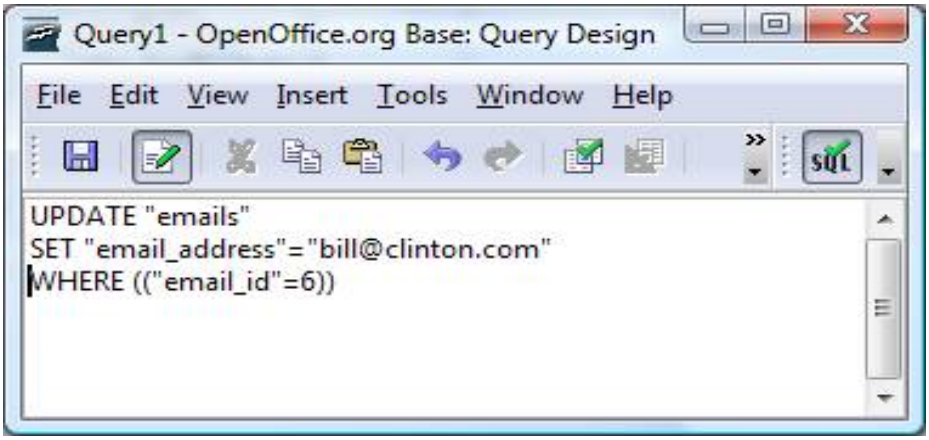

Fig. 10: SQL query OOo Base

Forms

Basically, forms can be used for entering and amending data or for display purposes only. The basis for forms are tables (including views) and queries. Forms can only offer the funcionality the underlying data sources have, i.e. you can only have a form for amending data if you use the tables or queries where data is amendable, queries with multiple data sources can therefore not be the basis for a form for amending or entering data. Forms are no OOo Base objects but writer documents which have the ability to interact with the base backend.

Like in all other database objects forms can be created by using a wizard or by creating them without any system help in design view. The wizard for forms is sufficient if you want to create ordinary forms with a maximum of one subform. The wizard guides you through the setup steps very efficiently and clearly. The user can select the fields out of the different data sources, he can set up a subform (based on existing relations or on manually set up relations), the user can then chose the look of the form and subform by just clicking on the arrangement of the data within the form. After that, the user can choose how the data shall be treated, i.e. whether the form is used for entering new data only or the form is restricted in funcionality in the sense that modification/deletion/addition of new data can be limited. At the end of the configuration steps the styles and name of the form can be set and the form is generated based on the chosen values.

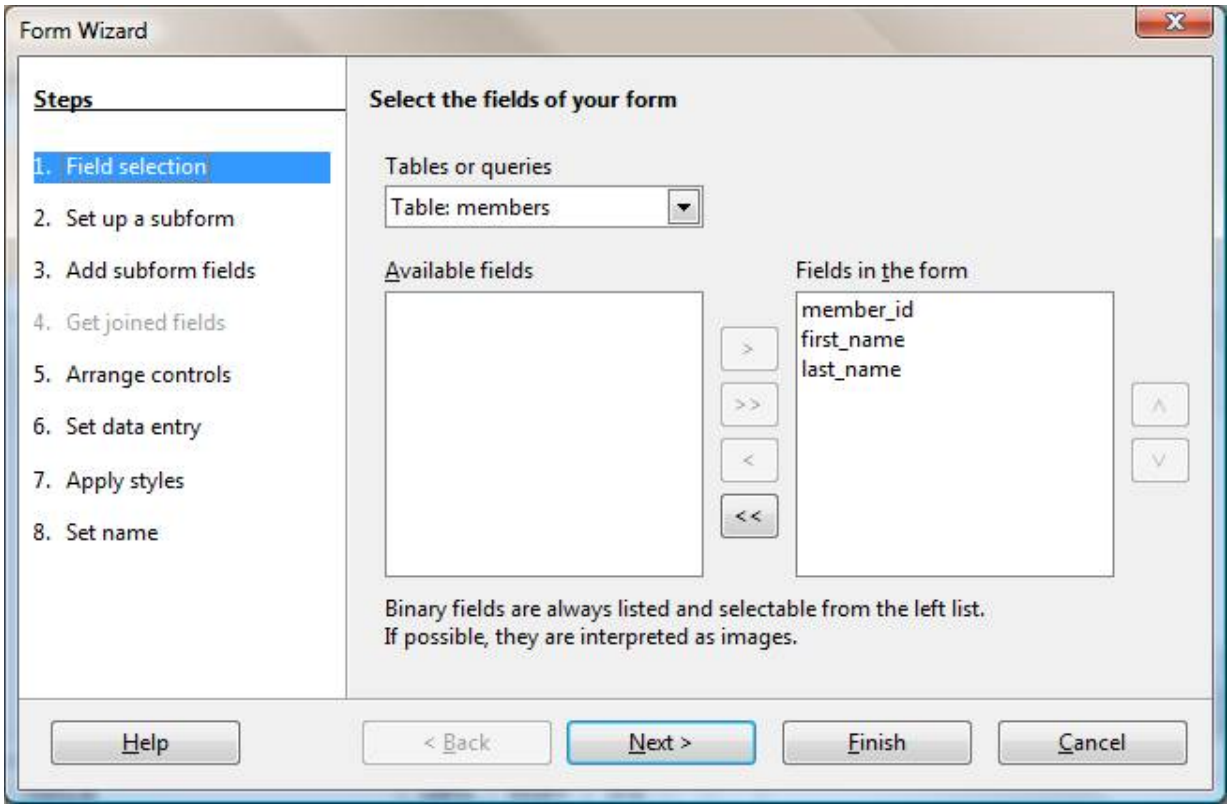

Fig. 11: Form wizard OOo Base – field selection

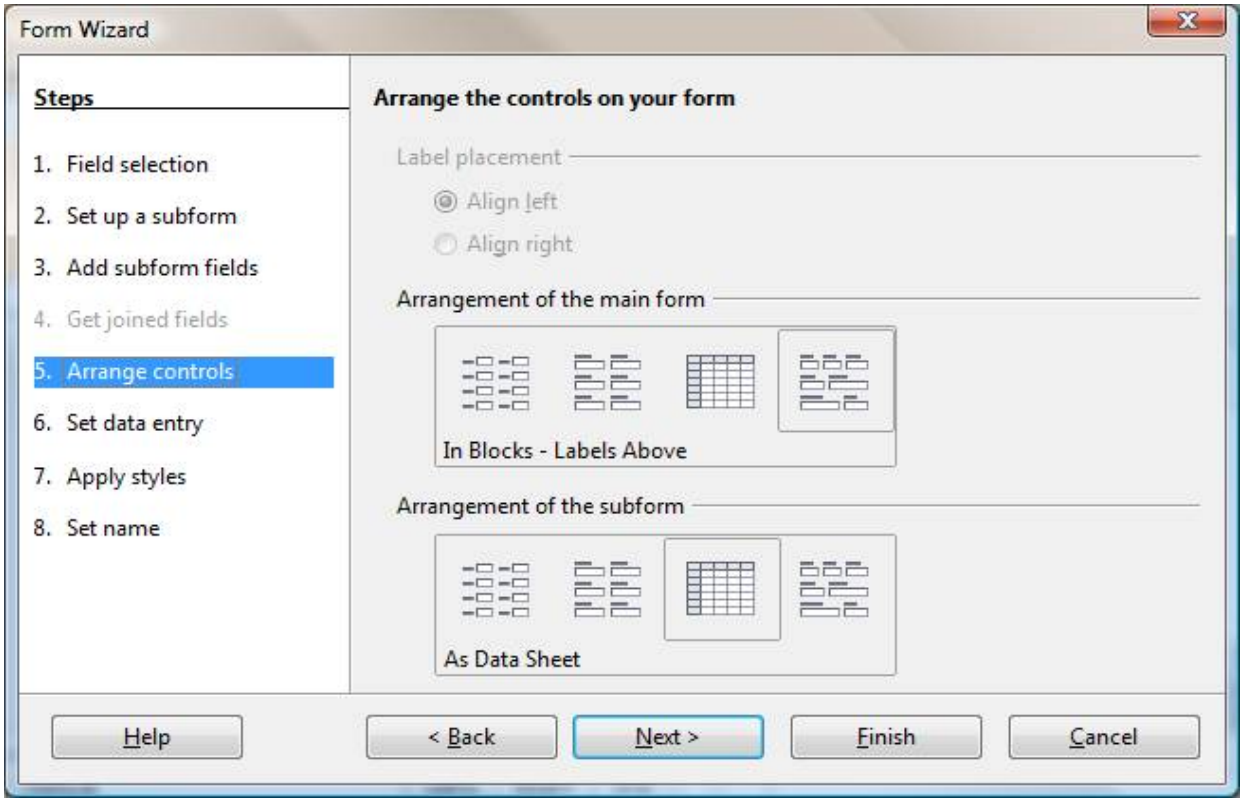

Fig. 12: Form wizard OOo Base – arrange controls

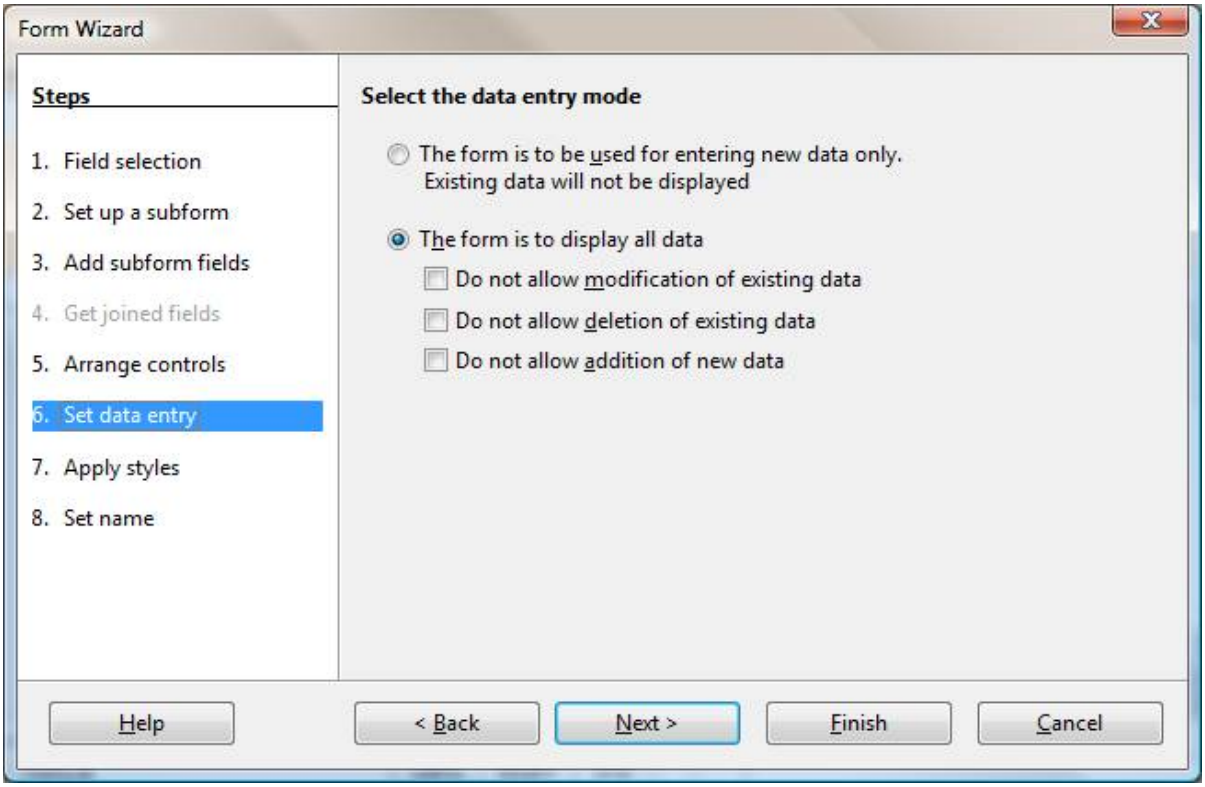

Fig. 13: Form wizard OOo Base – set data entry

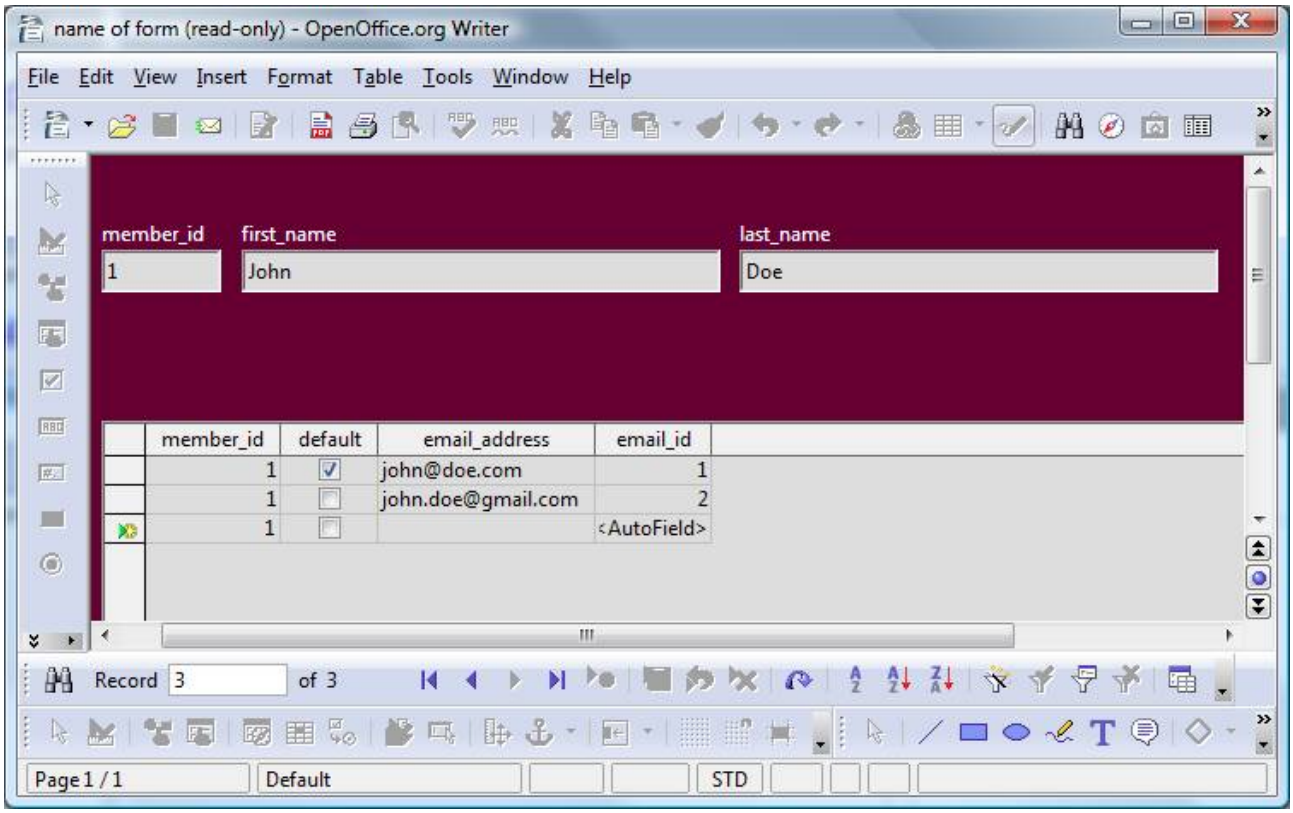

Fig. 14: Form OOo Base

When using the design view, a blank OOo Writer document is opened. The form navigator is then used to set up forms and subforms within this writer document. This funcionality can be used in writer documents independently and need not be done within a base document (use toolbar „form design“).

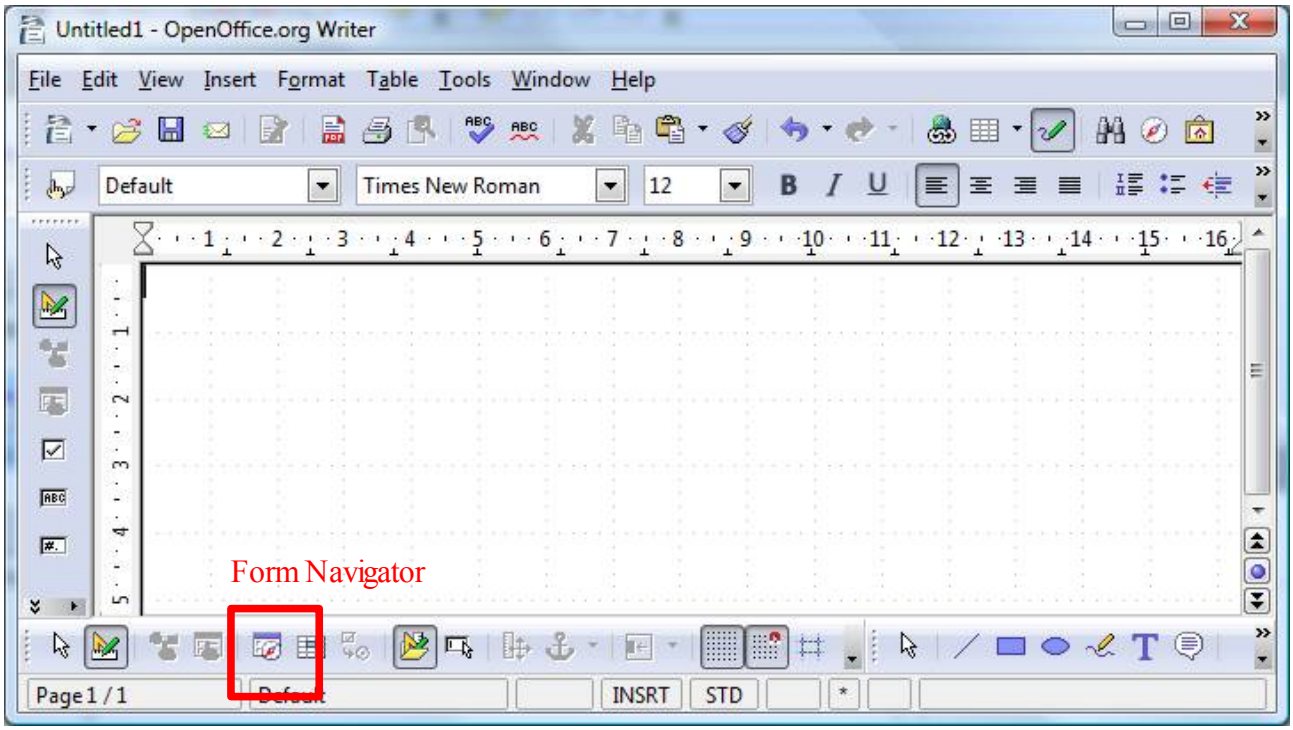

Fig. 15: Form navigator in OOo Writer

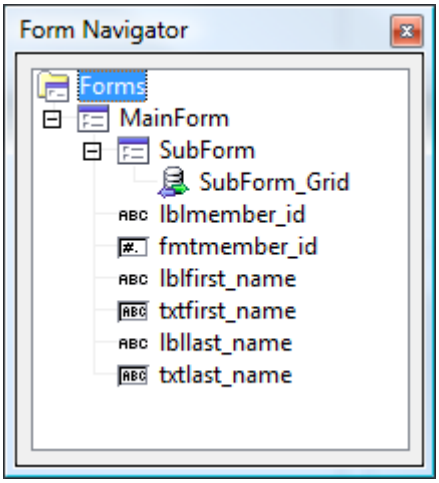

Fig. 16: Form navigator control

Reports

Reports can only be used for displaying data in a standardized, quick and easy way. They can be created using a wizard only with predefined layouts. Different from forms it is not possible to create subreports within a main report. Furthermore it is possible to select one datasource only, so for many cases it may be useful to define a view or query for the requested data. Similar to creating a form, after going through the steps of the wizard a new writer document based on the values of the wizard is created and the data is displayed. At the end the user can choose between a static and a dynamic report. The static report will always show the data as it was when creating the report, the dynamic report will always update the shown data when opening the report. Reports are not very flexible but are very suitable for quick results.

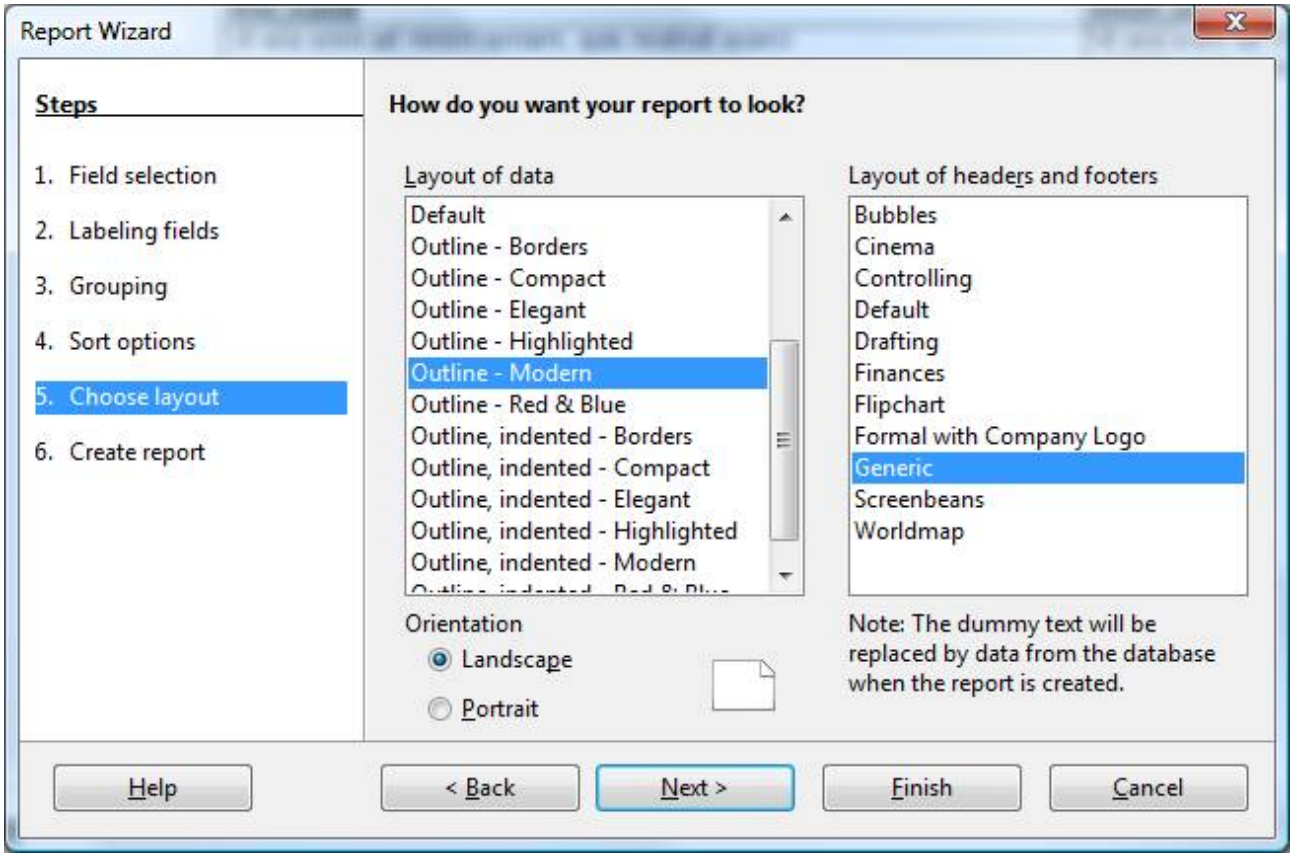

Fig. 17: Report wizard OOo Base – choose layout

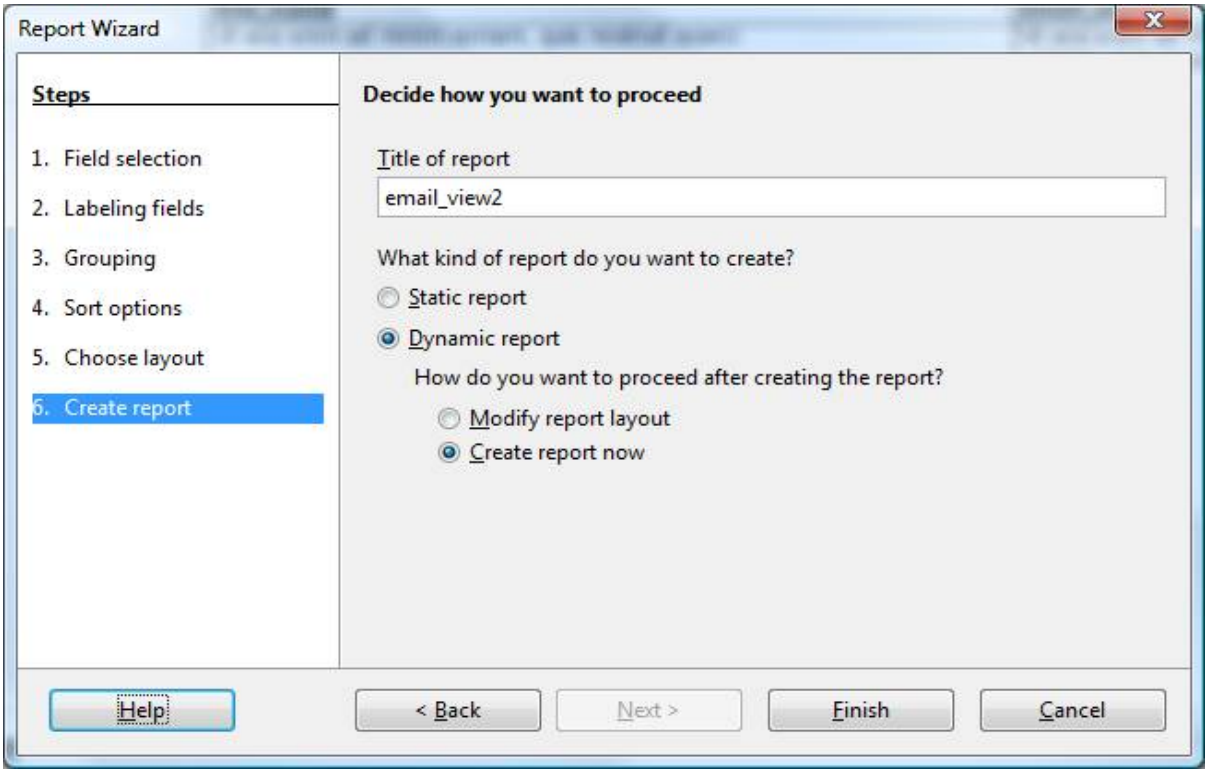

Fig. 18: Report wizard OOo Base – create report

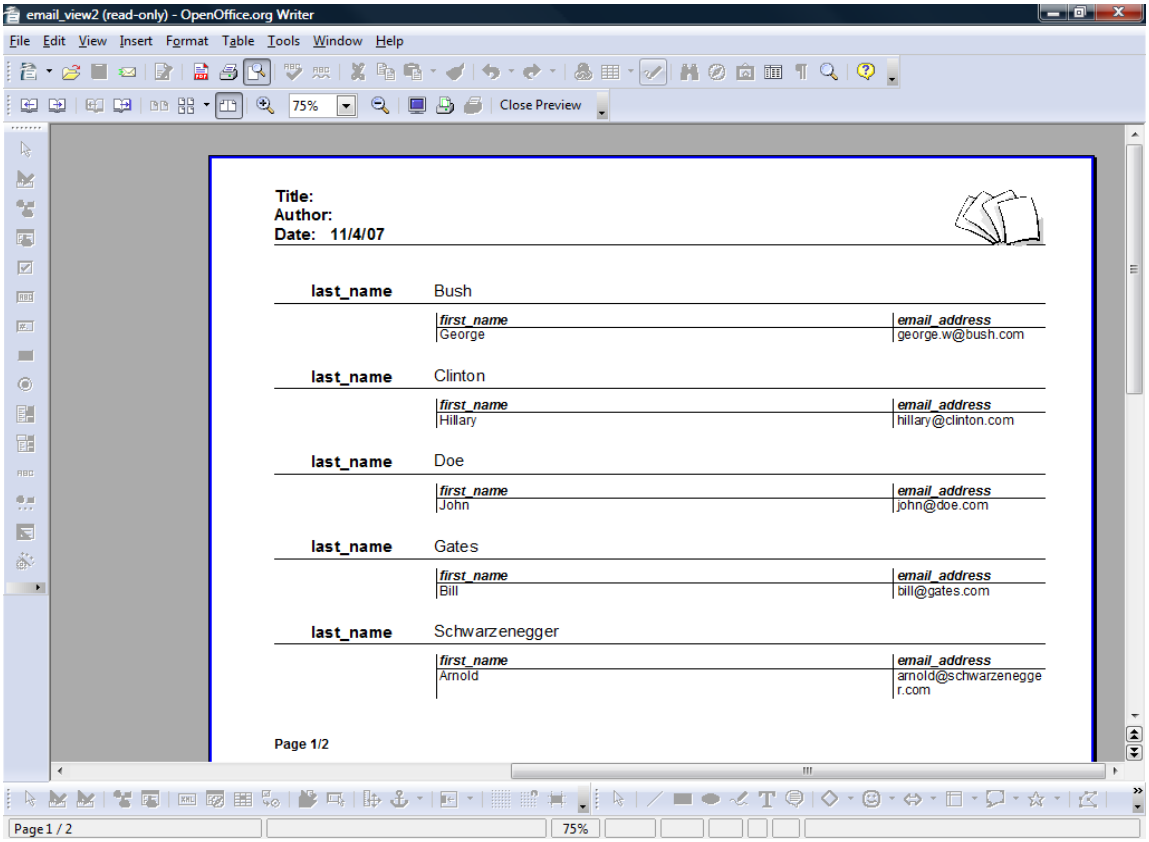

Fig. 19: Report OOo Base

1.1.2 Front End Microsoft Office Access

1.1.2.1 Creating A New Database In MS Access

The choices when starting MS Access are:

- Create a new database

  Creates a new database in MS Access format
- Open an existing database

  Opens an existing database of various types

1.1.2.2 GUI Elements In MS Access

The main screen is divided into two areas:

- The objects area on the left
- The elements area on the right

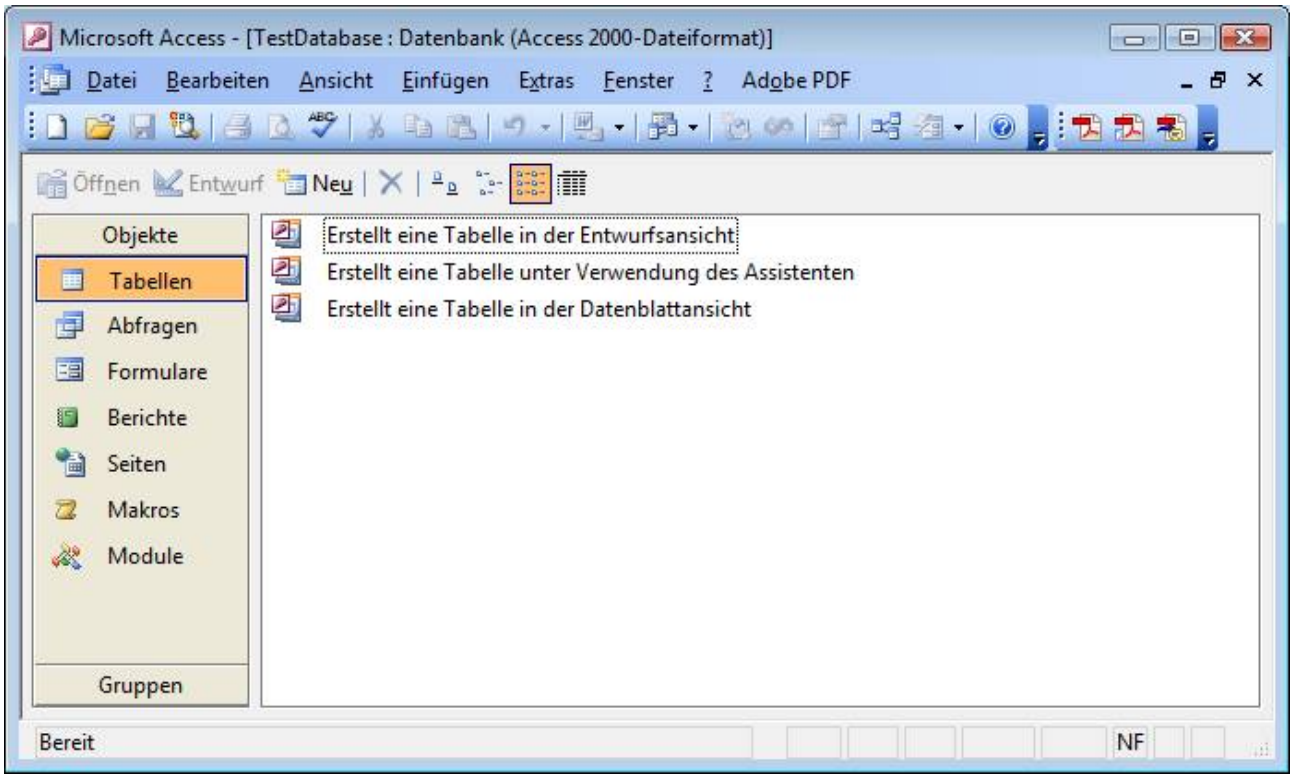

Fig. 20: GUI elements in MS Access

The objects area holds the seven main elements of the database: tables, queries, forms, reports, pages, macros and modules. The elements area on the right holds the different possible tasks which can be performed as well as the existing object instances.

Tables

Tables in MS Office are like tables in OOo Base the place to store the data. The wizard works similar to the wizard in OOo Base. You can select from a list of example tables to build up a database which fulfills common needs. The design view is very similar, too. The differences can be found in the data types and in the preferences. A function that can be found here is to create a table in the table view. An empty table is created where the user can fill in data. The front end itself determines the data type and the table is created, the column names can be renamed.

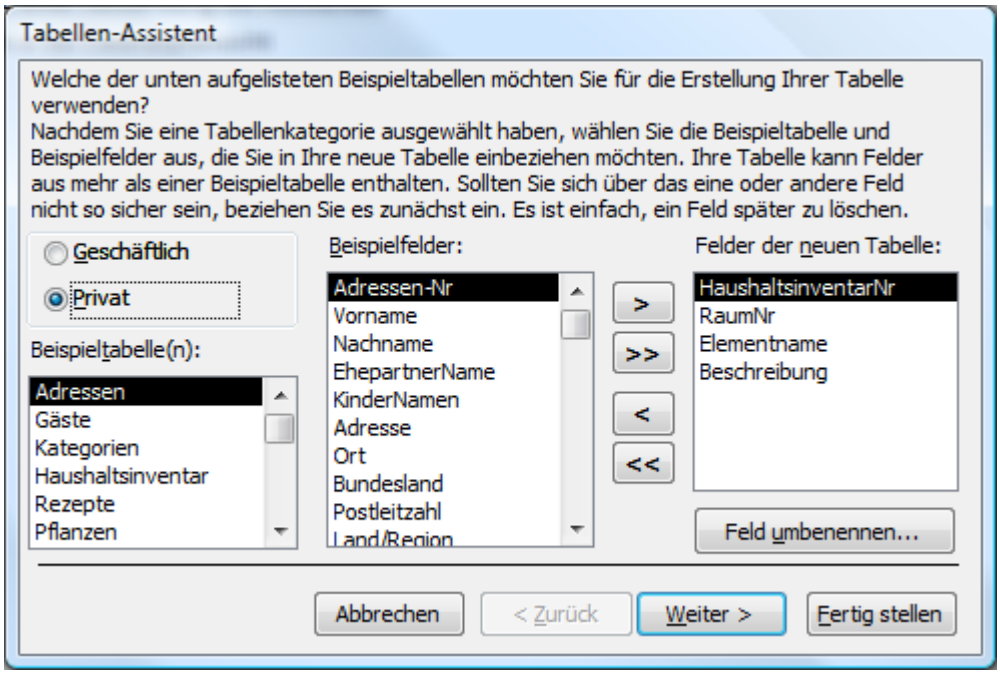

Fig. 21: Table wizard – MS Access

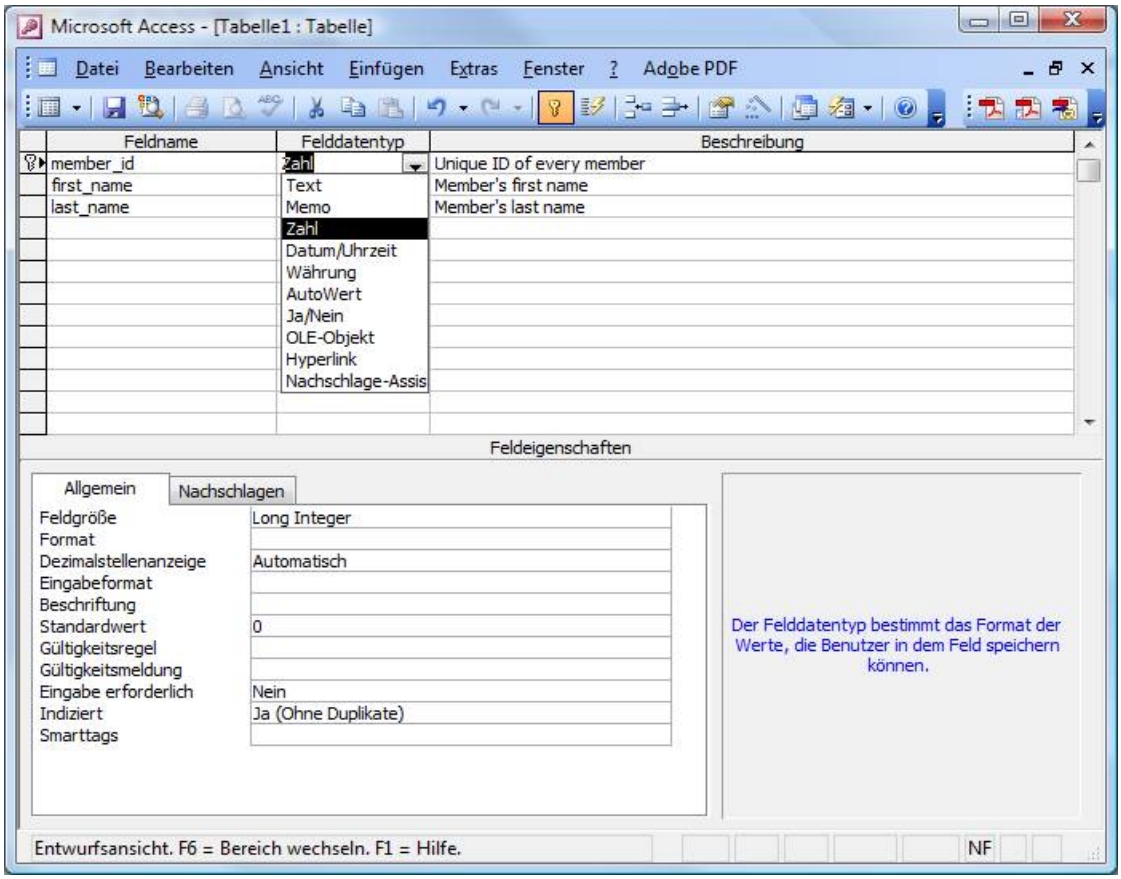

Fig. 22: Table design view MS Access

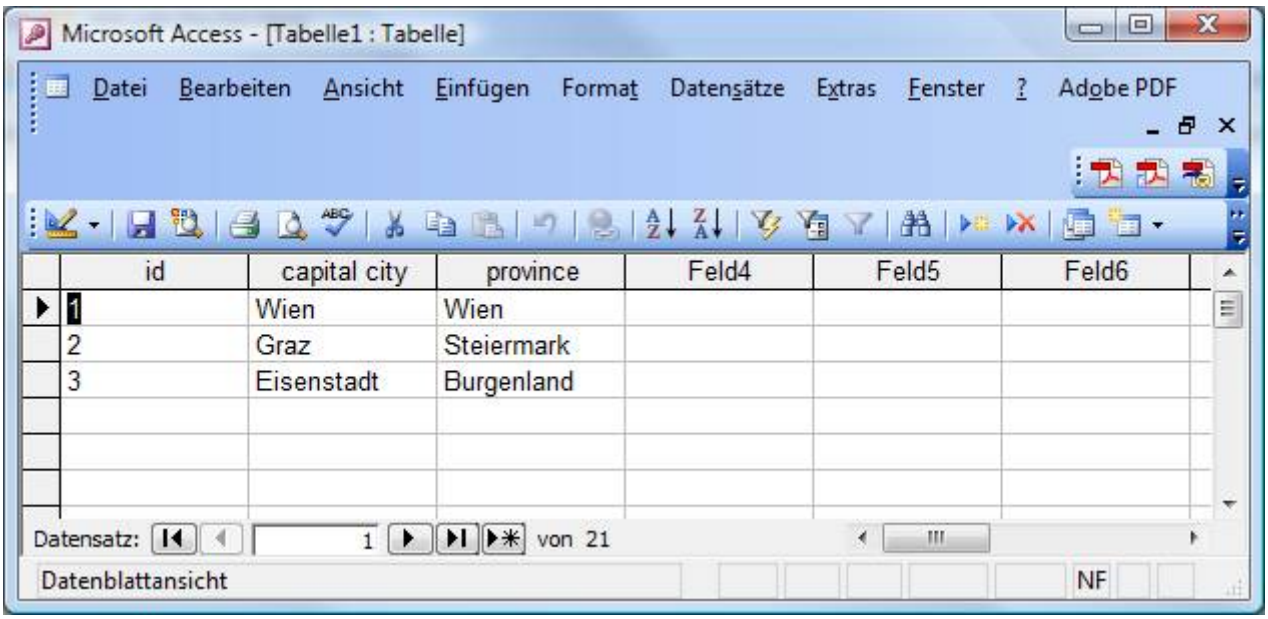

Fig. 23: Table entry MS Access

Queries

Queries in MS Access are much more powerful than in OOo Base. Different types of queries are availabe by default and need not be scripted: the select query, the crosstab query, the table create query, the update query, the append query and the delete query. All those queries have different parameter rows in the query design window. Queries can be created in two different ways. The first way is to use the query wizard. Contrary to OOo Base the wizard is not very powerful. You can only select one datasource (either tables or queries) and specify the fields to be displayed, you can choose whether you want to have a detailed query or a grouped one and basically that's it.

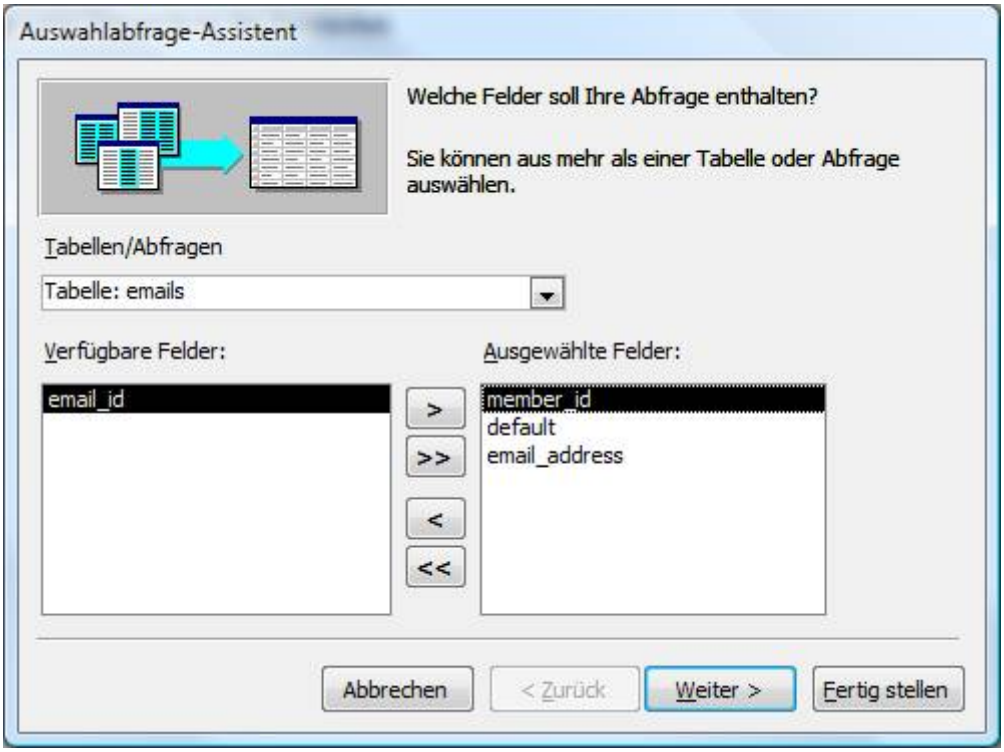

Fig. 24: Query wizard MS Access – select fields

The design view however is very powerful. Here you can combine different tables or queries, create and manage joins and set the parameters for the different types of queries. MS Access does not offer a seperate possibility for creating SQL queries directly, but that can be done by simply changing the view to SQL view. The MS Access SQL view can hold any kind of SQL statments that is implemented in the database and not only SELECT statements like in OOo Base. Beside that, every query can be used for entering data, no matter how many data sources there are in the query. But the creator of such a query needs to be aware of possible data inconsistencies as the program itself catches most exceptions but cannot know how the user intends the database to work.

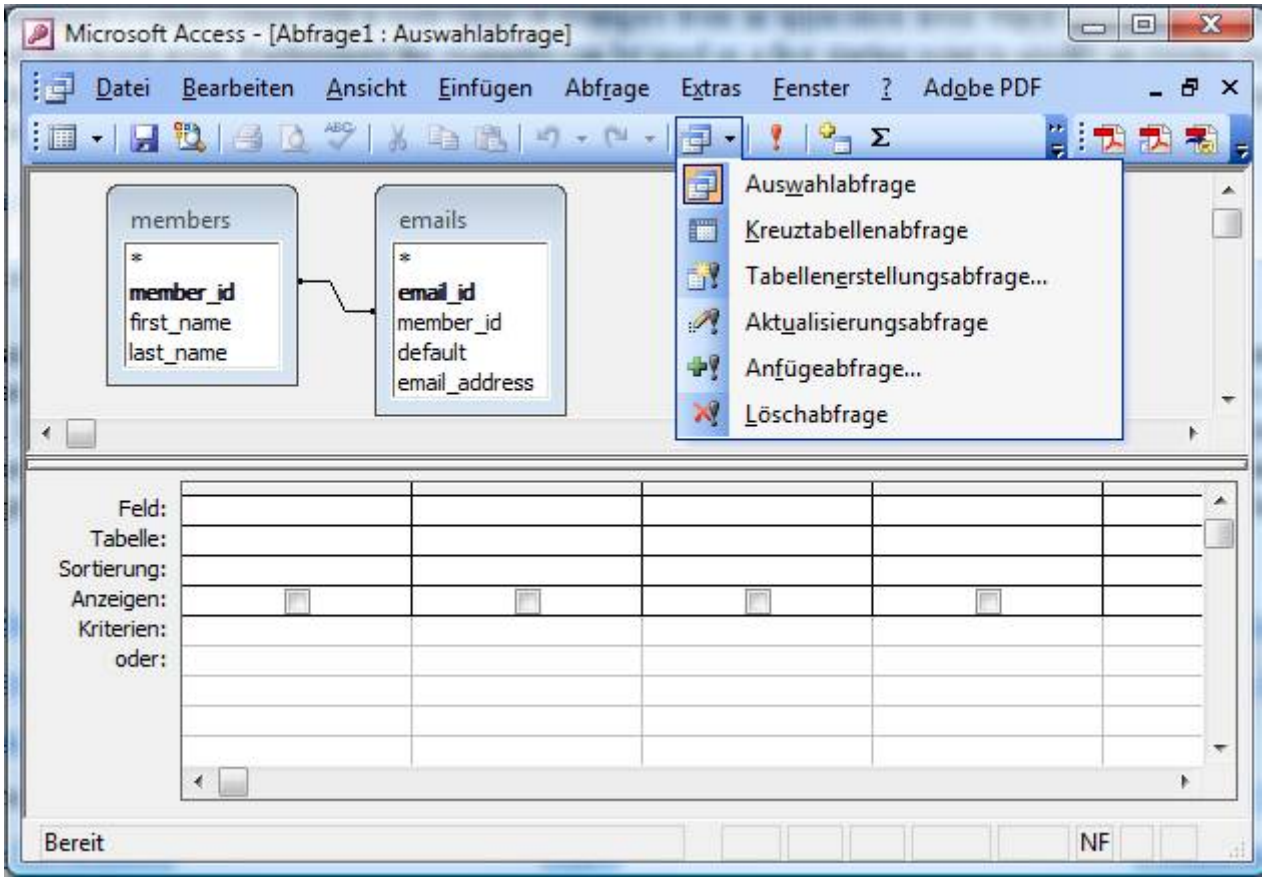

Fig. 25: Query design view

Forms

Forms can be used for displaying, entering and amending data. As tables and queries can be used for modifying data, forms based on these datasources can be used for that purpose as well. Restrictions can be made on form level by modifying the form properties.

For creating forms users may do so by using the form wizard. This wizard is suitable for simple forms with one datasource only. It is not possible to create subforms by using the wizard like in OOo Base, but it is possible to choose from predefined layouts with limited possibilities. For more sophisticated forms users have to use the forms design view. It's quite easy to create forms with multiple tabs and subforms there. Most of the properties which may be set are easy to adapt to the needed values.

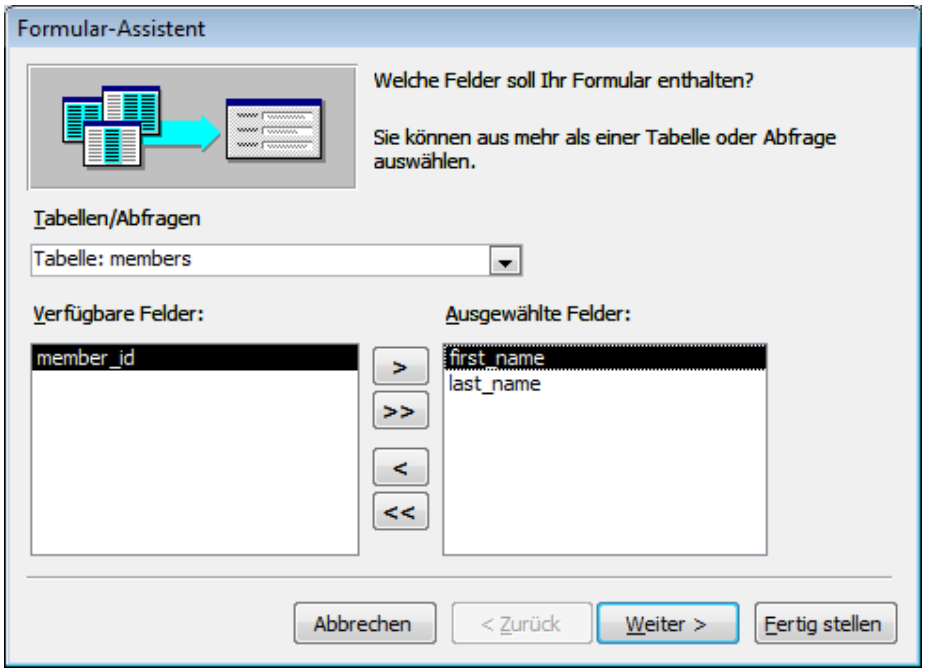

Fig. 26: Form wizard MS Access – select fields

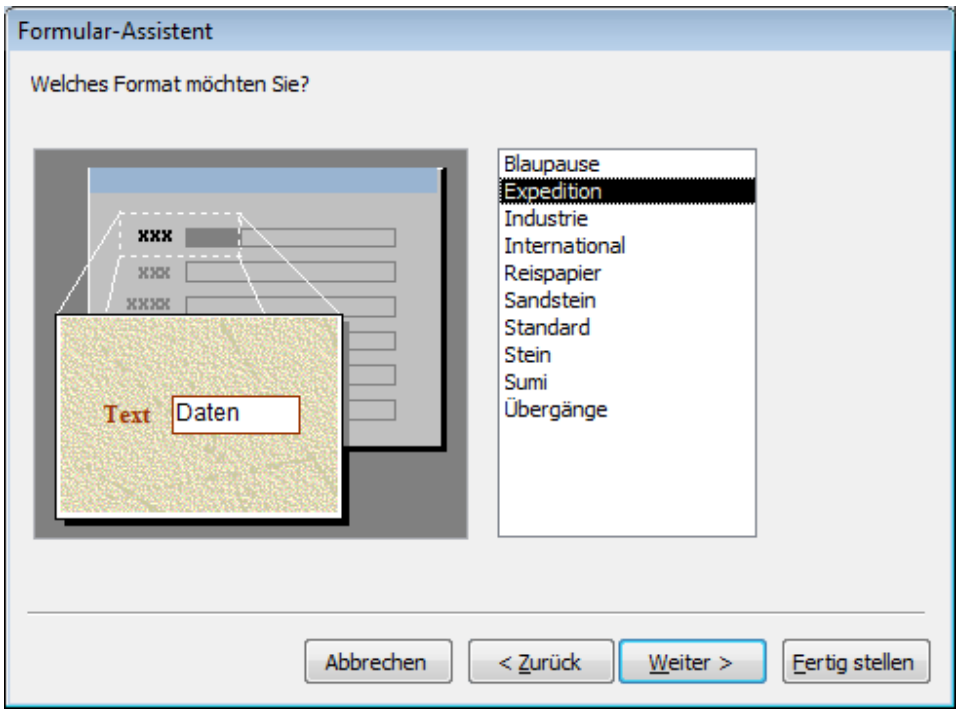

Fig. 27: Form wizard MS Access – choose layout

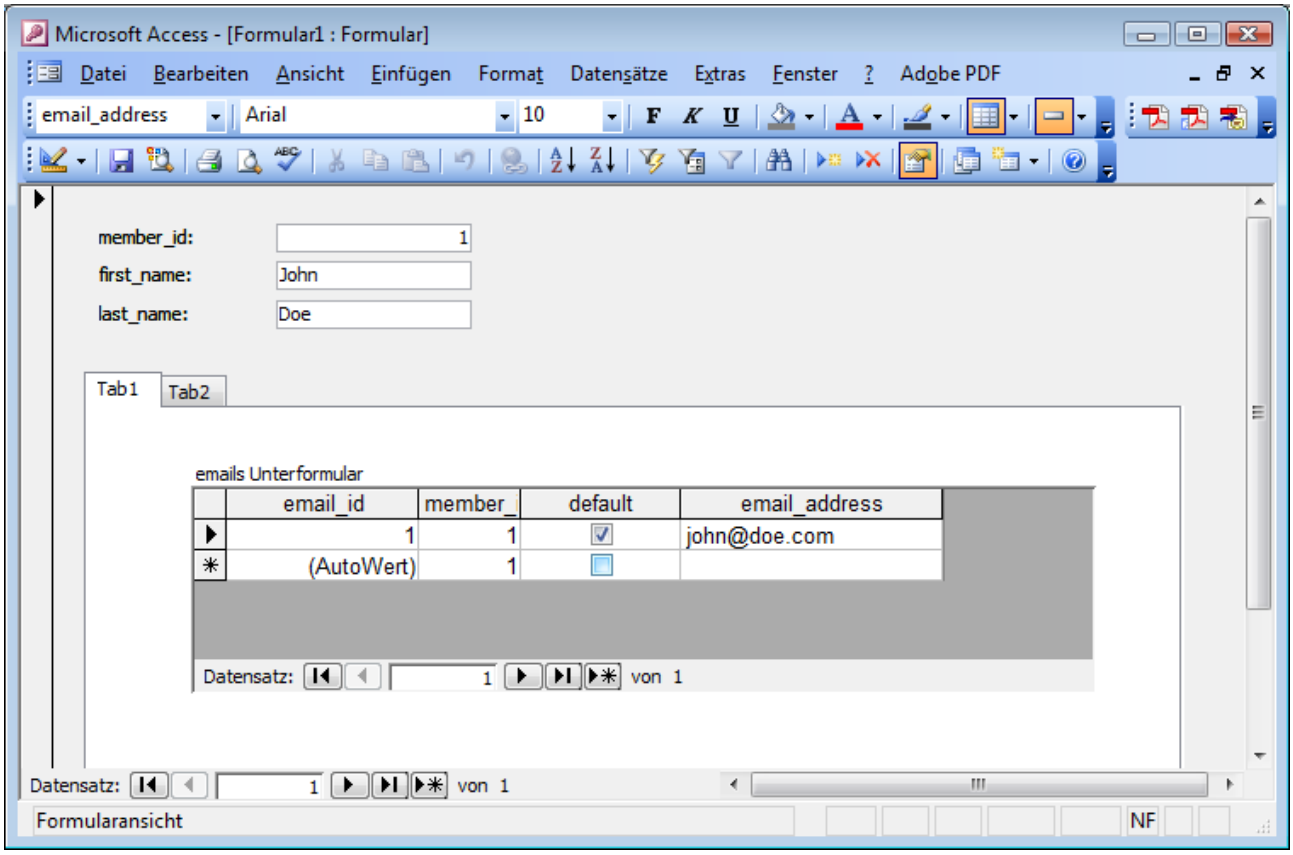

Fig. 28: Form MS Access

Reports

Reports in MS Access work similar to reports in OOo Base. They can only be used for displaying data, no data can be modified. The reports wizard is very similar to the OOo Base reports wizard. Only one datasource can be selected, no subreports can be included. It is suitable for quick reports, but for further reports it is necessary to enter the design mode. There it is not very difficult to build sophisticated reports with multiple subreports etc. Contrary to OOo Base reports, reports are always dynamic and cannot be saved as static reports.

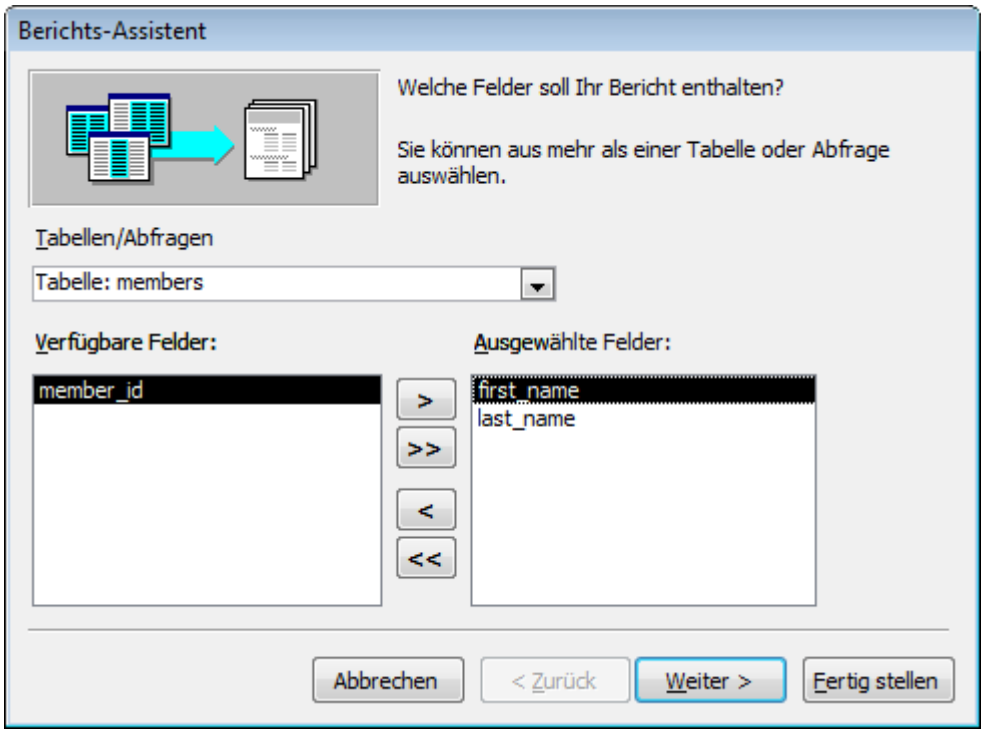

Fig. 29: Report wizard MS Access – select fields

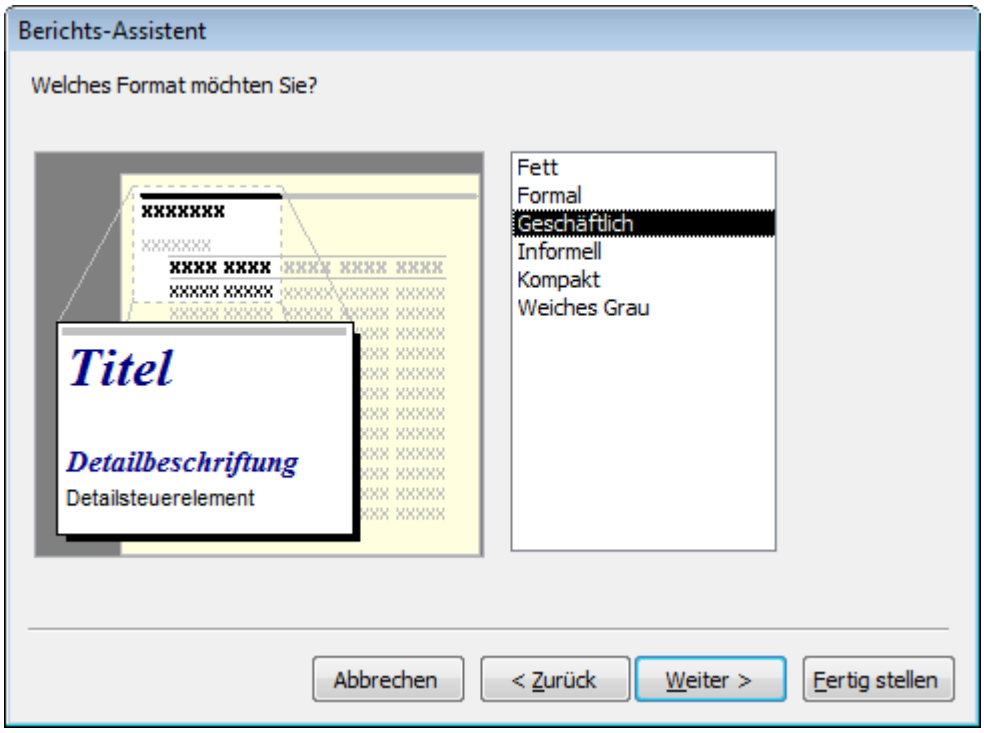

Fig. 30: Report wizard MS Access – select layout

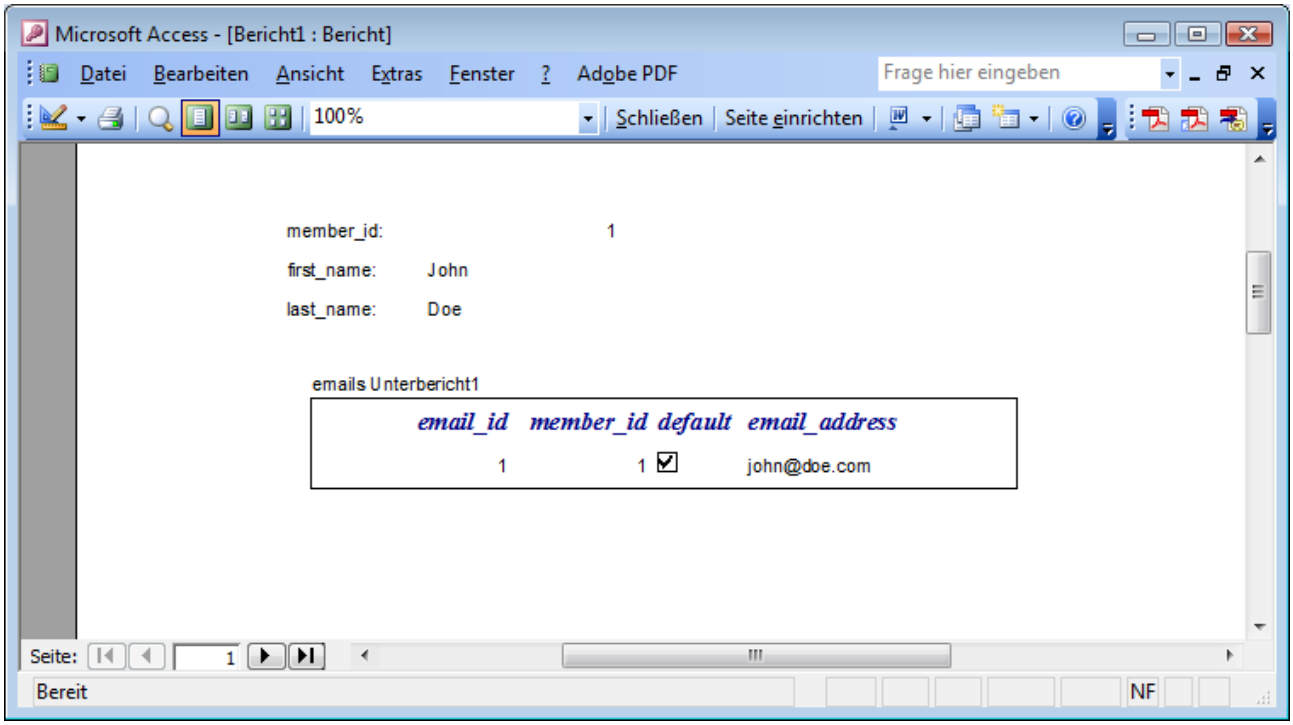

Fig. 31: Report MS Access

HTML Pages

HTML Pages offer the possibility of displaying and modifying data by opening a web page which then can be stored locally or within a network. It only possible to include single tables or queries, no subpages can be generated.

Fig. 32: HTML page MS Access – data entry

Macros

MS Access Macros are no macros as commonly understood as scripts written in a programming language but a collection of predefined functions which are stored in a macro object. OOo Base does not offer such a functionality.

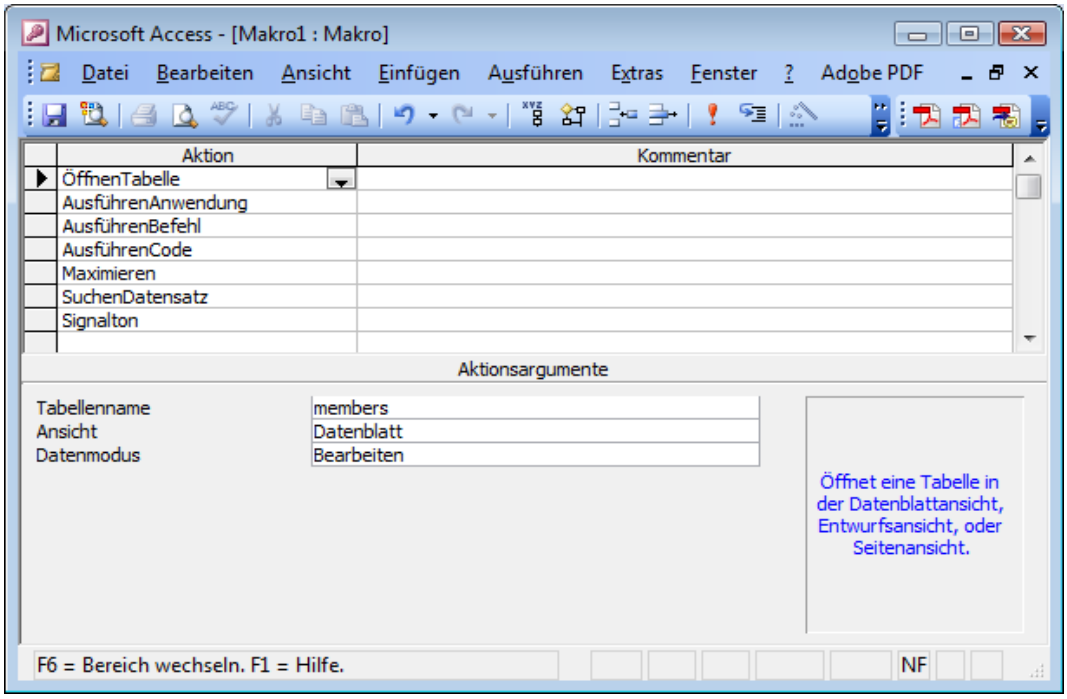

Fig. 33: Macro windows MS Access

Modules

Modules in MS Access offer the possibility to extend functionality by programming functions and macros. Modules are collections of such pieces of code and the used programming language is VBA.

1.2 Comparison Backend OOo Base - Microsoft Office Access

The backend is the physical composition of proprietary functions, methods and data design for providing database functionality. Backends usually don't have a GUI but different tools for administration.

1.2.1 Backend OOo Base

OOo Base comes with a full database engine which is HSQLDB. HSQLDB is a RDBMS entirely written in Java. Basically, HSQLDB can be set up in different modes. It can be run in different server modes (HSQLDB Server, HSQLDB Web Server and HSQLDB Servlet), in process (standalone) mode and as a memory only database. The server mode is the preferred and fastest mode. The memory only database should only be used for internal processing of application data as no information is written to disk. The standalone mode runs the database engine as part of an application in the same Java Virtual Machine. For many applications this mode can be faster as the data is not converted and sent over the network. The disadvantage of this mode is that while the application is running no other process is able to access the database file and cannot perform any action on it. [cp. HSQL08]

But the HSQLDB engine is just the database which comes with OOo Base by default. Natively, OOo Base can access many other databases as well, i.e. Adabas D, ADO, Microsoft Access, MySQL and dBase files. Furthermore, OOo Base can work with any database through standard ODBC and JDBC drivers. Additionally, LDAP compliant address books as well as common formats just like Microsoft Outlook, Microsoft Windows and Mozilla are supported. [cp. OOo08]

Examples for using OOo Base with other database engines will be described in a following chapter.

1.2.2 Backend Microsoft Office Access

The underlying database engine for Microsoft Office Access is the Microsoft Jet Red Database Engine. This engine is designed for desktop applications mainly, i.e. processing speed suffers when a Microsoft Office Access database is shared by multiple concurrent users. The developer edition of Office XP comes with MSDE (Microsoft SQL Server Desktop Engine) 2000 which is a basic version of MS SQL Server 2000 and may serve as an alternative to the Jet Red database engine. Access also includes an so called Upsizing Wizard that allows users to upsize their database to Microsoft SQL Server in order to use a true client-server database instead of the relatively weak capabilities of the Jet Red Engine in terms of scalability and multi user processing. Upsizing is a Microsoft term and means upgrading to the Microsoft SQL Server.
Similar to OOo Base, Access can be used as a frontend with different underlying databases like

Oracle or any ODBC-compliant data container. [cp. WIKI08]

The linking of the Microsoft Office Access frontend to other database files is no focus of this paper.

1.2.3 Common Aspects

Both default backends are suitable for smaller few concurrent users applications. They support a range of the ANSI SQL-92 standard, but lack complex functions like stored procedures or database triggers. The HSQLDB engine provides possibility for Java stored procedures and functions, though. Additionally, both engines are limited in space which restricts the possibilities of using them as backends for applications which handle large amounts of data. But as their structure and capabilities are limited their possible field of deployment is restrained to non critical applications handling smaller amounts of data in a non critical non highly available infrastructure environment. Furthermore, these systems have not achieved a status of high confidence in the sense of data loss security and high availability. [cp. HSQL08, WIKI08]

1.3 Getting Familiarized With OOo Base

This chapter can only be from a very subjective point of view as everybody has a different knowledge in working with new applications on one hand and in working with Microsoft Access and OOo Base on the other hand.

The basic functionality of both applications is the same. All different functions and objects have the same or a similar name and can be found in the same or similar places. Similar in this context means that the names are well chosen and the meaning can be intuitively understood. One thing that is different from MS Access is the use of the other Open Office applications as part of the base functionality like forms and reports which are done in OOo writer. Concerning the concept of Open Office and the use of the component model which is also the base for the UNO concept this use of only one module for similar tasks does not surprise. When coming from the Microsoft world this is a quite new idea. Suddenly the single objects are not propriety for the single applications but can be used throughout Open Offce.

Basic tasks like creating tables and queries are very much the same as in MS Access. The biggest difference and therefore the most complicated part was creating complex queries and forms for data input. This could not be done the way it is done in MS Access. Queries on more than one table cannot be taken for data input which is very confusing for an experienced MS Access user. Functionality however can easily be built up by scripting. OOo Basic is compatible to VBA, meaning that it is fairly easy to get quick results given that you know Microsoft Basic. Structures and syntax are the same, the objects follow the UNO model.

Finally, OOo Base is an application for experienced users and programmers. Ordinary users might be better with off MS Access as many things are done by the application without having to know about scripting. For the experienced user OOo Base offers a wide range of possibilities for linking it to other applications as well as to extend the functionality itself by scripting and programming.

2 OpenOffice.org And Scripting

Scripting and programming with OOo is done by using the UNO framework for accessing Open Office and is the API for the program. At the moment there are UNO bindings available for Java, C ++, OpenOffice.org Basic and the windows automation bridge (COM). [cp. OOSDK08] Furthermore, through the bean scripting framework it is possible to bind a lot of other scripting/programming languages to Java and therefore to Open Office. [cp. BSF08a]

2.1 Scripting Within The Application – OpenOffice.org Basic

OpenOffice.org Basic provides access to the OpenOffice.org API from within the office application. It hides the complexity of interfaces and simplifies the use of properties by making UNO objects look like Basic objects. It offers convenient functions and special Basic properties for UNO. Furthermore, Basic procedures can be easily hooked up to GUI elements, such as menus, toolbar icons and GUI event handlers. In Java and C++, it is necessary to obtain a reference to each interface before calling one of its methods. In Basic, every method of every supported interface can be called directly at the object without querying for the appropriate interface in advance. The '.' operator is used. [cp. OOSDK08]
Additionally it is possible to use other scripting languages within Open Office like ooRexx. This functionality is delivered by Java classes.

2.1.1 Scripting Example 1: Display Data Rows In Message Boxes

The first example queries the registered database and displays the results in popup windows. Note that starting OOo base is not required, only the UNO service „RowSet“ is created for sending the SQL statement to the database and receiving the results.

Fig. 34: Scripting Example 1: Message Box

For the sourcecode, please see Appendix A – Scripting Example 1: Display Data Rows In Message Boxes.

2.1.2 Scripting Example 2: Display Data Rows In OpenOffice.org Calc

The second example is very similar to the first one. The main difference is that the output is no message box but an OOo calc document. The script gets the data from the database, opens a calc document and writes the values into the cells. Again, no OOo base document is required, only the service „RowSet“ and a loaded calc component.

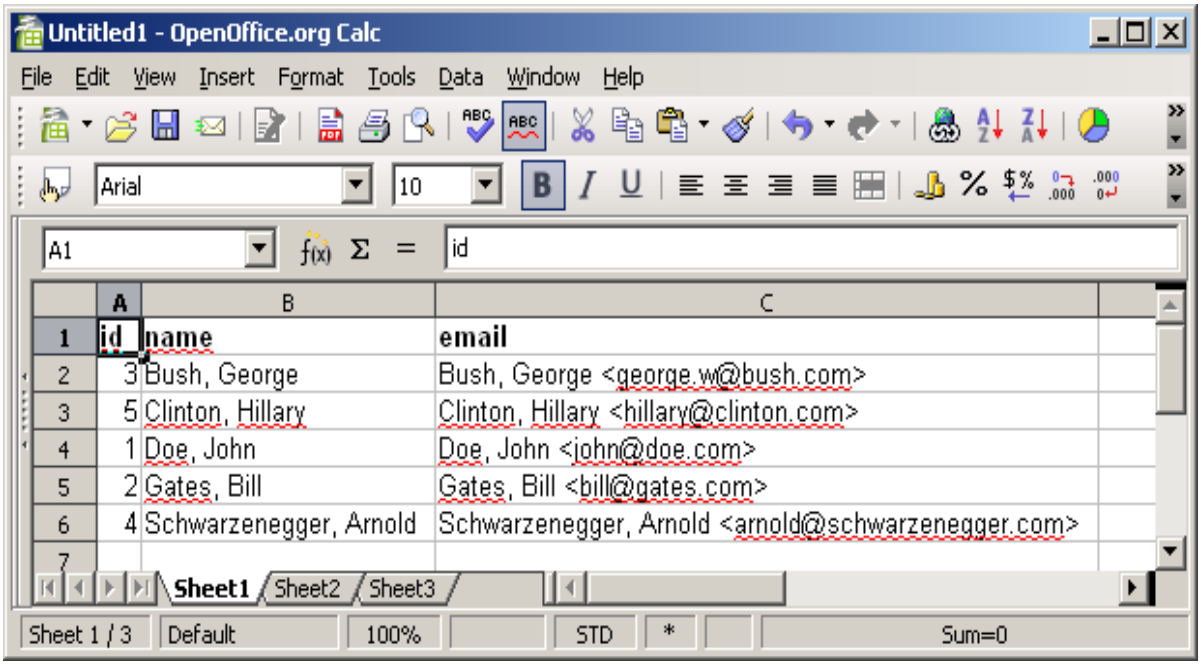

Fig. 35: Scripting Example 2: data in OOo Calc

For the sourcecode, please see Appendix A – Scripting Example 2: Display Data Rows In OpenOffice.org Calc.

2.1.3 Scripting Example 3: Display Data In OpenOffice.org Calc With Diagram

This example uses multiple services and basic elements. It uses a predefined dialogue to get the parameters for the report. The user selects the values from updated dropdown lists, the script gets

the according data from the database, puts the values in question into a OOo calc document and creates a pie diagram.

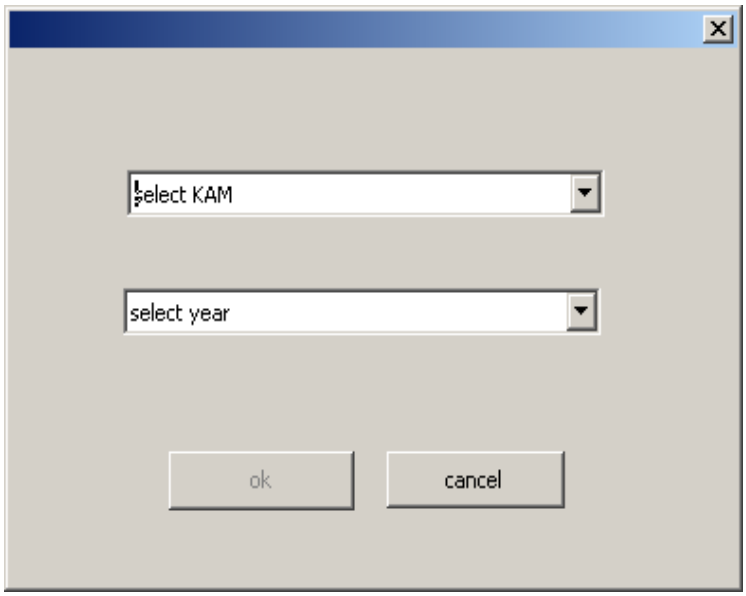

Fig. 36: Scripting Example 3: Dialog box

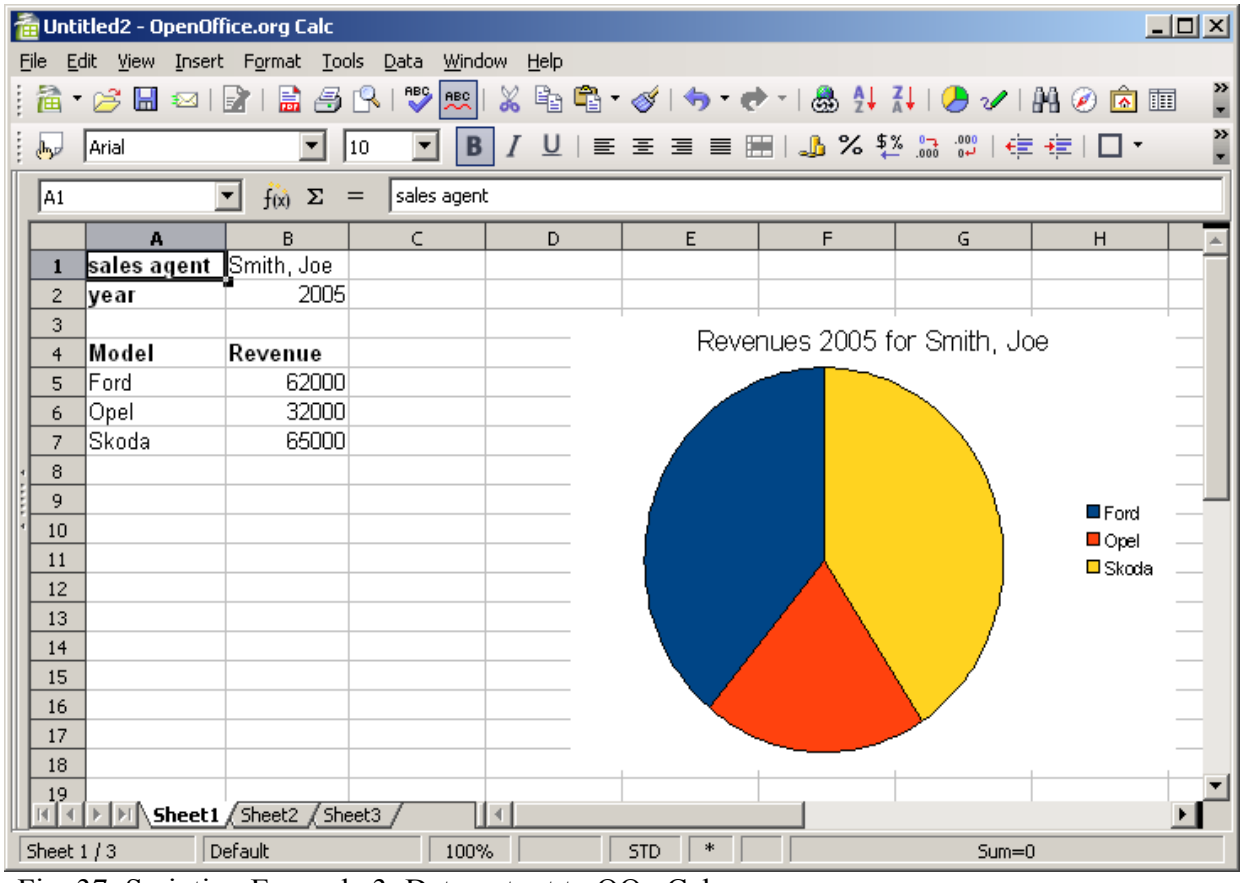

Fig. 37: Scripting Example 3: Data output to OOo Calc

For the sourcecode, please see Appendix A – Scripting Example 3: Display Data In OpenOffice.org Calc With Diagram.

2.2 Scripting From Outside – ooRexx And Java

Java can take advantage of the included UNO components designed for those purposes. A Java program just needs to reference the java classes coming with the OOo installation in order to use them for controlling Open Office. For ooRexx there is no such native way. Instead users can take advantage of the bean scripting framework for ooRexx, called bsf4rexx. This framework glues ooRexx to Java, i.e. it wraps java with ooRexx classes and functions and offers the functionality to the ooRexx programming language. [cp. OOSDK08, BSF08a]

2.2.1 Scripting Example 4: ooRexx – Print Data To Console

This script connects to the database, loops through the returned data rows and prints the data to the console. ooRexx gets the UNO support from the library UNO.CLS which wrapps the OOo Java classes.

Fig. 38: Scripting Example 4: ooRexx console output

For the sourcecode, please see Appendix A – Scripting Example 4: ooRexx – Print Data To Console.

2.2.2 Scripting Example 5: ooRexx – Insert Data Into OOo Writer

This script is similar to scripting example 4. The output is not sent to the console but to a OOo

writer document.

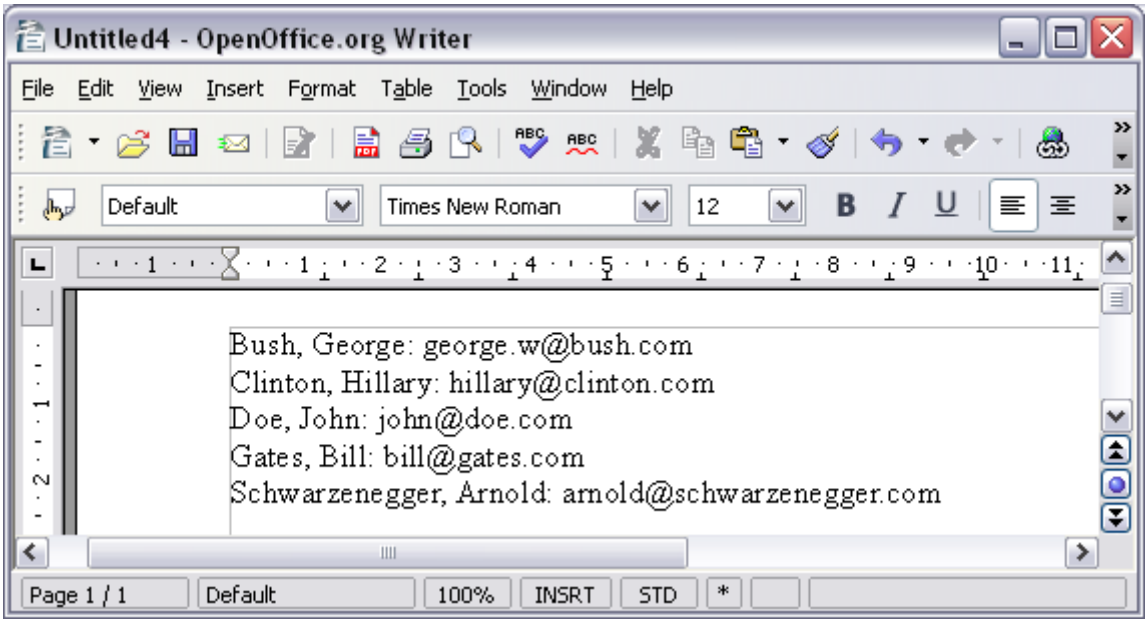

Fig. 39: Scripting Example 5: ooRexx output to OOo Writer

For the sourcecode, please see Appendix A – Scripting Example 5: ooRexx – Insert Data Into OOo Calc.

2.2.3 Scripting Example 6: ooRexx - Display Data In OpenOffice.org Calc With Diagram

This example asks for the parameters for the report. The user types the required values which are shown to the user in, the script gets the according data from the database, puts the values in question into a OOo calc document and creates a pie diagram.

For the sourcecode, please see Appendix A – Scripting Example 6: ooRexx – Display Data in OpenOffice.org Calc With Diagram.

2.2.4 Scripting Example 7: Java – Print Data To Console

This script connects to the database, executes the query and prints the resultset to the console.

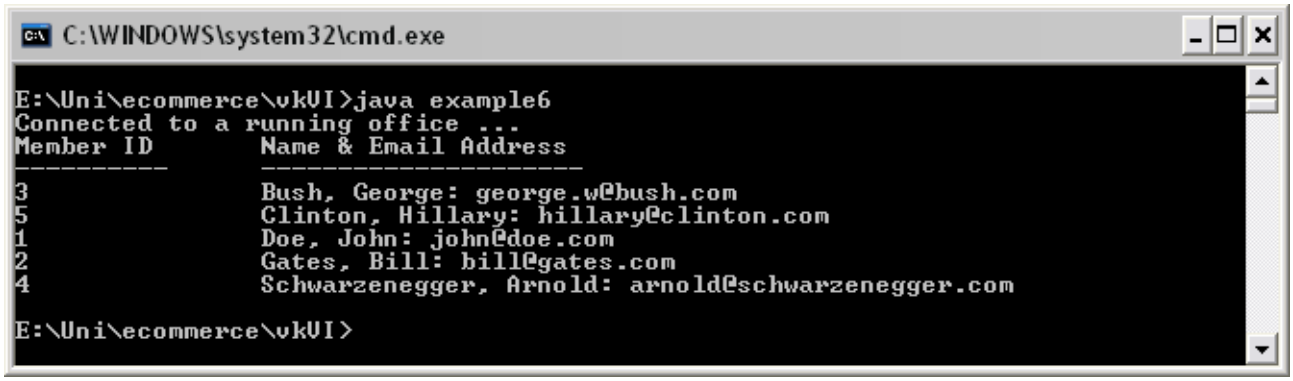

Fig. 40: Scripting Example 6: Java console output

For the sourcecode, please see Appendix A – Scripting Example 6: Java – Print Data To Console.

2.2.5 Scritping Example 8: Java – Insert Data Into OOo Writer

This script is similar to scripting example 6. The output however is not written to the console but to an empty OOo writer document.

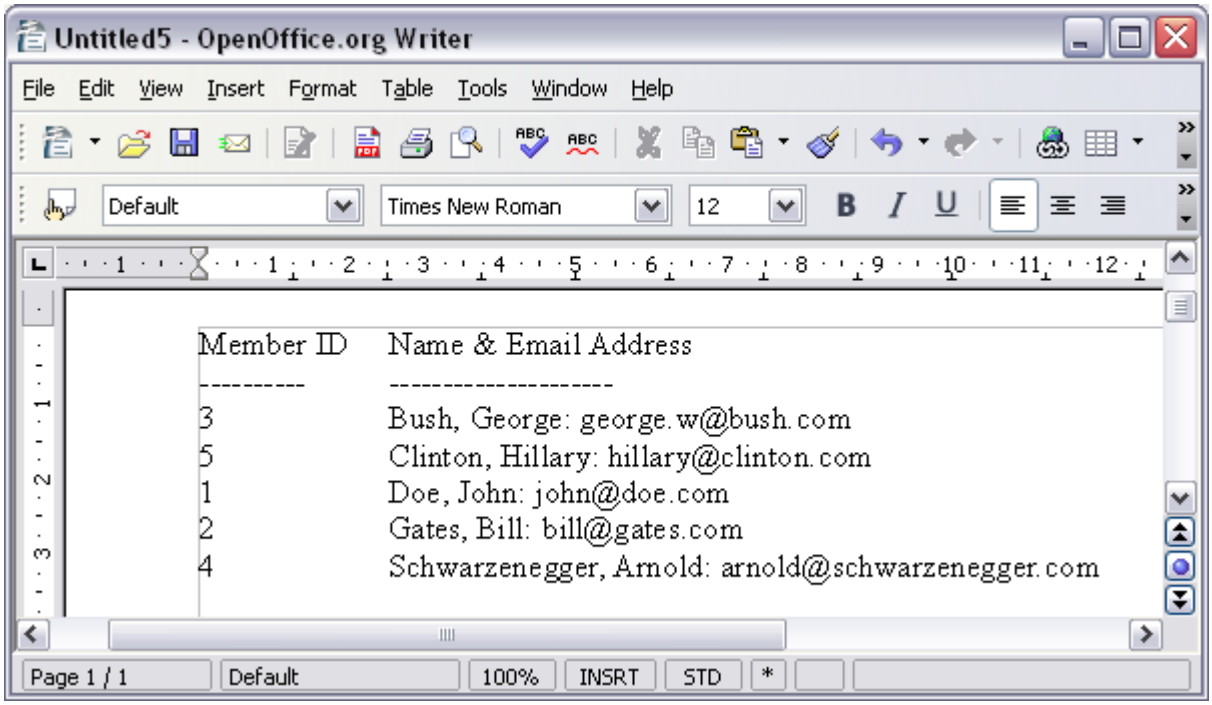

Fig. 41: Scripting Example 7: Java output to OOo Writer

For the sourcecode, please see Appendix A – Scripting Example 7: Java – Insert Data Into OOo

Writer.

2.2.6 Scripting Example 9: Java - Display Data In OpenOffice.org Calc With Diagram

This example needs to be invoked with the required paramters for the key account manager and the rquired year. In case it is started without parameters, the possible values are printed to the console.The example gets the according data from the database, puts the values in question into a OOo calc document and creates a pie diagram.

3 Working On Different Databases

The database engine provided by OOo Base will probably be sufficient for many end user purposes. Nevertheless, sometimes when changing or building up appliacations for end users already existing databases shall not be touched and remain as they are. That means that the end user gets a different or new front end which provides all possibilities for displaying, entering and amending data. The ways of connecting OOo Base as a front to different data sources are shown in this chapter by building up links to two widespread database engines MySql and MS Access.

3.1 Connecting To MySql

In this example a JDBC connection is used for connecting to MySql. JDBC is an API for JAVA which provides methods for querying and updating data in a database. [cp. JDBC08] As there are JDBC drivers for many database engines the setup for MySql may be taken as an example and template for the setup of any JDBC database connection within OOo Base.

The first precondition for the use of a JDBC connector is a JAVA installation on your computer. Additionally, Open Office has to be configured to use this JAVA installation. The configuration can be checked in the *Tools* > *Options* > *OpenOffice.org* > *Java* menu tab of Open Office. Make sure Open Office uses the same installation as your system does by executing the command *„java -version“* on the command line.

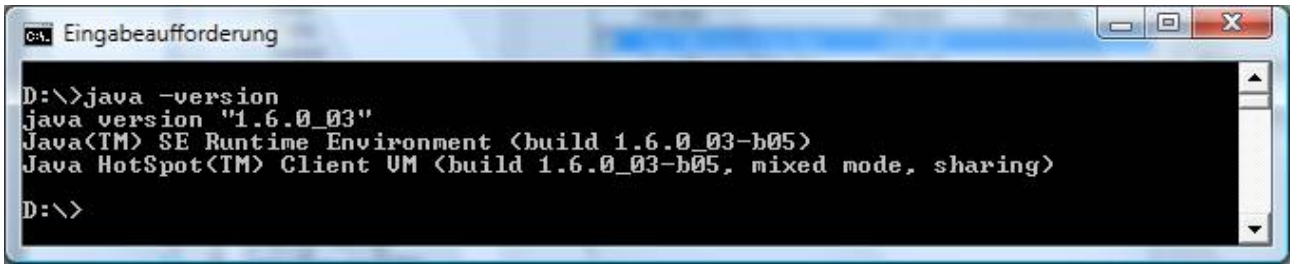

Fig. 42: Java version info

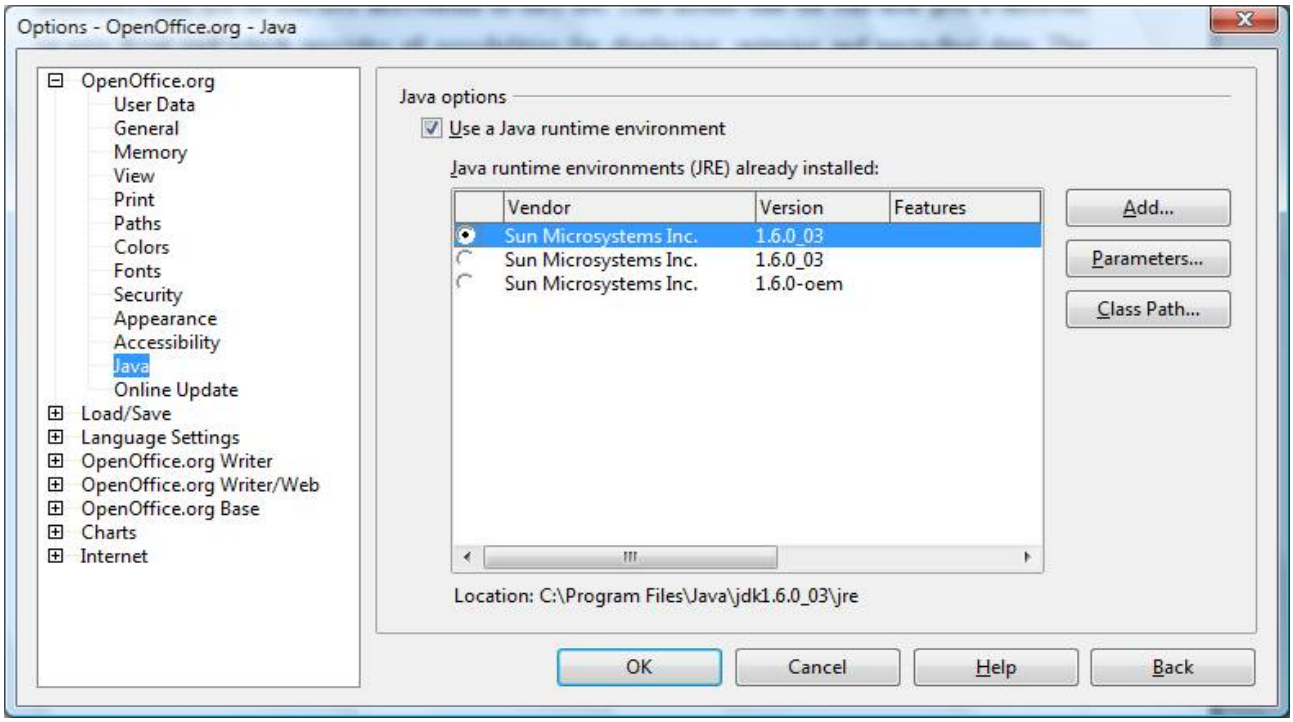

Fig. 43: OOo java settings

The next step is to get the appropriate JDBC connector class. Go to the homepage of the database engine provider (http://www.mysql.com in this case), get the „MySql Connector/J“ and unzip it to a directory of your choice. Then go to the *Tools* > *Options* > *OpenOffice.org* > *Java* tab and click the classpath button there. In the popup window add the unzipped archive to the classpath. You have to restart Open Office (including the quick starter) afterwards in order to make Oppen Office take effect of the changes.

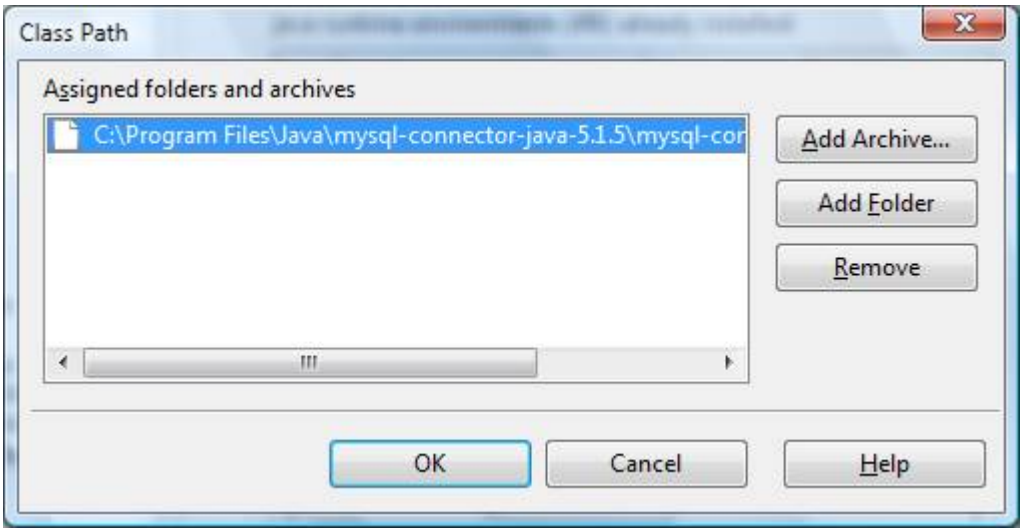

Fig. 44: Java class path in OOo

Now a new OOo base file can be created which is used as the frontend for the MySql database. When creating a new OOo base file choose *connect to an existing database* and select *MySql*. The second step is to choose the JDBC connectivity and to enter the connection parameters. At last save and use the OOo base file linked to the MySql database like an ordinary base file. The user will not notice that the database engine behind the OOo base GUI is pure MySql. [cp. Pito07]

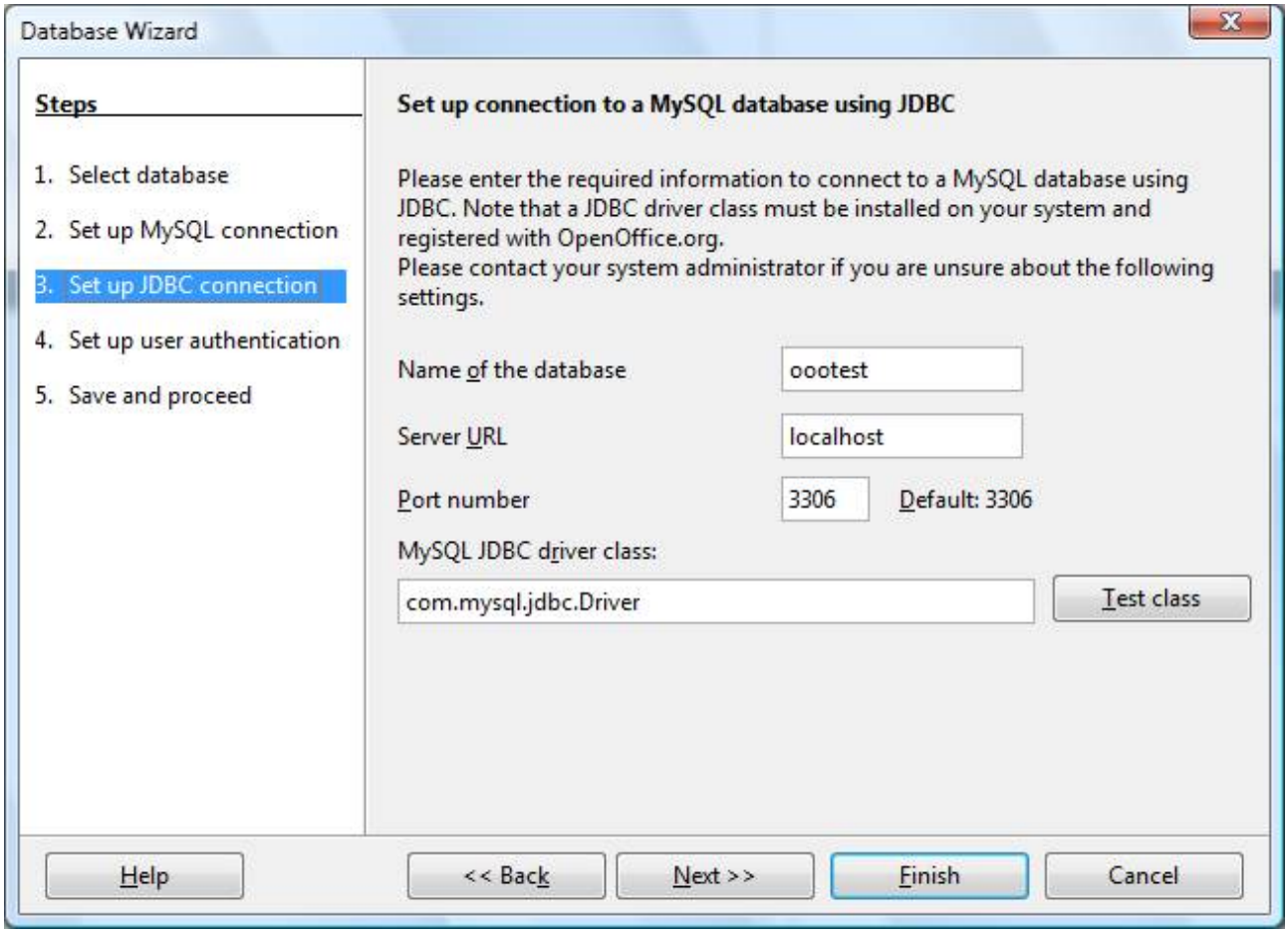

Fig. 45: Setting up database connection

3.2 Connecting To MS Access

Connecting to a MS Access database is very easy as OOo base is able to connect natively. When creating a new OOo base file choose *connect to an existing database* and select *Microsoft Access*. The next step is to select an existing MS Access database file and to save the OOo base file. The user can now use the OOo base file for any manipulation of the MS Access database file.

Fig. 46: Connecting to MS Access

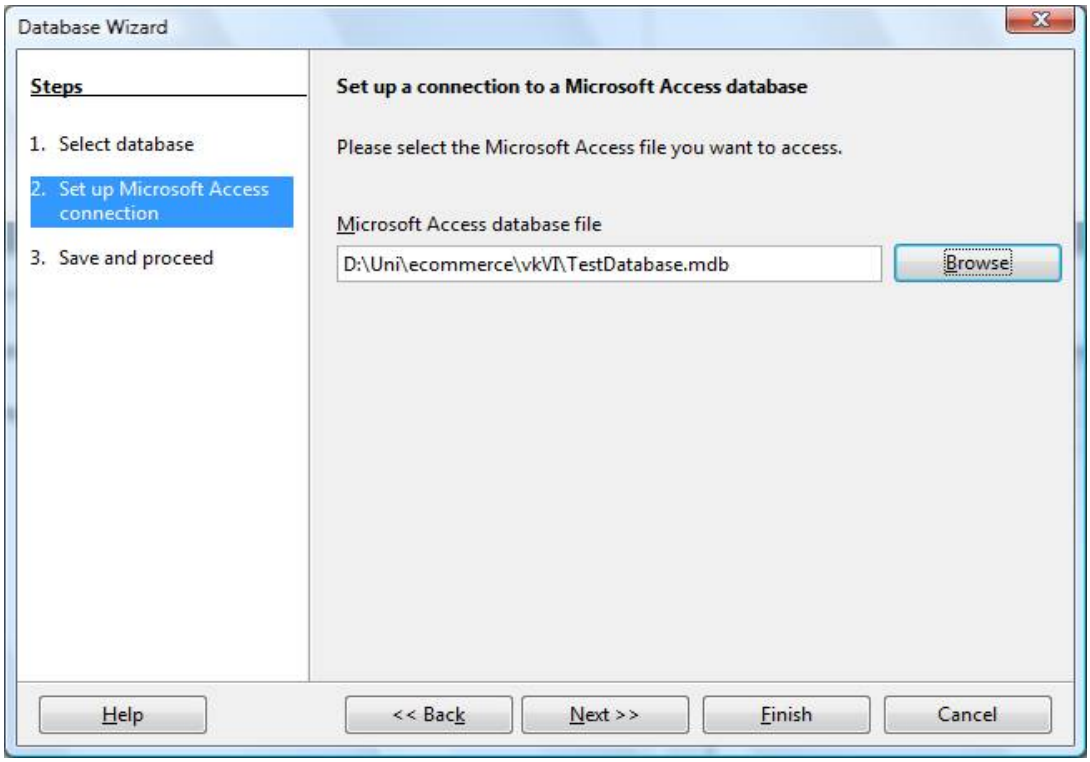

Fig. 47: Connecting to MS Access

3.3 Things To Be Aware Of

If the database is ment to be part of any scripts, automated process etc. within Open Office, it always needs to be registered within the application. Those base files linked to different database files are treated like ordinary OOo base files.

By using OOo base as a frontend, default data manipulation is limited by the application. Not all database specific methods, functions, datatypes etc. can be used by default. The problem can be solved on one hand by programming a user interface where it is possible either to send an SQL statement directly to the database engine or to select the appropriate commands from a GUI. On the other hand it can be reasonable to do the administration by using a frontend for the database engine in question and to use the OOo base frontend only as an data query and modification tool for the frontend users only.

4 Summary

OOo Base is a database application with a lot of funcionality. The application however is not always easy to work with. It is mighty, but not very user friendly so far. It is not difficult to create a database and it is not difficult to create queries and reports. Difficulties come up once you want to create more complex queries as a base for user defined forms. Complex forms for input and reporting are possible but need to be scripted. Depending on what the purpose of the usage is an OOo base database might be sufficient. Very convenient is the possibility of working with OOo components without having to start the applications themselves. This way users or applications can take advantage of functionality without having to start the respective application which is one of the greatest advantages from my point of view. Open Office base does not have all the functionality Microsoft Office Access has for the end user, but everything can be done with means of scripting and programming by the more experienced user.

5 Abbrevation Catalogue

API	Application Programming Interface
BSF	Bean Scripting Framework
BSF4REXX	Bean Scripting Framework for Rexx
BLOB	Binary Large Object
GUI	Graphical User Interface
HSQLDB	Hypersonic SQL database
JDBC	Java Database Connectivity
MS Access	Microsoft Access
MSDE	Microsoft SQL Server Desktop Engine
ODBC	Open Database Connectivity
OOo	OpenOffice.org
OOo Base	OpenOffice.org database application
PDF	Portable Document Format
RDBMS	Relational Database Management System
SQL	Structured Query Language
UNO	Universal Network Objects
VBA	Visual Basic for Applications

6 References

[Aham05]
Ahammer, Andreas: OpenOffice.org Automation: Object Model, Scripting Languages, „Nutshell"-Examples. Vienna University of Economics and Business Administration, 2005-11-06.

[Augu05]
Augustin, Walter: Examples for Open Office Automation with Scripting Languages. Vienna University of Economics and Business Administration, 2005-01-10.

[BSF08a]
The Apache Jakarta Project – Bean Scripting Framework
http://jakarta.apache.org/bsf/, accessed 2008-01-03

[BSF08b]
BSF4Rexx, version 2.6 ("The Vienna Version of BSF4Rexx"), 2007-01-28.
http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/, accessed 2008-01-03

[Flat05a]
Flatscher, Rony G.: AUTOMATING OPENOFFICE.ORG WITH OOREXX: ARCHITECTURE, GLUING TO REXX USING BSF4REXX. The 2005 international Rexx Symposium, Los Angeles, California. U.S.A., April 17th – April 21st, 2005.

[Flat05b]
Flatscher, Rony G.: AUTOMATING OPENOFFICE.ORG WITH OOREXX: OOREXX NUTSHELL EXAMPLES FOR WRITE AND CALC. The 2005 international Rexx Symposium, Los Angeles, California. U.S.A., April 17th – April 21st, 2005.

[Fosd05]
Fosdick, Howard: Rexx. Programmer's Reference. Wiley Publishing, Indianapolis, 2005.

[HSQL08]
HSQL Database Engine
http://www.hsqldb.org, accessed 2008-01-03

[JDBC08]
The Java Database Connectivity
http://java.sun.com/javase/technologies/database/, accessed 2008-01-03

[OOo08]
OpenOffice.org
http://www.openoffice.org, accessed 2008-01-03

[OOAPI08]
OpenOffice.org API Project
http://api.openoffice.org/, accessed 2008-01-03

[OOCS08]
OpenOffice.org Code Snippets
http://codesnippets.services.openoffice.org, accessed 2008-01-03

[OOSDK08]
The OpenOffice.org Software Development Kit
http://download.openoffice.org/2.3.0/sdk.html, accessed 2008-01-03

[OOSUP08]
OpenOffice.org free community support with forums, mailing lists, documentation etc.
http://support.openoffice.org,accessed 2008-01-03

[Pito07]
Pitonyak, Andrew: Using Macros With OOo Base, 2007-07-23.
http://www.pitonyak.org/database/AndrewBase.pdf, accessed 2008-01-03

[Rexx08]
The Open Object Rexx (ooRexx) Open Source project
http://www.oorexx.org/, accessed 2008-01-03

[WIKI08]
Microsoft Access
http://en.wikipedia.org/wiki/Microsoft_Access, accessed 2008-01-03

7 Appendix A – Source Codes

7.1 Scripting Example 1: Display Data Rows In Message Boxes

```
Sub Main
    dim numRows as integer

    ' Create a row-set to query the database
    RowSet = createUnoService("com.sun.star.sdb.RowSet")
    RowSet.DataSourceName = "TestDatabase"
    RowSet.CommandType = com.sun.star.sdb.CommandType.COMMAND
    RowSet.Command = "SELECT member_id, first_name, last_name FROM members"
    RowSet.execute()

    ' loop through the resultset until no record is left
    while RowSet.next()
        MsgBox "The member # " + rowSet.getString(1) + " is called " + rowSet.getString(2) + " " _
                + rowSet.getString(3)
    wend

End Sub
```

7.2 Scripting Example 2: Display Data Rows In OpenOffice.org Calc

```
Private oDoc as Object
Private members() as string
Private memails() as string
Private mids() as string
Private numRows as integer

Sub Main
   call initialQuery
   call doCalcDoc
End Sub


sub doCalcDoc
        dim x as integer
        dim col as object
        Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
        Dim url

        'Open a blank sheet in calc
        url = "private:factory/scalc"
        oDoc = StarDesktop.loadComponentFromURL(url,"_blank", 0, FileProperties())

        oSheet = oDoc.getSheets().getByName("Sheet1")

        If Not IsNull(oDoc) Then
                xCell = oSheet.getCellByPosition(0,0)
                xCell.setFormula("id")
                xCell.CharWeight = 140
                xCell = oSheet.getCellByPosition(1,0)
                xCell.setFormula("name")
                xCell.CharWeight = 140
                xCell = oSheet.getCellByPosition(2,0)
                xCell.setFormula("email")
                xCell.CharWeight = 140

                for x = 1 to numrows
```

```
                        xCell = oSheet.getCellByPosition(0,x)
                        xCell.setFormula(mids(x))
                        xCell = oSheet.getCellByPosition(1,x)
                        xCell.setFormula(members(x))
                        xCell = oSheet.getCellByPosition(2,x)
                        xCell.setFormula(memails(x))
                next x

                for x = 0 to 2
                        col = oSheet.Columns(x)
                        col.OptimalWidth = true
                next x
        end if
end sub


Sub initialQuery
        dim x as integer

    ' Create a row-set to query the database
    RowSet = createUnoService("com.sun.star.sdb.RowSet")
    RowSet.DataSourceName = "TestDatabase"
    RowSet.CommandType = com.sun.star.sdb.CommandType.COMMAND
    RowSet.Command = "SELECT member_id, first_name, last_name, email_address FROM _
                      member_defemail_view order by last_name,first_name,member_id"
    RowSet.execute()

    RowSet.last()
    numRows = RowSet.RowCount
        redim members(numRows)

    RowSet.first()

        redim members(numRows)
        redim memails(numRows)
        redim mids(numRows)

    for x = 1 to numRows
  members(x) = rowSet.getString(3) + ", " + rowSet.getString(2)
  memails(x) = rowSet.getString(3) + ", " + rowSet.getString(2) + " <" + rowSet.getString(4) _ + ">"
        mids(x) = rowSet.getString(1)
        RowSet.next()
    next x
end Sub
```

7.3 Scripting Example 3: Display Data In OpenOffice.org Calc With Diagram

```
Dim oDialog As Object
dim lstKams As Object
dim lstYears as Object
dim selKam as string
dim selYear as string
dim cmdOk as Object
dim kamDef as string
dim yearDef as string
private oDoc as Object

Sub Main()

        DialogLibraries.LoadLibrary("Standard")
        oDialog = CreateUnoDialog(DialogLibraries.Standard.revenues)

        'initialize variables
        kamDef = "select KAM"
        yearDef = "select year"
```

```
        'get the dropdown lists from the dialogue
        lstKams = oDialog.getControl("selectKam")
        lstYears = oDialog.getControl("selectYear")

        lstKams.Model.Text = kamDef
        lstYears.Model.Text = yearDef

        cmdOk = oDialog.getControl("CommandButton1")
        cmdOk.Model.Enabled = False

        'gets the return value from the dialogue
        Select Case oDialog.Execute()
        Case 0
           msgbox("cancelled by user")
        Case 1
           selKam = lstKams.SelectedText
           selYear = lstYears.selectedtext
           drawChart(selKam, selYear)
        End Select

        oDialog.dispose()
End Sub


'draws the chart in calc
Sub drawChart(selKam as string, selYear as string)
        dim x as integer
        dim col as object
        dim Charts as object
        dim Chart as object

        Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
        Dim url
        url = "private:factory/scalc"
        oDoc = StarDesktop.loadComponentFromURL(url,"_blank", 0, FileProperties())

        oSheet = oDoc.getSheets().getByName("Sheet1")

        If Not IsNull(oDoc) Then
                xCell = oSheet.getCellByPosition(0,0)
                xCell.setFormula("sales agent")
                xCell.CharWeight = 140
                xCell = oSheet.getCellByPosition(1,0)
                xCell.setFormula(selKam)
                xCell = oSheet.getCellByPosition(0,1)
                xCell.setFormula("year")
                xCell.CharWeight = 140
                xCell = oSheet.getCellByPosition(1,1)
                xCell.setFormula(selYear)
                xCell = oSheet.getCellByPosition(0,3)
                xCell.setFormula("Model")
                xCell.CharWeight = 140
                xCell = oSheet.getCellByPosition(1,3)
                xCell.setFormula("Revenue")
                xCell.CharWeight = 140

                RowSet = createUnoService("com.sun.star.sdb.RowSet")
            RowSet.DataSourceName = "TestDatabase"
       RowSet.CommandType = com.sun.star.sdb.CommandType.COMMAND
       RowSet.Command = "SELECT model, revenue FROM revenues, kams, models WHERE ryear="
       RowSet.Command = RowSet.Command + selYear + " AND concat(concat(kam_lname,', '),kam_fname)='"
       RowSet.Command = RowSet.Command + selKam + "' and revenues.kam_id = kams.kam_id "
       RowSet.Command = RowSet.Command + "and revenues.model_id = models.model_id order by
model"
        RowSet.execute()

            RowSet.last()
        numRows = RowSet.RowCount
            RowSet.first()

                for x = 1 to numrows
                        xCell = oSheet.getCellByPosition(0,x+3)
                        xCell.setFormula(rowSet.getString(1))
```

```
                        xCell = oSheet.getCellByPosition(1,x+3)
                        xCell.setFormula(rowSet.getString(2))
                        RowSet.next()
                next x

                for x = 0 to 2
                        col = oSheet.Columns(x)
                        col.OptimalWidth = true
                next x
        end if

        'get the rectangle service
        Dim Pie as new com.sun.star.awt.Rectangle
        Dim RangeAddress(0) as new com.sun.star.table.CellRangeAddress

        Charts = oSheet.Charts

        Pie.X = 8000
        Pie.Y = 1000
        Pie.Width = 10000
        Pie.Height = 7000

        RangeAddress(0).Sheet = 0
        RangeAddress(0).StartColumn = 0
        RangeAddress(0).StartRow = 3
        RangeAddress(0).EndColumn = 1
        RangeAddress(0).EndRow = 3 + numRows

        Charts.addNewByName("revenues", Pie, RangeAddress(), True, True)

        Chart = Charts.getByName("revenues").embeddedObject
        Chart.Diagram = Chart.createInstance("com.sun.star.chart.PieDiagram")

        Chart.HasMainTitle = True
        Chart.Title.String = "Revenues " + selYear + " for " + selKam
end sub

'enables the Ok button when selections are done
Sub proofOk
        if ((lstKams.SelectedText <> kamDef) AND (lstYears.selectedtext <> yearDef) AND _
                (lstKams.SelectedText <> "") AND (lstYears.selectedtext <> "")) THEN
                cmdOk.Model.Enabled = True
        end if
end sub

'initializes dropdown list for key account manager
Sub initSelectKam()

    ' Create a row-set to query the database
    RowSet = createUnoService("com.sun.star.sdb.RowSet")
    RowSet.DataSourceName = "TestDatabase"
    RowSet.CommandType = com.sun.star.sdb.CommandType.COMMAND
    RowSet.Command = "SELECT kam_long from kams_query order by kam_long"
    RowSet.execute()

    RowSet.last()
    numRows = RowSet.RowCount
    RowSet.first()

    for x = 1 to numRows
        lstKams.addItem(rowSet.getString(1),x-1)
        RowSet.next()
    next x
end sub

'initializes dropdown list for reporting year
Sub initSelectYear()

    ' Create a row-set to query the database
    RowSet = createUnoService("com.sun.star.sdb.RowSet")
    RowSet.DataSourceName = "TestDatabase"
    RowSet.CommandType = com.sun.star.sdb.CommandType.COMMAND
    RowSet.Command = "SELECT distinct ryear from revenues order by ryear"
    RowSet.execute()
```

```
    RowSet.last()
    numRows = RowSet.RowCount
    RowSet.first()

    for x = 1 to numRows
        lstYears.addItem(rowSet.getString(1),x-1)
        RowSet.next()
    next x
end sub
```

7.4 Scripting Example 4: ooRexx – Print Data To Console

```
/* initialize connection to server, get XContext */
xContext = UNO.connect()  -- connect to server and retrieve the XContext object
XMcf = xContext~getServiceManager   -- retrieve XMultiComponentFactory

-- first we create our RowSet object and get its XRowSet interface
oRowSet = xMcf~createInstanceWithContext("com.sun.star.sdbc.RowSet", xContext)
xRowSet = oRowSet~XRowSet

-- set the properties needed to connect to a database
xProp = xRowSet~XPropertySet

-- the DataSourceName can be a data source registered,
-- among other possibilities
xProp~setPropertyValue("DataSourceName", "TestDatabase")

-- the CommandType must be TABLE, QUERY or COMMAND - here we use COMMAND
xProp~setPropertyValue("CommandType", ,
                       box("int",bsf.getStaticValue("com.sun.star.sdb.CommandType","COMMAND")))

-- the Command could be a table or query name or a SQL command, depending on
-- the CommandType
mycommand = "SELECT last_name, first_name, email_address FROM members, emails "
mycommand = mycommand"WHERE members.member_id = emails.member_id and edefault = 1 "
mycommand = mycommand"ORDER BY last_name"
xProp~setPropertyValue("Command", mycommand)

xRowSet~execute

-- prepare the XRow interface for column access
xRow = oRowSet~XRow

SAY "Member with default email address"
SAY "-----------------------------------------------"

DO WHILE xRowSet~next > 0
        lastname  = xRow~getString(1)
        firstname = xRow~getString(2)
        email     = xRow~getString(3)
        say lastname"," firstname":" email
END

::requires UNO.CLS    -- get UNO support
```

7.5 Scripting example 5: ooRexx – insert data into OOo writer

```
/* initialize connection to server, get XContext */
xContext = UNO.connect()  -- connect to server and retrieve the XContext object
XMcf = xContext~getServiceManager   -- retrieve XMultiComponentFactory

-- create RowSet object and get its XRowSet interface
oRowSet = xMcf~createInstanceWithContext("com.sun.star.sdbc.RowSet", xContext)
xRowSet = oRowSet~XRowSet
```

```
-- set the properties needed to connect to a database
xProp = xRowSet~XPropertySet

-- the DataSourceName can be a data source registered,
-- among other possibilities
xProp~setPropertyValue("DataSourceName", "TestDatabase")

-- the CommandType must be TABLE, QUERY or COMMAND - here we use COMMAND
xProp~setPropertyValue("CommandType",box("int",bsf.getStaticValue("com.sun.star.sdb.CommandType", ,
                       "COMMAND")))

-- the Command could be a table or query name or a SQL command, depending on
-- the CommandType
mycommand = "SELECT last_name, first_name, email_address FROM members, emails "
mycommand = mycommand"WHERE members.member_id = emails.member_id and edefault = 1 "
mycommand = mycommand"ORDER BY last_name"
xProp~setPropertyValue("Command",mycommand)

xRowSet~execute

-- prepare the XRow interface for column access
xRow = oRowSet~XRow


-- Retrieve the Desktop object, we need its XComponentLoader interface to load
-- a new document
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader
                                                       -- interface
/* open the blank *.sxw - file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

/* create the TextObject */
xWriterDocument = xWriterComponent~XTextDocument
xText = xWriterDocument~getText

/* define line feed */
cr="13"~d2c

/* output to writer */
DO WHILE xRowSet~next > 0
        lastname  = xRow~getString(1)
        firstname = xRow~getString(2)
        email     = xRow~getString(3)
        xText~getEnd~setString(lastname"," firstname":" email)
        xText~getEnd~setString(cr);
END

::requires UNO.CLS    -- get UNO support
```

7.6 Scripting Example 6: ooRexx - Display Data In OpenOffice.org Calc With Diagram

```
/* initialize connection to server, get XContext */
xContext = UNO.connect()  -- connect to server and retrieve the XContext object
XMcf = xContext~getServiceManager   -- retrieve XMultiComponentFactory

-- create RowSet object and get its XRowSet interface
oRowSet = xMcf~createInstanceWithContext("com.sun.star.sdbc.RowSet", xContext)
xRowSet = oRowSet~XRowSet

-- set the properties needed to connect to a database
xProp = xRowSet~XPropertySet

-- the DataSourceName can be a data source registered,
```

```
-- among other possibilities
xProp~setPropertyValue("DataSourceName", "TestDatabase")

-- the CommandType must be TABLE, QUERY or COMMAND - here we use COMMAND
xProp~setPropertyValue("CommandType", box("int", bsf.getStaticValue("com.sun.star.sdb.CommandType",
"COMMAND")))

-- the Command could be a table or query name or a SQL command, depending on
-- the CommandType
xProp~setPropertyValue("Command", "SELECT kam_long from kams_query order by kam_long")

xRowSet~execute

-- prepare the XRow interface for column access
xRow = oRowSet~XRow

numkam = 0
/* output to console */
say
say "Please, choose one key account manager and type in it's number below:"
say

DO WHILE xRowSet~next > 0
   numkam = numkam + 1
        kamname.numkam  = xRow~getString(1)
        say numkam": "kamname.numkam
END

say
say "Type in number of key account manager please:"
parse pull selkam .

say "Your choice: "kamname.selkam
say


-- the Command could be a table or query name or a SQL command, depending on
-- the CommandType
xProp~setPropertyValue("Command", "SELECT distinct ryear from revenues order by ryear")

xRowSet~execute

-- prepare the XRow interface for column access
xRow = oRowSet~XRow

/* output to console */
say
say "Please, choose a year:"
say


DO WHILE xRowSet~next > 0
        kamyear  = xRow~getString(1)
        say kamyear
END

say
say "Type in required year:"
parse pull selyear .

say "Your choice: "selyear
say

-- Retrieve the Desktop object, we need its XComponentLoader interface to load
-- a new document
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader
                                                       -- interface
/* open the blank *.sxw - file */
url = "private:factory/scalc"
xCalcComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

/* get first sheet in spreadsheet */
xSheet = xCalcComponent~XSpreadSheetDocument~getSheets~XIndexAccess~getByIndex(0) ~XSpreadSheet
```

```
/* insert some text */
CALL UNO.setCell xSheet, 0, 0, "sales agent"
CALL UNO.setCell xSheet, 1, 0, kamname.selkam
CALL UNO.setCell xSheet, 0, 1, "year"
CALL UNO.setCell xSheet, 1, 1, selyear
CALL UNO.setCell xSheet, 0, 3, "model"
CALL UNO.setCell xSheet, 1, 3, "revenue"

/* get the cell style container */
xFamiliesSupplier = xCalcComponent~XSpreadSheetDocument~XStyleFamiliesSupplier
xCellStyle = xFamiliesSupplier~getStyleFamilies~getByName("CellStyles")~XNameContainer

/* create a new cell style */
xServiceManager = xCalcComponent~XSpreadSheetDocument~XMultiServiceFactory
oCellStyle = xServiceManager~createInstance("com.sun.star.style.CellStyle")
xCellStyle~insertByName("MyNewCellStyle", oCellStyle)

/* modify properties of the new style */
xPropertySet = oCellStyle~XPropertySet
xPropertySet~setPropertyValue("CharWeight", box("float", 140))

/* create a CellRange and set the PropertyStyle */
xCellRange = xSheet~getCellRangeByPosition(0, 0, 0, 1)
xCellRange~XPropertySet~setPropertyValue("CellStyle", "MyNewCellStyle")

xCellRange = xSheet~getCellRangeByPosition(0, 3, 1, 3)
xCellRange~XPropertySet~setPropertyValue("CellStyle", "MyNewCellStyle")

sqlcommand = "SELECT model, revenue FROM revenues, kams, models WHERE ryear="selyear
sqlcommand = sqlcommand " AND concat(concat(kam_lname,', '),kam_fname)='"kamname.selkam"'"
sqlcommand = sqlcommand " and revenues.kam_id = kams.kam_id"
sqlcommand = sqlcommand " and revenues.model_id = models.model_id order by model"

-- the Command could be a table or query name or a SQL command, depending on
-- the CommandType
xProp~setPropertyValue("Command", sqlcommand)

xRowSet~execute

-- prepare the XRow interface for column access
xRow = oRowSet~XRow

/* output to calc */
countrow = 0
DO WHILE xRowSet~next > 0
   countrow = countrow + 1
   CALL UNO.setCell xSheet, 0, countrow+3, xRow~getString(1)
   CALL UNO.setCell xSheet, 1, countrow+3, xRow~getString(2)
END

/* create the frame for the Chart */
oRect = .bsf~new("com.sun.star.awt.Rectangle")
oRect~X      = 8000
oRect~Y      = 1000
oRect~Width  = 10000
oRect~Height = 7000

/* catch the underlying data and make a CellRange*/
myRange=xSheet~XCellRange ~getCellRangeByName("A4:B7")
myAddr = myRange~XCellRangeAddressable~getRangeAddress

/* create the CellRangeAddress for the Chart */
CALL UNO.loadClass "com.sun.star.table.CellRangeAddress"
oAddr = bsf.createArray(.UNO~CellRangeAddress, 1)     -- create Java array
oAddr[1] = myAddr

/* get the Sheet's ChartsSupplier, add a new Chart */
xTableCharts = xSheet~XTableChartsSupplier~getCharts
xTableCharts~addNewByName("revenues", oRect, oAddr, .true, .true)

xChart = xTableCharts~getByName("revenues")

/* get the embedded Object and set properties*/
```

```
xComponent = xChart~XEmbeddedObjectSupplier~getEmbeddedObject
xChartDocument = xComponent~XChartDocument
xMsf = xChartDocument~XMultiServiceFactory
xDiagram = xMsf~createInstance("com.sun.star.chart.PieDiagram")~xDiagram
xChartDocument~setDiagram(xDiagram)

xComponent~XPropertySet~setPropertyValue("HasMainTitle", box("bool", .true))
xChartDocument~getTitle~XPropertySet~setPropertyValue("String", ,
               box("string", "Revenues "selyear" for "kamname.selkam))


::requires UNO.CLS     -- get UNO support
```

7.7 Scripting Example 7: Java – Print Data To Console

```
import com.sun.star.bridge.XUnoUrlResolver;
import com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XComponentContext;
import com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.beans.XPropertySet;

public class example7 {

    private XComponentContext xContext = null;
    private XMultiComponentFactory xMCF = null;

    /** Creates a new instance of example7 */
    public example7() {
    }

    /* main */
    public static void main(String[] args) {
        example7 example7_1 = new example7();
        try {
            example7_1.example7();
        }
        catch (java.lang.Exception e){
            e.printStackTrace();
        }
        finally {
            System.exit(0);
        }
    }

    protected void example7() throws com.sun.star.uno.Exception, java.lang.Exception {
        try {
            // get the remote office component context
            xContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
            System.out.println("Connected to a running office ...");
            xMCF = xContext.getServiceManager();
        }
        catch( Exception e) {
            System.err.println("ERROR: can't get a component context from a running office ...");
            e.printStackTrace();
            System.exit(1);
        }

        // first we create our RowSet object and get its XRowSet interface
        Object rowSet = xMCF.createInstanceWithContext(
            "com.sun.star.sdb.RowSet", xContext);

        com.sun.star.sdbc.XRowSet xRowSet = (com.sun.star.sdbc.XRowSet)
            UnoRuntime.queryInterface(com.sun.star.sdbc.XRowSet.class, rowSet);

        // set the properties needed to connect to a database
        XPropertySet xProp = (XPropertySet)UnoRuntime.queryInterface(XPropertySet.class, xRowSet);

        // the DataSourceName can be a data source registered with [PRODUCTNAME], among other
        //  possibilities
        xProp.setPropertyValue("DataSourceName","TestDatabase");
```

```
        // the CommandType must be TABLE, QUERY or COMMAND, here we use COMMAND
        xProp.setPropertyValue("CommandType",new Integer(com.sun.star.sdb.CommandType.COMMAND));

        // the Command could be a table or query name or a SQL command, depending on the CommandType
String mycommand = "SELECT member_id, last_name, first_name, email_address FROM members, emails ";
mycommand += "WHERE members.member_id = emails.member_id and edefault = 1 ORDER BY last_name";
xProp.setPropertyValue("Command",mycommand);

        xRowSet.execute();

        // prepare the XRow and XColumnLocate interface for column access
        // XRow gets column values
        com.sun.star.sdbc.XRow xRow = (com.sun.star.sdbc.XRow)UnoRuntime.queryInterface(
            com.sun.star.sdbc.XRow.class, xRowSet);
        // XColumnLocate finds columns by name
        com.sun.star.sdbc.XColumnLocate xLoc = (com.sun.star.sdbc.XColumnLocate)
            UnoRuntime.queryInterface(
                com.sun.star.sdbc.XColumnLocate.class, xRowSet);

        // print output header
        System.out.println("Member ID\tName & Email Address");
        System.out.println("----------\t--------------------");

        // output result rows
        while ( xRowSet != null && xRowSet.next() ) {
            String memberid = xRow.getString(xLoc.findColumn("member_id"));
            String lname = xRow.getString(xLoc.findColumn("last_name"));
            String fname = xRow.getString(xLoc.findColumn("first_name"));
            String email = xRow.getString(xLoc.findColumn("email_address"));
            System.out.println(memberid + "\t\t" + lname + ", " + fname + ": " + email);
        }
    }
}
```

7.8 Scripting Example 8: Java - Insert Data Into OOo Writer

```
import com.sun.star.bridge.XUnoUrlResolver;
import com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XComponentContext;
import com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.beans.XPropertySet;

public class example8 {

    private XComponentContext xContext = null;
    private XMultiComponentFactory xMCF = null;
    com.sun.star.frame.XDesktop xDesktop = null;

    /** Creates a new instance of example8 */
    public example8() {
    }

    /* main */
    public static void main(String[] args) {
        example8 example8_1 = new example8();
        try {
            example8_1.example8();
        }
        catch (java.lang.Exception e){
            e.printStackTrace();
        }
        finally {
            System.exit(0);
        }
    }

    protected void example8() throws com.sun.star.uno.Exception, java.lang.Exception {
        try {
            // get the remote office component context
```

```
            xContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
            System.out.println("Connected to a running office ...");
            xMCF = xContext.getServiceManager();
        }
        catch( Exception e) {
            System.err.println("ERROR: can't get a component context from a running office ...");
            e.printStackTrace();
            System.exit(1);
        }

        // first we create our RowSet object and get its XRowSet interface
        Object rowSet = xMCF.createInstanceWithContext(
            "com.sun.star.sdb.RowSet", xContext);

        com.sun.star.sdbc.XRowSet xRowSet = (com.sun.star.sdbc.XRowSet)
            UnoRuntime.queryInterface(com.sun.star.sdbc.XRowSet.class, rowSet);

        // set the properties needed to connect to a database
        XPropertySet xProp = (XPropertySet)UnoRuntime.queryInterface(XPropertySet.class, xRowSet);

// the DataSourceName can be a data source registered with [PRODUCTNAME], among other possibilities
        xProp.setPropertyValue("DataSourceName","TestDatabase");

        // the CommandType must be TABLE, QUERY or COMMAND, here we use COMMAND
        xProp.setPropertyValue("CommandType",new Integer(com.sun.star.sdb.CommandType.COMMAND));

// the Command could be a table or query name or a SQL command, depending on the CommandType
String mycommand = "SELECT member_id, last_name, first_name, email_address FROM members, emails ";
mycommand += "WHERE members.member_id = emails.member_id and edefault = 1 ORDER BY last_name";
xProp.setPropertyValue("Command",mycommand);

        xRowSet.execute();

        // prepare the XRow and XColumnLocate interface for column access
        // XRow gets column values
        com.sun.star.sdbc.XRow xRow = (com.sun.star.sdbc.XRow)UnoRuntime.queryInterface(
            com.sun.star.sdbc.XRow.class, xRowSet);
        // XColumnLocate finds columns by name
        com.sun.star.sdbc.XColumnLocate xLoc = (com.sun.star.sdbc.XColumnLocate)
            UnoRuntime.queryInterface(
                com.sun.star.sdbc.XColumnLocate.class, xRowSet);

                        //start up an instance of office
         Object oDesktop = xMCF.createInstanceWithContext(
         "com.sun.star.frame.Desktop", xContext);

         //get the XDesktop interface object
         xDesktop = (com.sun.star.frame.XDesktop)
         com.sun.star.uno.UnoRuntime.queryInterface(
         com.sun.star.frame.XDesktop.class, oDesktop);

         //get the desktop's component loader interface object
         com.sun.star.frame.XComponentLoader xComponentLoader =
             (com.sun.star.frame.XComponentLoader)
         com.sun.star.uno.UnoRuntime.queryInterface(
             com.sun.star.frame.XComponentLoader.class, xDesktop);

         //create an empty text ("swriter") document
         com.sun.star.beans.PropertyValue xEmptyArgs[] = // empty property array
             new com.sun.star.beans.PropertyValue[0];

         //create an empty word processor ("swriter") component (document)
         com.sun.star.lang.XComponent xComponent = // text document
         xComponentLoader.loadComponentFromURL( "private:factory/swriter","_blank",0, xEmptyArgs);

         //get Text interface object and set a text
         com.sun.star.text.XTextDocument xTextDocument =
             (com.sun.star.text.XTextDocument)
             com.sun.star.uno.UnoRuntime.queryInterface(
             com.sun.star.text.XTextDocument.class, xComponent);

         com.sun.star.text.XText xText = xTextDocument.getText();

                        String output = "";
```

```
        // print output header
        output += "Member ID\tName & Email Address\r";
        output += "----------\t--------------------\r";

        // output result rows
        while ( xRowSet != null && xRowSet.next() ) {
            String memberid = xRow.getString(xLoc.findColumn("member_id"));
            String lname = xRow.getString(xLoc.findColumn("last_name"));
            String fname = xRow.getString(xLoc.findColumn("first_name"));
            String email = xRow.getString(xLoc.findColumn("email_address"));
            output += memberid + "\t\t" + lname + ", " + fname + ": " + email + "\r";
        }
        //write to writer document
        xText.setString(output);
    }
}
```

7.9 Scripting Example 9 : Java - Display Data In OpenOffice.org Calc With Diagram

```
import com.sun.star.awt.Rectangle;
import com.sun.star.beans.PropertyValue;
import com.sun.star.beans.XPropertySet;
import com.sun.star.chart.XDiagram;
import com.sun.star.chart.XChartDocument;
import com.sun.star.container.XIndexAccess;
import com.sun.star.container.XNameAccess;
import com.sun.star.container.XNameContainer;
import com.sun.star.document.XEmbeddedObjectSupplier;
import com.sun.star.frame.XDesktop;
import com.sun.star.frame.XComponentLoader;
import com.sun.star.lang.XComponent;
import com.sun.star.lang.XMultiServiceFactory;
import com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XInterface;
import com.sun.star.uno.XComponentContext;
import com.sun.star.sheet.XCellRangeAddressable;
import com.sun.star.sheet.XSpreadsheet;
import com.sun.star.sheet.XSpreadsheets;
import com.sun.star.sheet.XSpreadsheetDocument;
import com.sun.star.style.XStyleFamiliesSupplier;
import com.sun.star.table.CellRangeAddress;
import com.sun.star.table.XCell;
import com.sun.star.table.XCellRange;
import com.sun.star.table.XTableChart;
import com.sun.star.table.XTableCharts;
import com.sun.star.table.XTableChartsSupplier;

public class example9 {

    private XComponentContext xContext = null;
    private XMultiComponentFactory xMCF = null;
    com.sun.star.frame.XDesktop xDesktop = null;

    /** Creates a new instance of example9 */
    public example9() {
    }

    /* main */
    public static void main(String[] args) {
        example9 example9_1 = new example9();
        try {
                                if (args.length < 2)
                                {
                        example9_1.noargs();
                }
```

```
                else
                {
                        example9_1.withargs(args[0], args[1]);
                }
        }
        catch (java.lang.Exception e){
            e.printStackTrace();
        }
        finally {
            System.exit(0);
        }
    }

    protected void noargs() throws com.sun.star.uno.Exception,
    java.lang.Exception { try {
                        System.out.println();
                        System.out.println("Please, invoke this program with 2 parameters.");
                        System.out.println();
                        System.out.println("Parameter 1: ID of key account manager");
                        System.out.println("Parameter 2: Year");
                        System.out.println();
                        System.out.println("Choose from the following list:");
                        System.out.println();

            // get the remote office component context
            xContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
            xMCF = xContext.getServiceManager();
        }
        catch( Exception e) {
            System.err.println("ERROR: can't get a component context from a running office ...");
            e.printStackTrace();
            System.exit(1);
        }

        // first we create our RowSet object and get its XRowSet interface
        Object rowSet = xMCF.createInstanceWithContext(
            "com.sun.star.sdb.RowSet", xContext);

        com.sun.star.sdbc.XRowSet xRowSet = (com.sun.star.sdbc.XRowSet)
            UnoRuntime.queryInterface(com.sun.star.sdbc.XRowSet.class, rowSet);

        // set the properties needed to connect to a database
        XPropertySet xProp = (XPropertySet)UnoRuntime.queryInterface(XPropertySet.class, xRowSet);

        // the DataSourceName can be a data source registered with [PRODUCTNAME], among other
       // possibilities
        xProp.setPropertyValue("DataSourceName","TestDatabase");

        // the CommandType must be TABLE, QUERY or COMMAND, here we use COMMAND
        xProp.setPropertyValue("CommandType",new Integer(com.sun.star.sdb.CommandType.COMMAND));

        // the Command could be a table or query name or a SQL command, depending on the CommandType
xProp.setPropertyValue("Command","SELECT kam_lname,kam_fname, kam_id FROM kams order by kam_lname");

        xRowSet.execute();

        // prepare the XRow and XColumnLocate interface for column access
        // XRow gets column values
        com.sun.star.sdbc.XRow xRow = (com.sun.star.sdbc.XRow)UnoRuntime.queryInterface(
            com.sun.star.sdbc.XRow.class, xRowSet);
        // XColumnLocate finds columns by name
        com.sun.star.sdbc.XColumnLocate xLoc = (com.sun.star.sdbc.XColumnLocate)
            UnoRuntime.queryInterface(
                com.sun.star.sdbc.XColumnLocate.class, xRowSet);

        // print output header
        System.out.println("KAM name and ID");
        System.out.println("----------------------");

        // output result rows
        while ( xRowSet != null && xRowSet.next() ) {
            String kamid = xRow.getString(xLoc.findColumn("kam_id"));
            String lname = xRow.getString(xLoc.findColumn("kam_lname"));
            String fname = xRow.getString(xLoc.findColumn("kam_fname"));
```

```
            System.out.println(lname + ", " + fname + ": " + kamid);
        }
        System.out.println();

        // the Command could be a table or query name or a SQL command, depending on the CommandType
        xProp.setPropertyValue("Command","SELECT distinct ryear FROM revenues order by ryear");

        xRowSet.execute();

        // print output header
        System.out.println("Year");
        System.out.println("-----");

        // output result rows
        while ( xRowSet != null && xRowSet.next() ) {
            String ryear = xRow.getString(xLoc.findColumn("ryear"));
            System.out.println(ryear);
        }
        System.out.println();
    }

    protected void withargs(String kamid, String year) throws com.sun.star.uno.Exception,
java.lang.Exception {
        try {
                // get the remote office component context
         xContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
         xMCF = xContext.getServiceManager();
        }
   catch( Exception e) {
            System.err.println("ERROR: can't get a component context from a running office ...");
            e.printStackTrace();
            System.exit(1);
        }

                // first we create our RowSet object and get its XRowSet interface
        Object rowSet = xMCF.createInstanceWithContext(
            "com.sun.star.sdb.RowSet", xContext);

                  //start up an instance of office
         Object oDesktop = xMCF.createInstanceWithContext(
         "com.sun.star.frame.Desktop", xContext);

         //get the XDesktop interface object
         xDesktop = (com.sun.star.frame.XDesktop)
         com.sun.star.uno.UnoRuntime.queryInterface(
         com.sun.star.frame.XDesktop.class, oDesktop);

         //get the desktop's component loader interface object
         com.sun.star.frame.XComponentLoader xComponentLoader =
             (com.sun.star.frame.XComponentLoader)
         com.sun.star.uno.UnoRuntime.queryInterface(
             com.sun.star.frame.XComponentLoader.class, xDesktop);

         //create an empty calc document
         com.sun.star.beans.PropertyValue xEmptyArgs[] = // empty property array
             new com.sun.star.beans.PropertyValue[0];

         //create an empty calc component (document)
         com.sun.star.lang.XComponent xComponent = // calc document
             xComponentLoader.loadComponentFromURL( "private:factory/scalc","_blank",
                       0, xEmptyArgs);

         XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
UnoRuntime.queryInterface(XSpreadsheetDocument.class, xComponent);

         XSpreadsheets xSheets = xSpreadsheetDocument.getSheets() ;
            XIndexAccess oIndexSheets = (XIndexAccess) UnoRuntime.queryInterface(
                XIndexAccess.class, xSheets);
         XSpreadsheet xSheet = (XSpreadsheet) UnoRuntime.queryInterface(
                XSpreadsheet.class, oIndexSheets.getByIndex(0));

                  insertIntoCell(0,0,"sales agent",xSheet,"");
                  //insertIntoCell(1,0,"kam name",xSheet,"");
                  insertIntoCell(0,1,"year",xSheet,"");
```

```
            insertIntoCell(1,1,year,xSheet,"");
            insertIntoCell(0,3,"model",xSheet,"");
            insertIntoCell(1,3,"revenue",xSheet,"");

            //Setting cell style
            XStyleFamiliesSupplier xSFS = (XStyleFamiliesSupplier)
          UnoRuntime.queryInterface(XStyleFamiliesSupplier.class, xSpreadsheetDocument);
      XNameAccess xSF = (XNameAccess) xSFS.getStyleFamilies();
      XNameAccess xCS = (XNameAccess) UnoRuntime.queryInterface(
          XNameAccess.class, xSF.getByName("CellStyles"));
      XMultiServiceFactory oDocMSF = (XMultiServiceFactory)
          UnoRuntime.queryInterface(XMultiServiceFactory.class, xSpreadsheetDocument );
      XNameContainer oStyleFamilyNameContainer = (XNameContainer)
          UnoRuntime.queryInterface(
          XNameContainer.class, xCS);
      XInterface oInt2 = (XInterface) oDocMSF.createInstance(
          "com.sun.star.style.CellStyle");
      oStyleFamilyNameContainer.insertByName("MyNewCellStyle", oInt2);
      XPropertySet oCPS = (XPropertySet)UnoRuntime.queryInterface(
          XPropertySet.class, oInt2 );
      oCPS.setPropertyValue("CharWeight", new Float(140));

            XCellRange xCR = null;
            xCR = xSheet.getCellRangeByPosition(0,0,0,1);
            XPropertySet xCPS = (XPropertySet)UnoRuntime.queryInterface(
      XPropertySet.class, xCR );
  xCPS.setPropertyValue("CellStyle", "MyNewCellStyle");
            xCR = xSheet.getCellRangeByPosition(0,3,1,3);
            xCPS = (XPropertySet)UnoRuntime.queryInterface(
      XPropertySet.class, xCR );
  xCPS.setPropertyValue("CellStyle", "MyNewCellStyle");

            //preparing rowset
  com.sun.star.sdbc.XRowSet xRowSet = (com.sun.star.sdbc.XRowSet)
      UnoRuntime.queryInterface(com.sun.star.sdbc.XRowSet.class, rowSet);

  // set the properties needed to connect to a database
  XPropertySet xProp = (XPropertySet)UnoRuntime.queryInterface(XPropertySet.class, xRowSet);

  // the DataSourceName can be a data source registered with [PRODUCTNAME], among other
  // possibilities
  xProp.setPropertyValue("DataSourceName","TestDatabase");

  // the CommandType must be TABLE, QUERY or COMMAND, here we use COMMAND
  xProp.setPropertyValue("CommandType",new Integer(com.sun.star.sdb.CommandType.COMMAND));

  // the Command could be a table or query name or a SQL command, depending on the CommandType
  String sqlcommand;
  sqlcommand = "SELECT concat(concat(kam_lname,', '),kam_fname) as kamname, model,"
  sqlcommand += " revenue FROM revenues, models, kams WHERE ryear="+year;
  sqlcommand += " AND kam_id="+kamid+" and revenues.kam_id = kams.kam_id";
  sqlcommand += " and revenues.model_id = models.model_id order by model";
  xProp.setPropertyValue("Command",sqlcommand);

  xRowSet.execute();

  // prepare the XRow and XColumnLocate interface for column access
  // XRow gets column values
  com.sun.star.sdbc.XRow xRow = (com.sun.star.sdbc.XRow)UnoRuntime.queryInterface(
      com.sun.star.sdbc.XRow.class, xRowSet);
  // XColumnLocate finds columns by name
  com.sun.star.sdbc.XColumnLocate xLoc = (com.sun.star.sdbc.XColumnLocate)
      UnoRuntime.queryInterface(
          com.sun.star.sdbc.XColumnLocate.class, xRowSet);

  // output result rows
  int countrow=0;
  String kamname="";
  while ( xRowSet != null && xRowSet.next() ) {
                  if (countrow == 0)
                  {
                          kamname = xRow.getString(xLoc.findColumn("kamname"));

                  }
```

```
                    countrow++;
            String model = xRow.getString(xLoc.findColumn("model"));
            String revenue = xRow.getString(xLoc.findColumn("revenue"));
            insertIntoCell(1,0,kamname,xSheet,"");
            insertIntoCell(0,countrow+3,model,xSheet,"");
            insertIntoCell(1,countrow+3,revenue,xSheet,"");
        }

                //Creating chart
                Rectangle oRect = new Rectangle();
        oRect.X = 8000;
        oRect.Y = 1000;
        oRect.Width = 10000;
        oRect.Height = 7000;

        XCellRange oRange = (XCellRange)UnoRuntime.queryInterface(
            XCellRange.class, xSheet);
        XCellRange myRange = oRange.getCellRangeByName("A4:B7");
        XCellRangeAddressable oRangeAddr = (XCellRangeAddressable)
            UnoRuntime.queryInterface(XCellRangeAddressable.class, myRange);
        CellRangeAddress myAddr = oRangeAddr.getRangeAddress();

        CellRangeAddress[] oAddr = new CellRangeAddress[1];
        oAddr[0] = myAddr;
        XTableChartsSupplier oSupp = (XTableChartsSupplier)UnoRuntime.queryInterface(
            XTableChartsSupplier.class, xSheet);

        XTableChart oChart = null;

        XTableCharts oCharts = oSupp.getCharts();
        oCharts.addNewByName("revenues", oRect, oAddr, true, true);

                //setting chart properties
             oChart = (XTableChart) (UnoRuntime.queryInterface(
                XTableChart.class, ((XNameAccess)UnoRuntime.queryInterface(
                            XNameAccess.class, oCharts)).getByName("revenues")));
            XEmbeddedObjectSupplier oEOS = (XEmbeddedObjectSupplier)
                UnoRuntime.queryInterface(XEmbeddedObjectSupplier.class, oChart);
            XInterface oInt = oEOS.getEmbeddedObject();
            XChartDocument xChart = (XChartDocument) UnoRuntime.queryInterface(
                XChartDocument.class,oInt);
            XPropertySet oCHPS = (XPropertySet)UnoRuntime.queryInterface(
                XPropertySet.class, xChart );
            oCHPS.setPropertyValue("HasMainTitle", new Boolean(true));
            XDiagram oDiag = (XDiagram) xChart.getDiagram();
            XPropertySet oTPS = (XPropertySet)UnoRuntime.queryInterface(
                XPropertySet.class, xChart.getTitle() );
            oTPS.setPropertyValue("String","Revenues "+year+" for "+kamname);

                //changing chart type
        XMultiServiceFactory xMSF = (XMultiServiceFactory)
            UnoRuntime.queryInterface( XMultiServiceFactory.class, xChart );
        Object object = xMSF.createInstance( "com.sun.star.chart.PieDiagram" );
        oDiag = (XDiagram) UnoRuntime.queryInterface(XDiagram.class, object);

        xChart.setDiagram(oDiag);
    }

    //function for setting cell values
    public static void insertIntoCell(int CellX, int CellY, String theValue,
                                      XSpreadsheet TT1, String flag)
     {
        XCell xCell = null;

        try {
            xCell = TT1.getCellByPosition(CellX, CellY);
        } catch (com.sun.star.lang.IndexOutOfBoundsException ex) {
            System.err.println("Could not get Cell");
            ex.printStackTrace(System.err);
        }

        if (flag.equals("V")) {
            xCell.setValue((new Float(theValue)).floatValue());
        } else {
```

```
            xCell.setFormula(theValue);
        }
    }
}
```