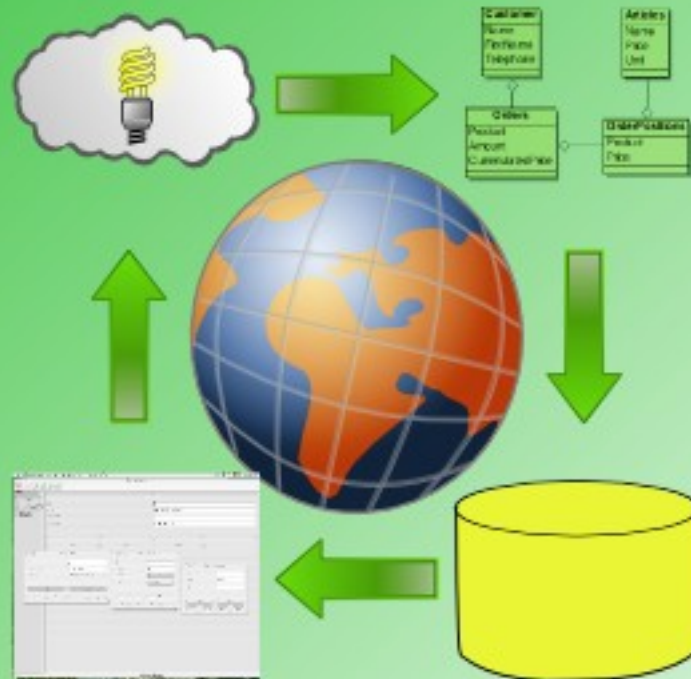


# wxWrapper 1.0 rc 2



From ideas to  
Prototypes in  
minutes. [www.lollisoft.de](http://www.lollisoft.de)

**Fast Application Prototyping**

**driven by wxWrapper and its IbDMF  
framework in version 1.0rc2**

Fast Application Prototyping

Documentation

Creating a database prototype in minutes

© 2000 - 2007 Lothar Behrens

\$Revision: 1.5 \$

## Table of contents

Introduction .....	5
Concept.....	5
Quickstart .....	6
Creating an UML model.....	6
Prepare UML usage.....	7
New datatypes.....	7
New stereotypes.....	8
Creating the first UML classes.....	9
Add the customer class to the model.....	9
Add the address class, aggregate to customer .....	10
Add the contacts class, aggregate to customer .....	12
Exporting your UML model as XMI file.....	13
Importing the UML model via XMI file.....	14
Step 1. Creating the application database.....	14
Step 2. Creating the application model definition .....	15
Deleting an application definition .....	16
Installing the database.....	17
Installing on Windows.....	17
Installing on Linux .....	22
Installing on Mac OS X.....	24
Installing on Solaris.....	27
Configuring the database at first time .....	27
Configuring on Windows.....	28
Configuring on Linux.....	33
Configuring on Mac OS X.....	34
Configuring on Solaris.....	34
Setup ODBC configuration .....	35
Setup ODBC on Windows.....	35
Setup ODBC on Linux .....	36
Setup ODBC on Mac OS X.....	37
Setup ODBC on Solaris.....	39
Installing the client application .....	40
Installing from source distribution .....	40
Preparing to build the source distribution .....	40
Prepare on Windows.....	40
Install wxWidgets package.....	40
Prepare on Linux .....	40
Building wxWidgets on Linux from source TGZ.....	41
Prepare on Mac OS X.....	42
Prepare on Solaris.....	42
Installing SRPM distribution .....	42
Installing TGZ distribution .....	42
Installing from Setup executable.....	43
Build the source.....	43
Build from source RPM file.....	44
Build from TGZ file.....	44
Installing a binary package.....	44

Installing on Windows.....	45
Installing on Linux.....	48
Installing on Mac OS X.....	49
Installing on Solaris.....	49

# Introduction

Writing an application is not easy. Especially creating database applications, such as CRM systems or simple CD cataloging software.

wxWrapper enables you to load various application modules defined in a configuration database.

One predefined configuration is 'lbDMF Manager' that enables you to define other database applications. This definition is really based on a dynamic application module who uses further configuration data defined for a specific application. In this case 'lbDMF Manager'.

The DynamicApp module enables you to access to databases by a definition stored in a configuration database.

With this application module it would be a simple and less time consuming work until you have a first version of your database application. You don't need to write one line of code. All you have to do, is defining what data you would make available in the application.

A database application consists of various database forms, that will provide you with different views on to data. For all these views you will define one form definition per view.

## Concept

Without any extra tool (UML would be such an extra tool) you may manually define your application. A simple CD collection database would be shown in the API documentation that is available online or in the Documentation package of the [lbDMF](#) project.

The API documentation can be found [here](#).

Further in this document I like to introduce you in using an [UML](#) modelling tool to get the maximum speed in developing a database prototype. Any other modelling tool may be usable when XML export would be possible and a suitable import template exists.

So the main concept is UML modelling, export, import and try out.

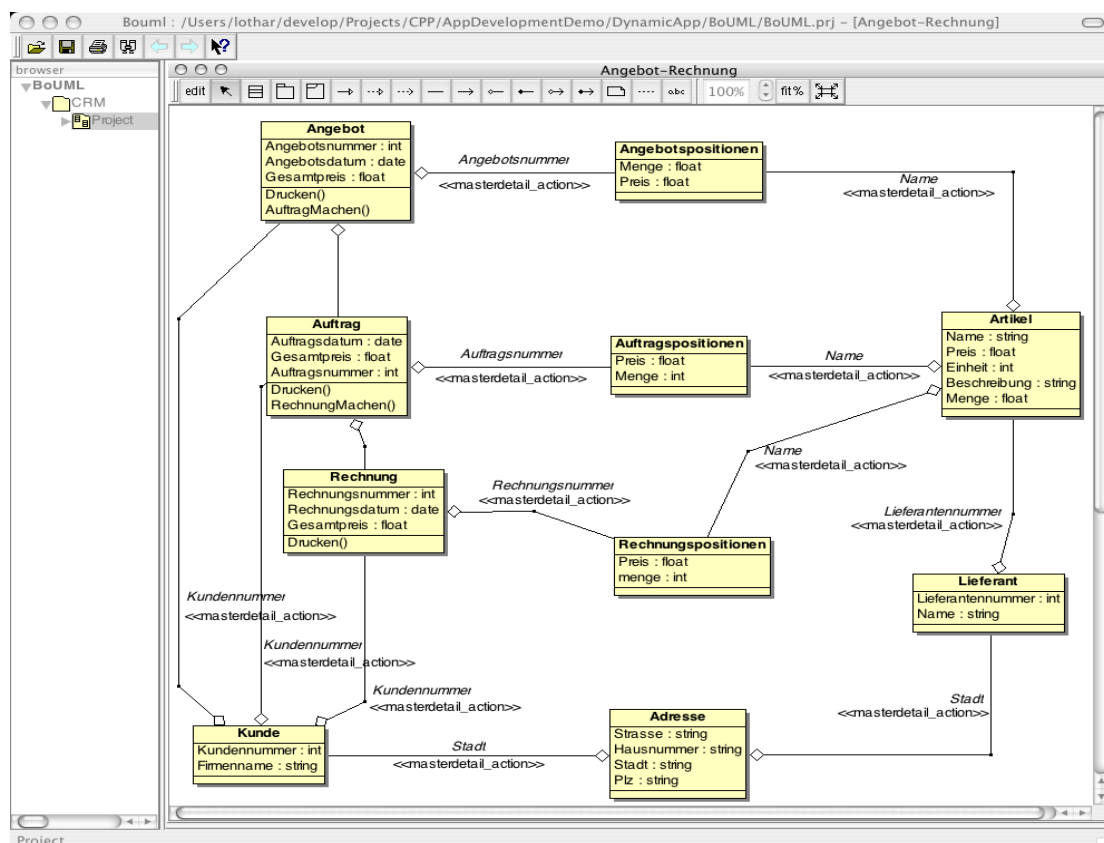
# Quickstart

In the quickstart chapter I will give you a short overview on using the tools to show you what is possible and what is not yet possible.

## Creating an UML model

To create an UML model, you will need to open BoUML. You then go to the Project menu and click on 'New'. You will be asked for a file name. Enter CRM for that. The first entry in the left tree would be the package name 'CRM' and this is important. It would be the application name in the Prototype.

The UML model from the project will look like picture 1.



It is a small CRM system you would be able to enter customers, articles, addresses, contacts, invoices and the like.

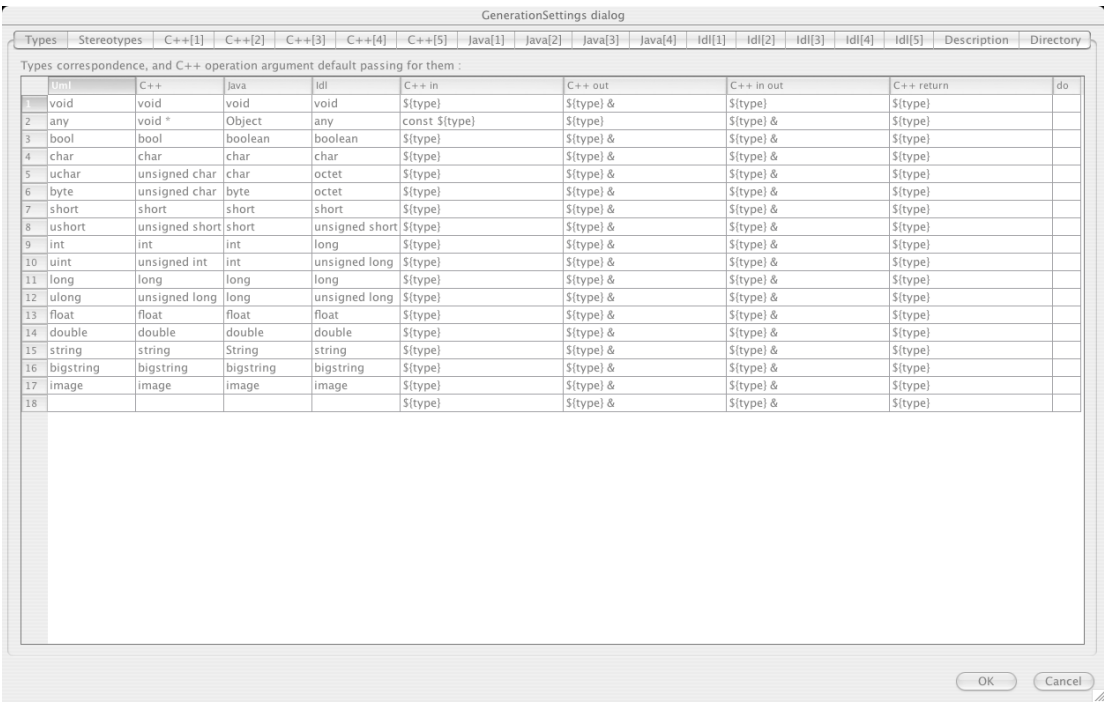
# Prepare UML usage

Before you enter any UML classes, you need to setup some stereotypes and datatypes. These types are explained later.

## New datatypes

Go to the toplevel entry in the browser pane on the left. Open the Project menu, in there open Edit- >Edit generation settings.

You will see this screen:



The following types have been added in comparsion to a new UML project:

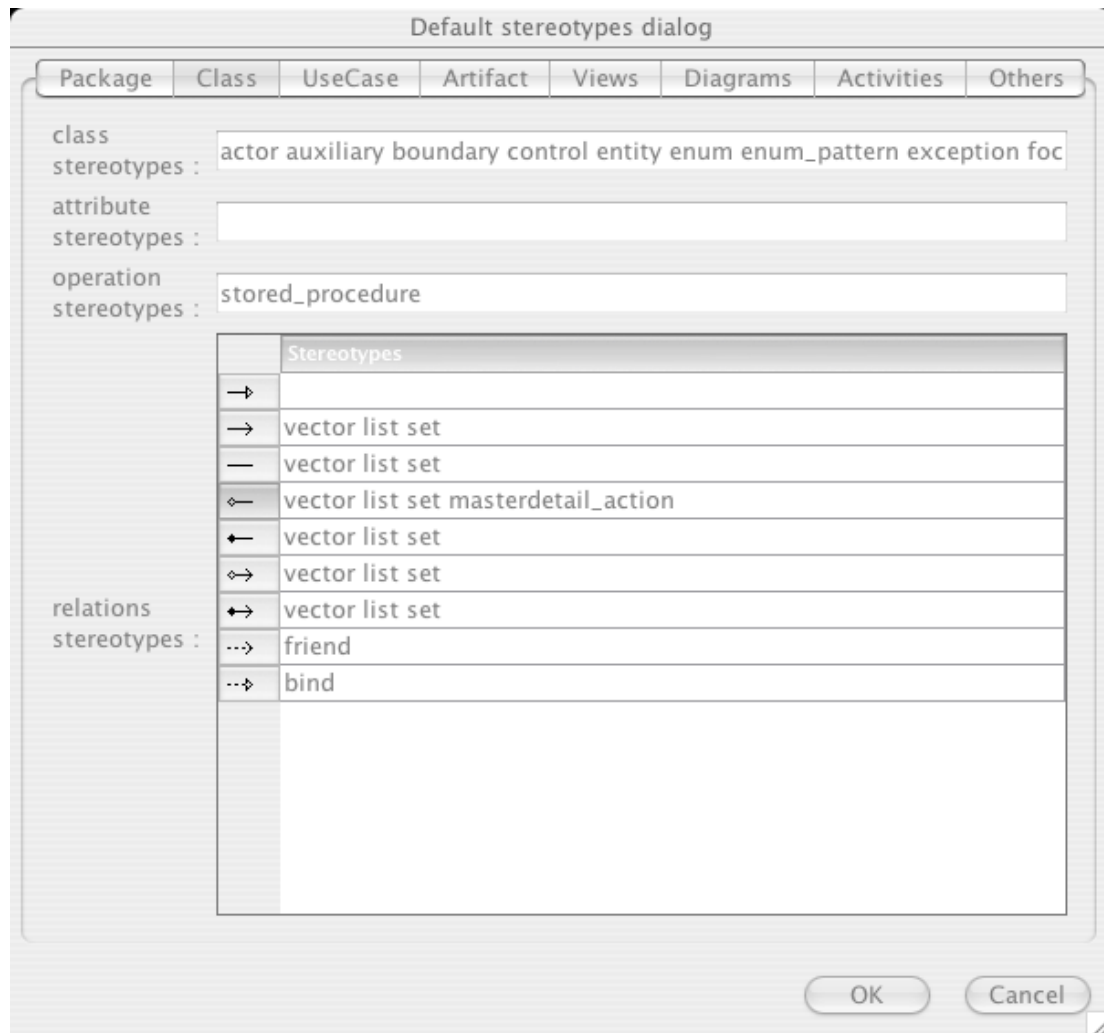
Type bigstring: The bigstring datatype is used to store large text into a database table. Use this for your memofields.

Type image: The image datatype is used to store various image formats in a database table.

## New stereotypes

Go to the toplevel entry in the browser pane on the left. Open the Project menu, in there open Edit- >Edit default stereotypes.

You will see this screen (Picture 3):



The following stereotypes have been added in comparsion to a new UML project:

Stereotype masterdetail\_action:

The masterdetail\_action is used to indicate that an action button will be used in the master form (Adresse is a master) to open a detail form.



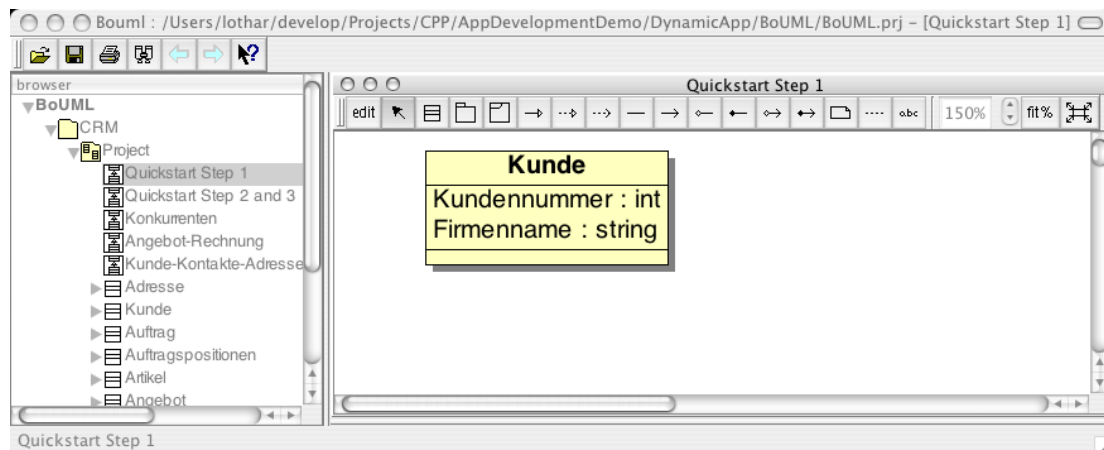
## Creating the first UML classes

We will begin the UML model for our small CRM system with the customer class (Kunde). This may be the first class in mind for a CRM.

To show this, I will create separate class view, because I still have the model. You could use the main project's classview (Project). It takes no matter how the class view name looks like. I do not document the complete UML diagram because you will get it from the [project](#).

## Add the customer class to the model

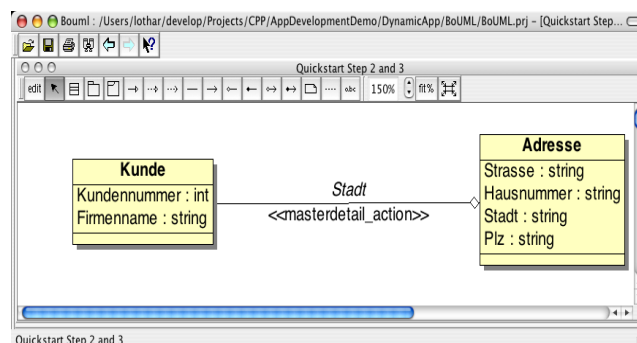
The customer class looks like this (Picture 4):



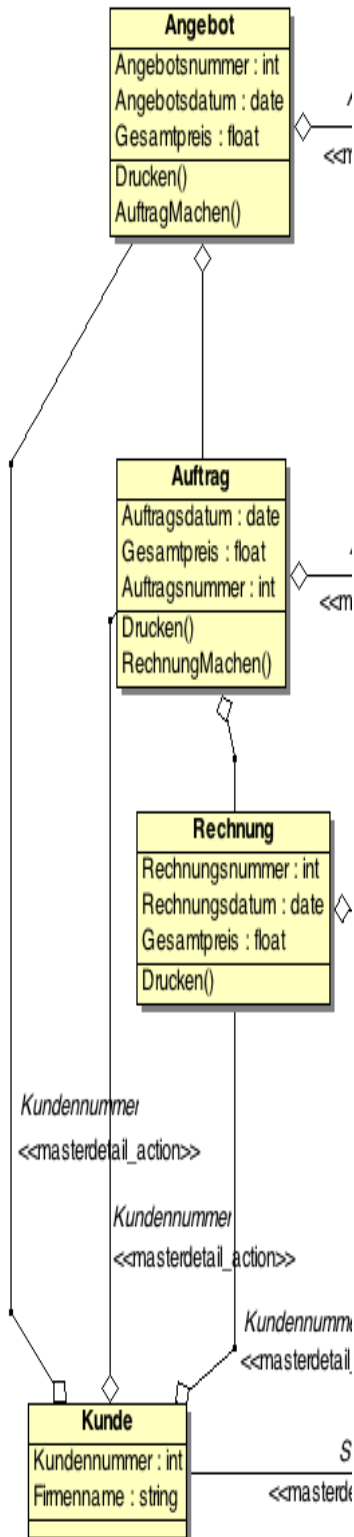
You see, that I do not add any address related data to this class. This is because an employee may also located at any address. Therefore we will add a separate class for the address (Adresse).

## Add the address class, aggregate to customer

The address and an aggregate that links the customer to the address (Picture 5):



The new class Adresse is linked to the customer class with an aggregate. This implies that the customer will have exactly one address, but the address may have multiple customers. **Note: The direction of the aggregate has been changed.**



Now let us add some more classes and link them correctly to the customer class. For our CRM system we need an offering, the order and the invoice class.

The offer is for taking place an offer to the customer. Link them to the customer as shown.

To be able to open the offers per customer, you need to specify the stereotype **'masterdetail\_action'**. With this information, the application adds a button to the prototype application.

Do the same with the remaining classes in this sample.

There are some methods in the classes that let you do more things on that class. The functions are dummies for now, but with the definition of clear stereotypes it would be possible to create actions on the form that – say – do call some stored procedures that will do the work. In the case of CreateOrder: AuftragMachen().

Another sample would be the print functions: Drucken(). Here another action could trigger predefined reports to produce some stuff on paper.

As of the fact not all things are defined, you may do it for your need as long the application has the ability to do what you like to to. If not extend it.

At least, add the contacts class and link them properly to the customer. Create a link from customer to contact.

After you have defined the UML model you will be able to import the definition to be prototyped. The import process would be handled in the a sepatate chapter. Here you will see a preview of what would be possible with your model at this stage.

The customer mask now looks like here (picture 8):

The screenshot shows a software application window titled "Dynamic sample". The window has a standard macOS-style title bar with red, yellow, and green window control buttons. Below the title bar is a toolbar with a red square button containing a white 'i', a wrench and screwdriver icon, and a row of ten star icons. The main content area is titled "Kunde" in a blue button. It contains four input fields: "Kundennummer" with the value "100000", "Firmenname" with the value "Lollisoft Inc", "Adresse" with the value "Lenningen", and "Kontakte" which is empty. Below these fields are three buttons: "Rechnung", "Auftrag", and "Angebot". Underneath these are three more buttons: "Add", "Delete", and "Refresh". At the bottom of the main area are four buttons: "First", "Prev", "Next", and "Last". The status bar at the bottom of the window shows "Ready" on the left and "Load database configuration done." on the right.

Currently I could only use aggregates for an application prototype. This is because the way I import the UML model described later. As noted above, you can extend it for your needs.

Here I will explain the details on aggregations. On the given sample you will see that the aggregate has an associated stereotype of masterdetail\_action. This is important for the later navigation between the database forms. As modelled, you can open the offerings, orders and invoices per customer (Angebot, Auftrag and Rechnung).

The bad side on this model is, that you do not see the address of a customer at the same time because I do not support sub panels to show all the details of the selected address. Also there is not automatically an address tuple for this customer. It may be empty for now. But you can extent it.

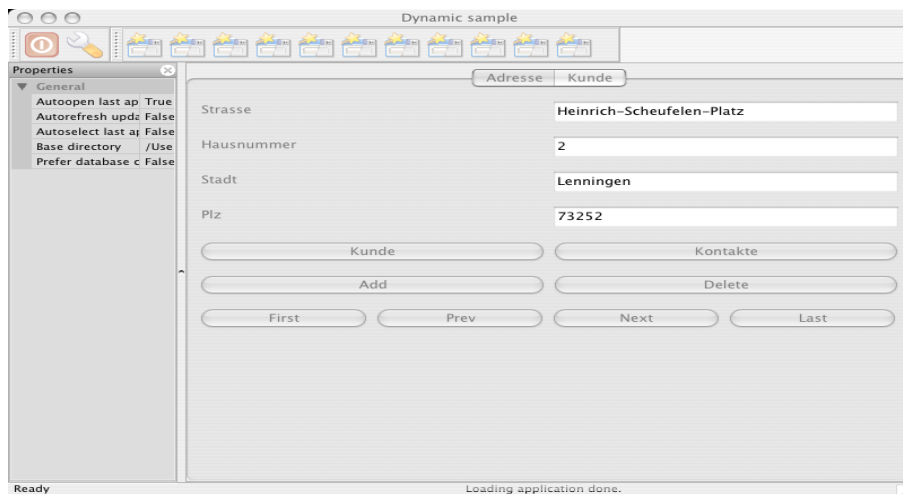
## Try out the generated prototype

After you have exported the UML model to XML and imported it into the prototyping application you would be able to directly run the newly created prototype.

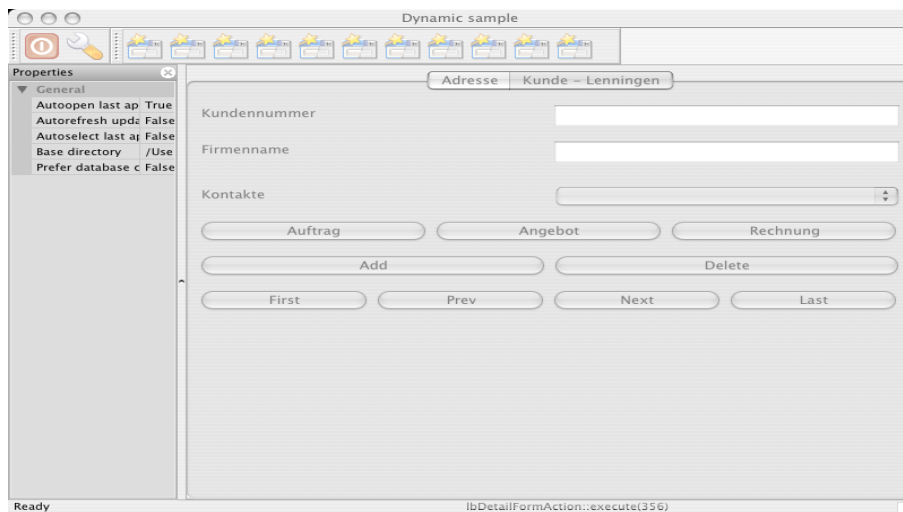
To enter a full customer entry with an address, you first need to enter the address and then save it by navigating once next and back. I do not have a save button yet. Then open the action 'Kunde' from the selected address.

You will get an empty form without any customers. No customers have their location at the given address. Simply press 'Add', then edit the customer data. You do not need to choose the address this way because you have opened the customer form from a given address. This automatically associates the address to the customer. This has advantages and disadvantages.

Here is a screenshot of the address form (Picture 6):



The opened customer form from address form (Picture 7):



There is no data because you have newly created an address. You will see that there is no Address dropdown control. Compare to the picture in adding the customer class. By the way, you see the contacts control. This is there because I have added a contacts class and also associated it to the customer. The resulting UML model is shown on the next page.

## Extend the UML model by a contacts class

At this time you will not be able to do more than storing customers, addresses, contacts, create offers, orders and invoices.

What is missing ?

The offers, orders and invoices do not contain any details about what to be offered and thus it makes no sense. But a first part has been prototyped and you could get some feedback of your customers.

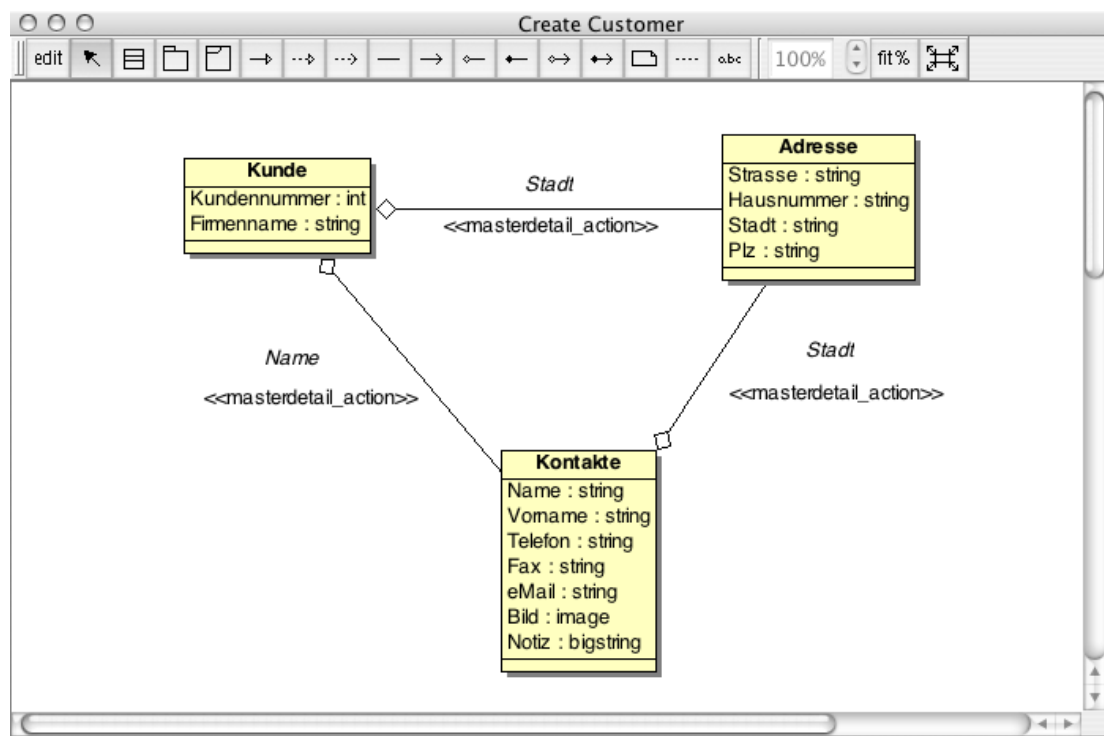
You need to extend the existing UML model to enabling real selling. As while your customers trying out the current prototype you could add the missing stuff and if it is time, reimport the UML model into the prototyping application.

Currently it is very simple to do this. Delete the old application definition and then do the reimport. **Note:** The database model would be still overwritten due to the lack of a suitable XSLT template.

## **Add the contacts class, aggregate to customer**

A customer may have various employees. Each employee may be your contact person. Therefore this is a contact.

The UML model until now (Picture 9):



Here you will see, that a contact may have an address, but it could also assigned the same address as the customer entry has. Also you will see the stereotypes given to each aggregate.

You may open contacts or customers from a given address. The aggregate has in both situations the master link on 'Adresse'. Now you will understand the picture [Picture 6](#)

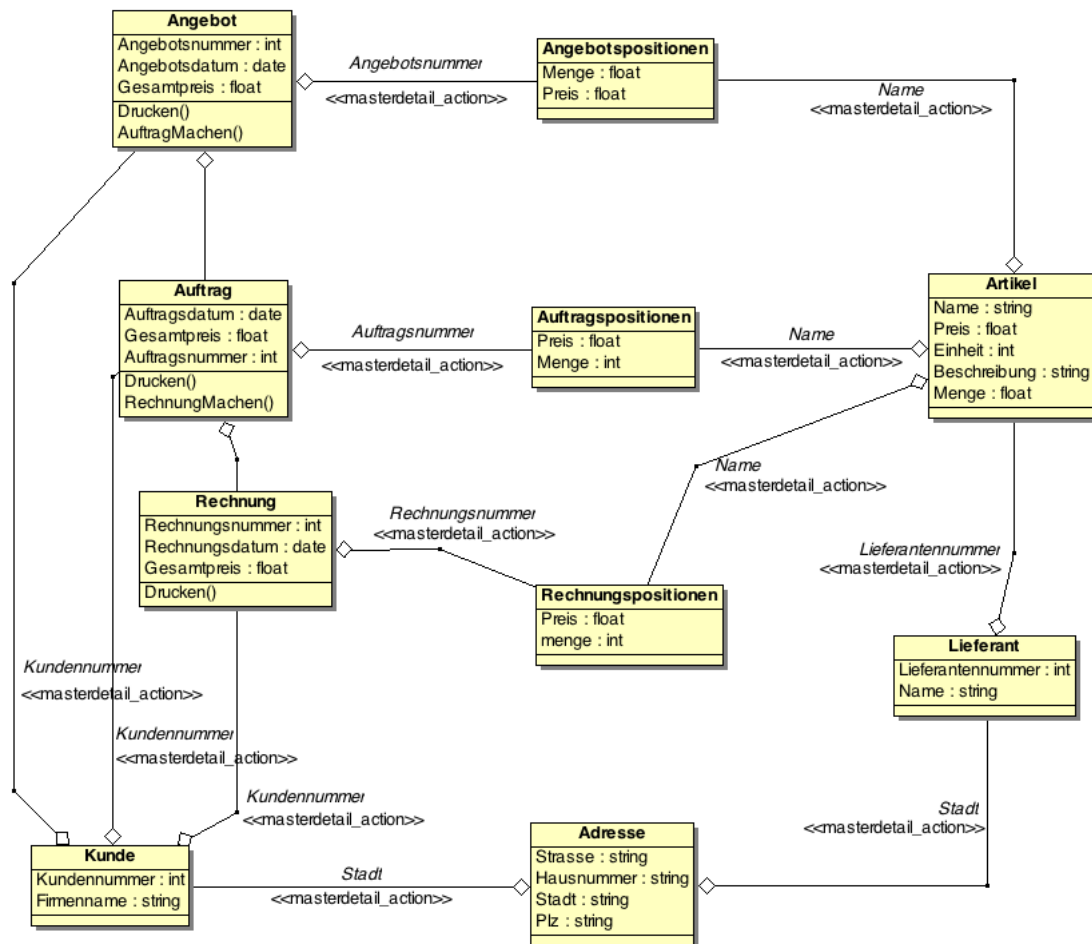
What you also will see – after some customers have been entered – are the address and contact fields showing the town (Stadt) and surname (Name) field's data.

This is a direct relation to the text above the stereotype in [Picture 9](#). This UML model is as complete as a database schema could be created and also an application model could be generated when exported. (XMI)

## Add the link tables linking to articles and the rest

Your offers, orders and invoices need to be enhanced about what articles are about to be offered/sold.

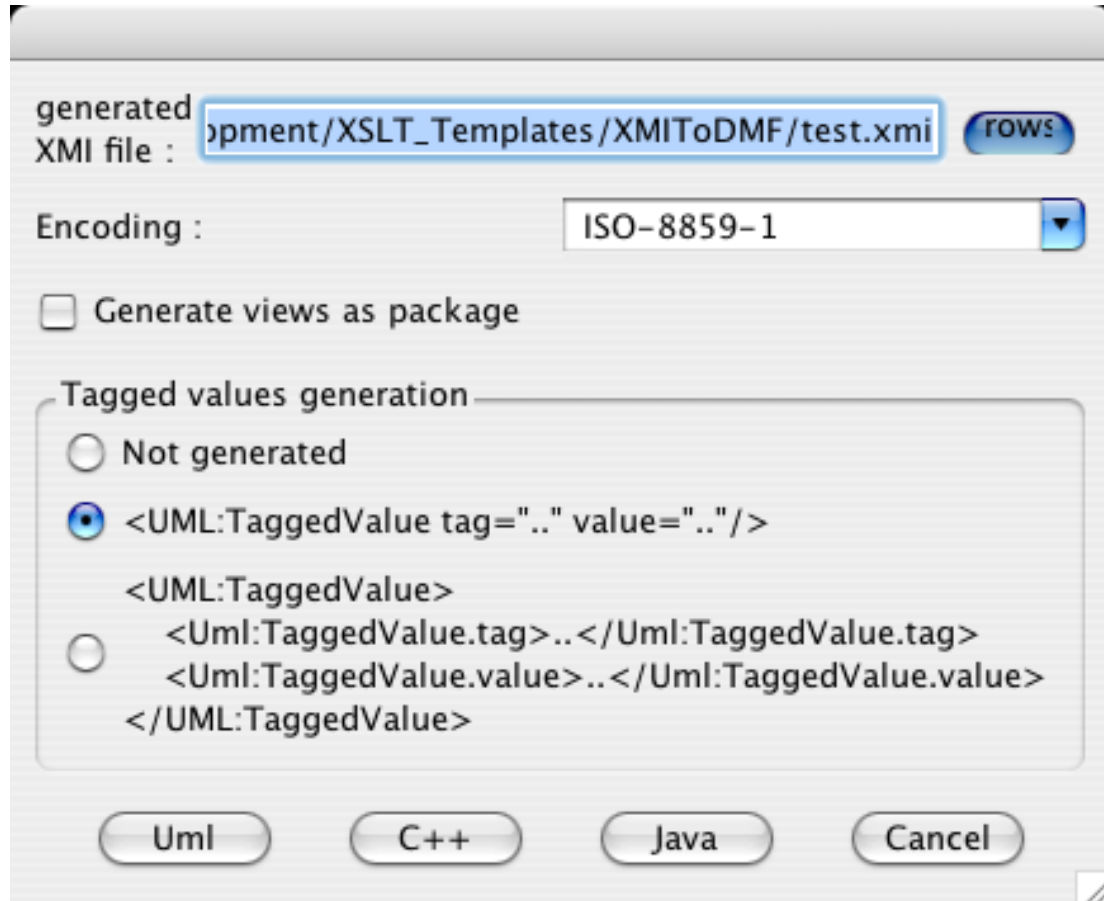
Here you will see how the linking tables to articles look like:



After you have created these classes and linked them together you only need to reexport the UML model and do the reimport steps described in a separate chapter.

## ***Exporting your UML model as XMI file***

After you have created an UML model, you are ready to export it for further development. To do this, open the Tools menu in BoUML and click on 'Generate XMI'. It looks like this (Picture 10):



You need to specify the output file once and leave the rest as shown. After the settings are correct, click on 'Uml' to do the export.

Note:

On Mac OS X, I have to switch the window by pressing 'Meta'+ 'Tab' to see this window.



## Importing the UML model via XMI file

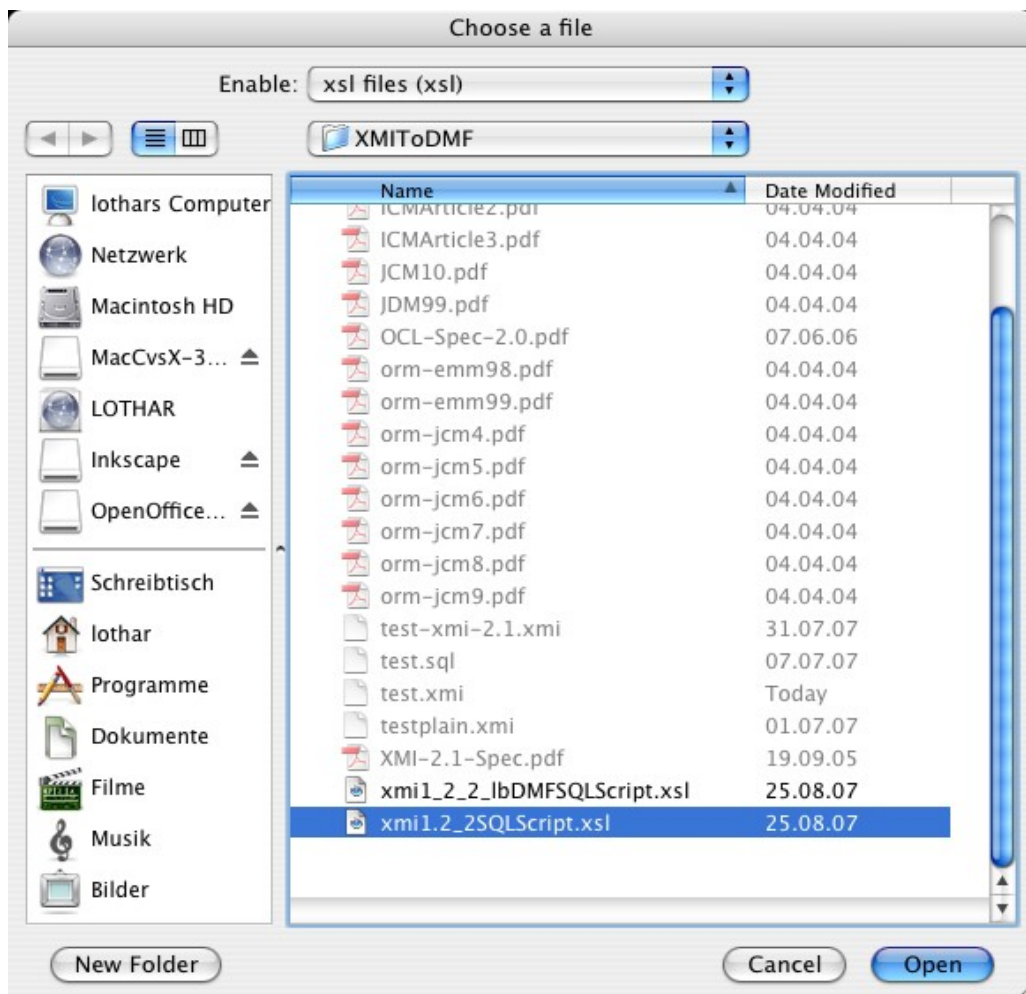
Before you import any XMI file, be sure you not yet have imported that file before. If so have a look at [Deleting an application definition](#)

To import an application definition, start wxWrapper, login with the default user 'user' using 'TestUser' as password. Then select the application 'IbDMF Manager' and proceed with importing by opening the File menu and clicking 'import application from UML (as XMI file)'

The first file to be opened then is the related XMI file you will import. Select your XMI file. Now you get asked to create the database for the application. Choose 'Yes' if you not yet have it imported.

### Step 1. Creating the application database

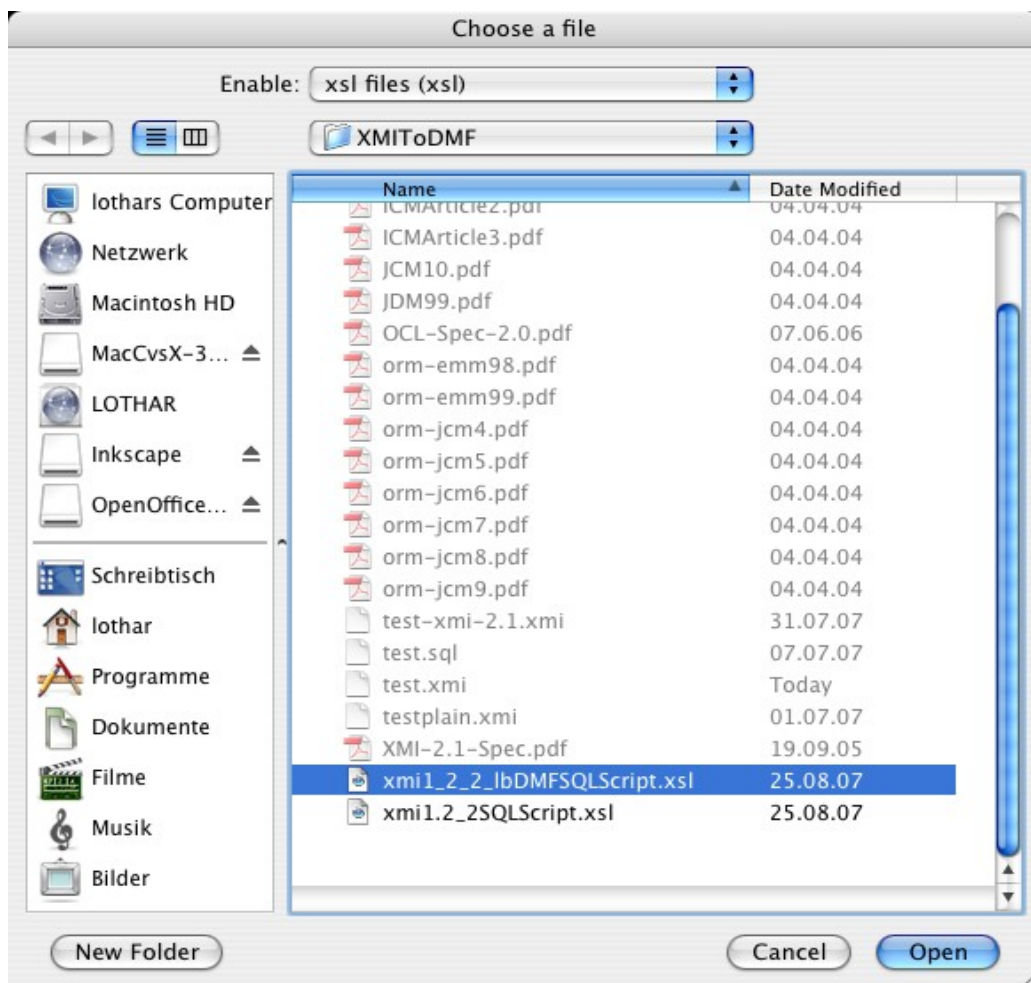
If 'Yes' was selected, you must choose my predefined template as of the picture 12:



If 'No' was pressed, proceed with [Step 2. Creating the application model definition](#)

## Step 2. Creating the application model definition

You will be asked to import the application model definition. Do that by selecting the file shown in Picture 13:



Now you have a new application prototype. To test it, you need to uncheck the menu entry 'Autoload application' in the Edit menu. This is required to enable manual login to a different application.

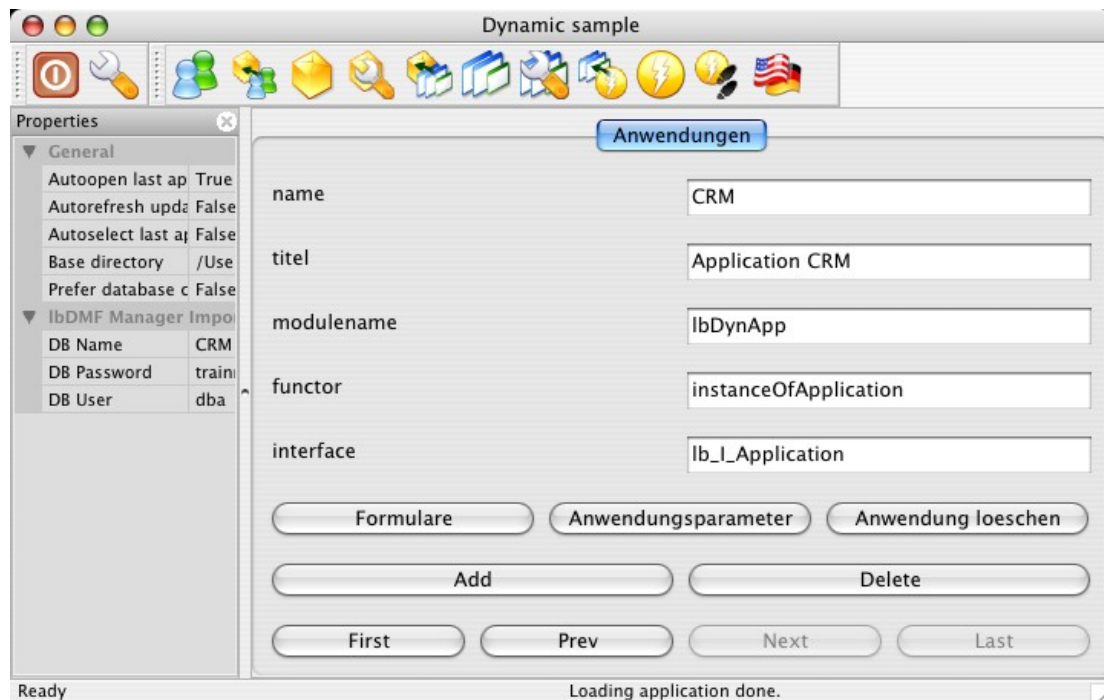
## ***Deleting an application definition***

If you have an application that was imported from an XMI file and you like to reimport it, you need to delete the affected application definition.

Deleting an application definition does not delete the database schema of that application. It only deletes it from the configuration database.

To delete an application definition, start wxWrapper, login with the default user 'user' using 'TestUser' as password. Then select the application 'IbDMF Manager' and proceed with opening the form 'Anwendungen' in menu 'IbDMF Manager' or clicking the yellow box (third from left). Select the intended application and press 'Anwendung loeschen', not 'Delete'. This invokes a stored procedure that cleans up all data related to the selected application.

Picture 11 shows how it looks like:



## Installing the database

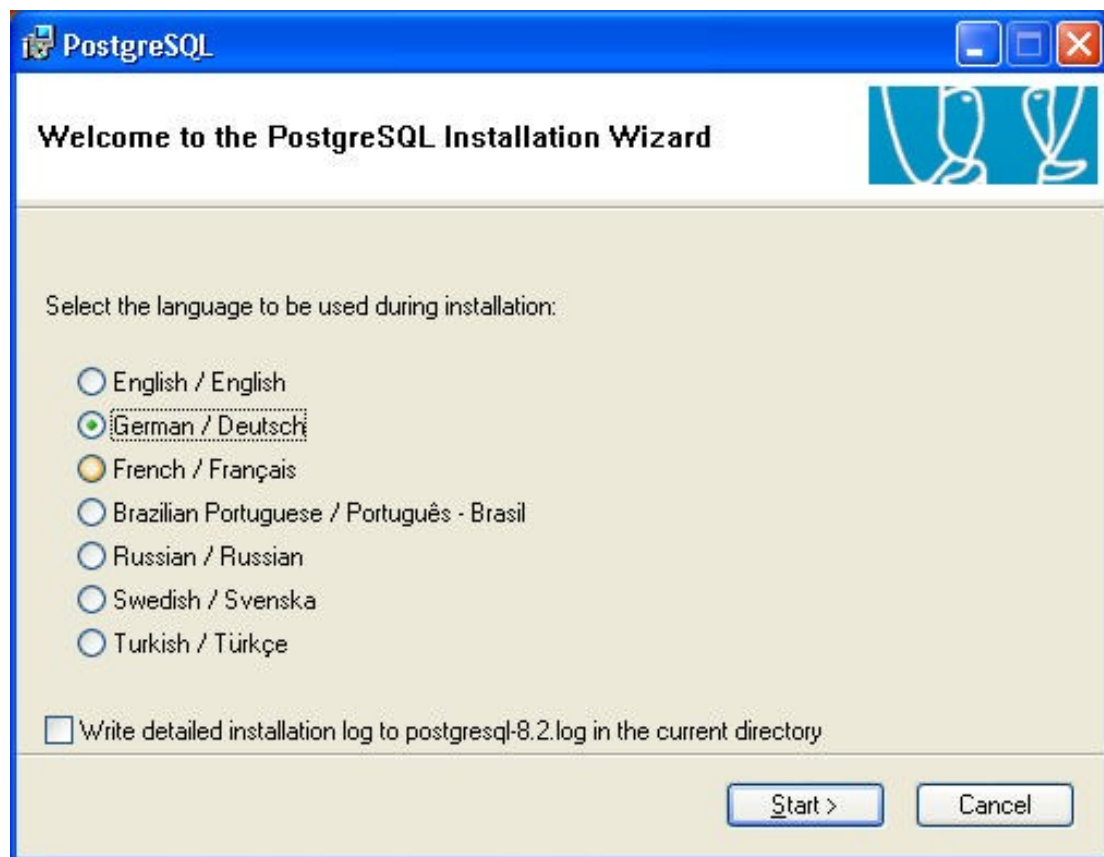
This chapter is divided to the different platforms I do currently support. Before you proceed, you have to decide what database you like to use.

Currently I support [PostgreSQL](#). Other databases are supported, but they are currently not as actual as PostgreSQL. Also the XMI import templates are only available for PostgreSQL yet. Installing other databases may be similar and I expect you have some experience.

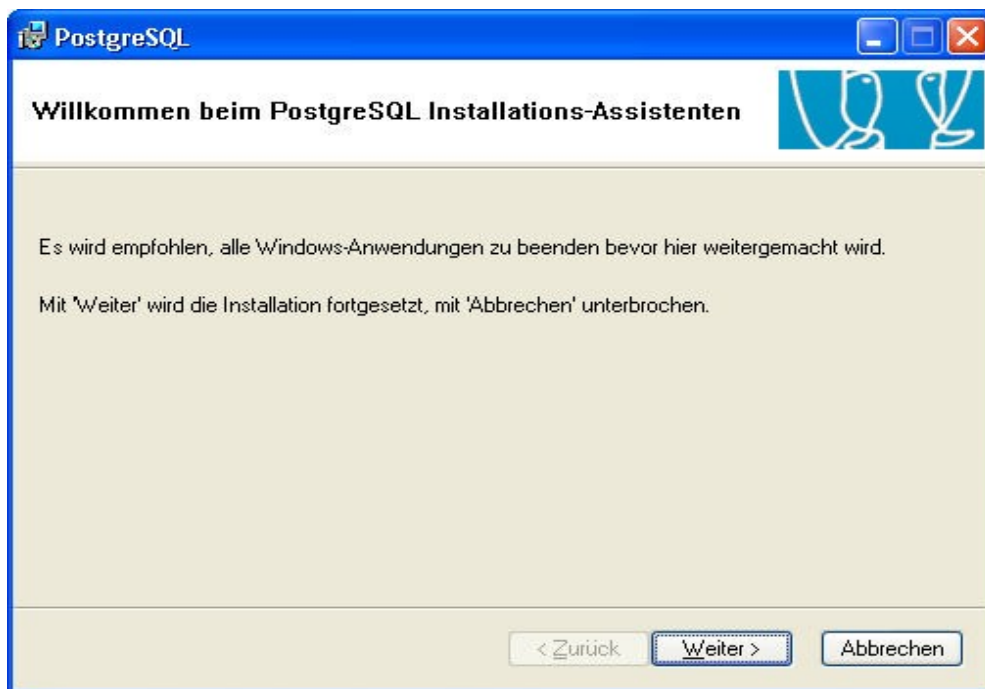
### *Installing on Windows*

Download the [PostgreSQL](#) package for Windows from any mirror in your area. Follow the installation steps as shown in the pictures.

Choose your language (Picture 14):



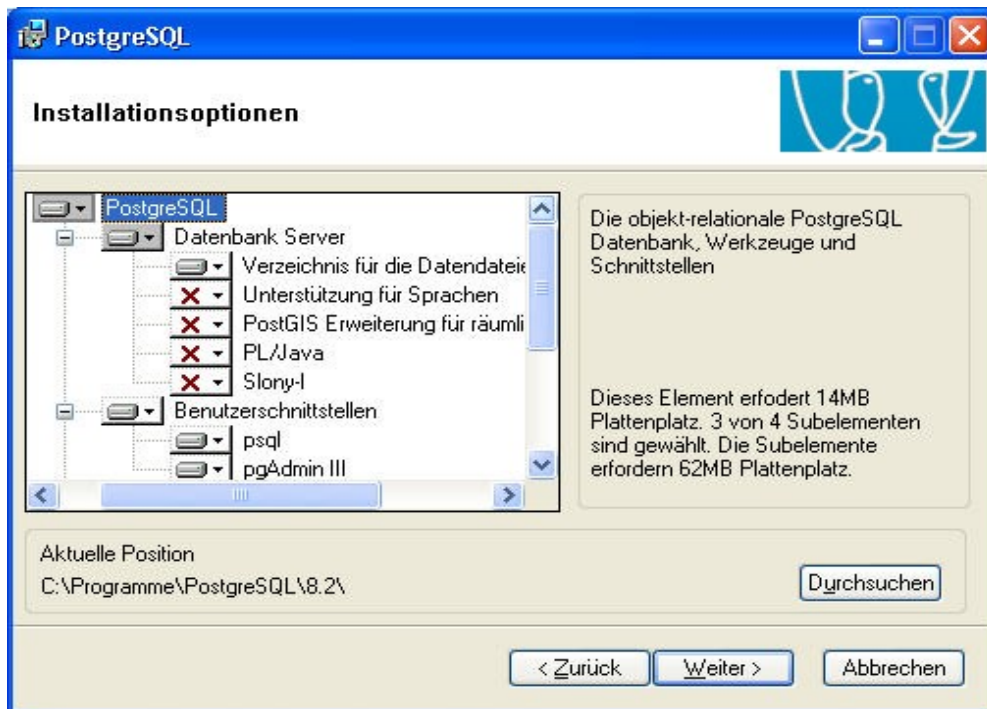
Close all other applications (Picture 15):



Some notes and licensing issues (Picture 16):

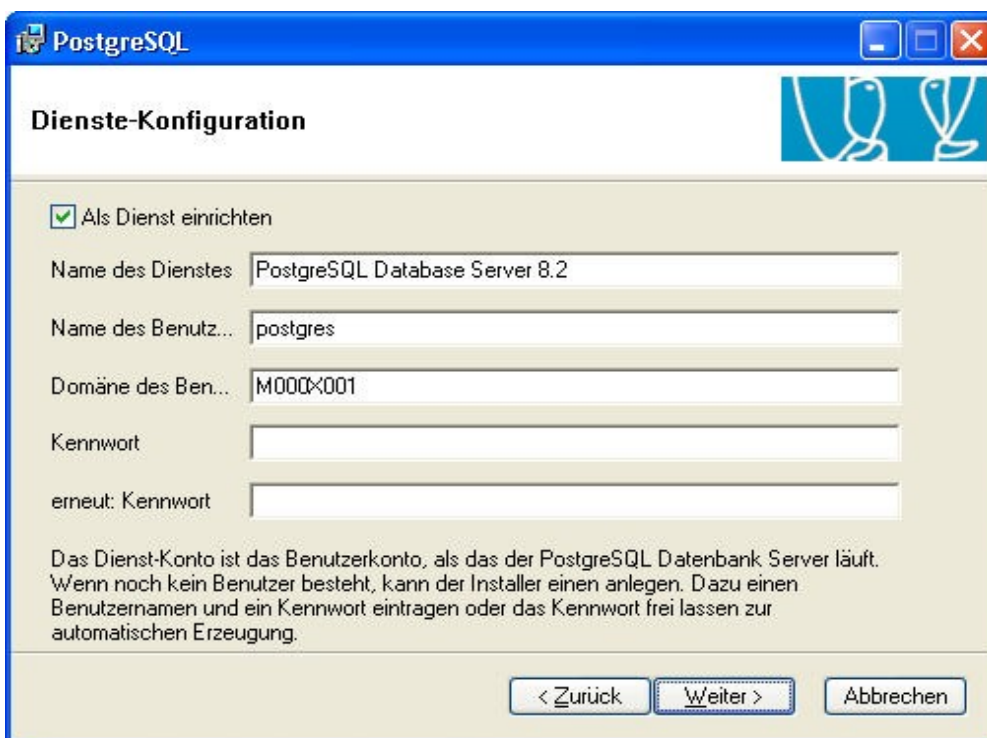


Leave the default settings (Picture 17):



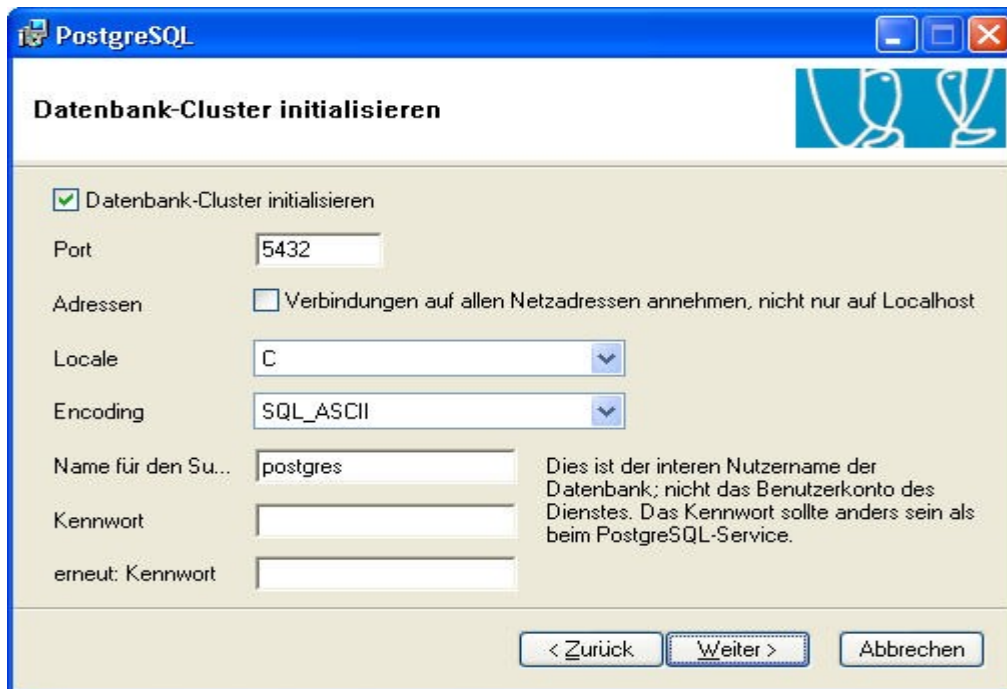
Note: You must at least install an ODBC driver and the database itself.

Setup an user account for the database (Picture 18):





Setup some global database settings (Picture 19):



The screenshot shows the 'Datenbank-Cluster initialisieren' window in the PostgreSQL installer. The window has a blue title bar with the PostgreSQL logo and standard window controls. The main area is light beige. At the top, the title 'Datenbank-Cluster initialisieren' is displayed. Below it, there is a checkbox labeled 'Datenbank-Cluster initialisieren' which is checked. The 'Port' is set to '5432'. The 'Adressen' section has a checkbox 'Verbindungen auf allen Netzadressen annehmen, nicht nur auf Localhost' which is unchecked. The 'Locale' is set to 'C' and the 'Encoding' is set to 'SQL\_ASCII'. The 'Name für den Su...' field contains 'postgres'. The 'Kennwort' and 'erneut: Kennwort' fields are empty. A note on the right states: 'Dies ist der interne Nutzernamen der Datenbank; nicht das Benutzerkonto des Dienstes. Das Kennwort sollte anders sein als beim PostgreSQL-Service.' At the bottom, there are three buttons: '< Zurück', 'Weiter >', and 'Abbrechen'.

**Datenbank-Cluster initialisieren**

☒ Datenbank-Cluster initialisieren

Port: 5432

Adressen: ☐ Verbindungen auf allen Netzadressen annehmen, nicht nur auf Localhost

Locale: C

Encoding: SQL\_ASCII

Name für den Su...: postgres

Kennwort:

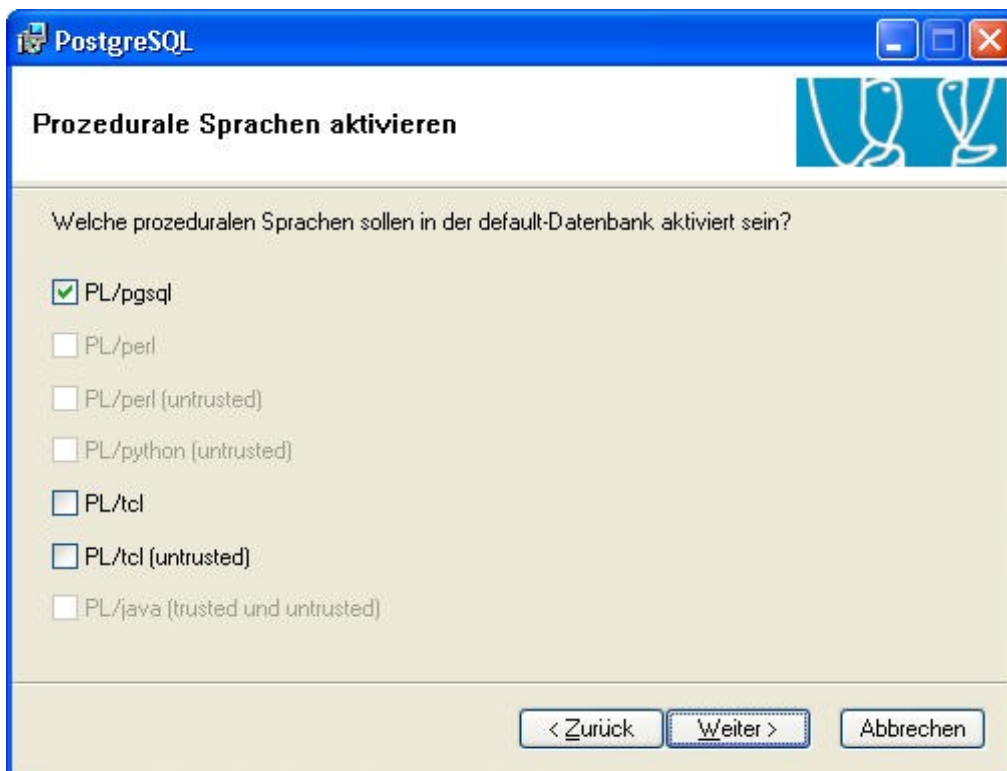
erneut: Kennwort:

Dies ist der interne Nutzernamen der Datenbank; nicht das Benutzerkonto des Dienstes. Das Kennwort sollte anders sein als beim PostgreSQL-Service.

< Zurück Weiter > Abbrechen

Note: You don't need to enter the password. It would be generated.

Leave these settings (Picture 20):



The screenshot shows the 'Prozedurale Sprachen aktivieren' window in the PostgreSQL installer. The window has a blue title bar with the PostgreSQL logo and standard window controls. The main area is light beige. At the top, the title 'Prozedurale Sprachen aktivieren' is displayed. Below it, the question 'Welche prozeduralen Sprachen sollen in der default-Datenbank aktiviert sein?' is shown. There are several checkboxes for procedural languages: 'PL/pgsql' (checked), 'PL/perl' (unchecked), 'PL/perl (untrusted)' (unchecked), 'PL/python (untrusted)' (unchecked), 'PL/tcl' (unchecked), 'PL/tcl (untrusted)' (unchecked), and 'PL/java (trusted und untrusted)' (unchecked). At the bottom, there are three buttons: '< Zurück', 'Weiter >', and 'Abbrechen'.

**Prozedurale Sprachen aktivieren**

Welche prozeduralen Sprachen sollen in der default-Datenbank aktiviert sein?

☒ PL/pgsql

☐ PL/perl

☐ PL/perl (untrusted)

☐ PL/python (untrusted)

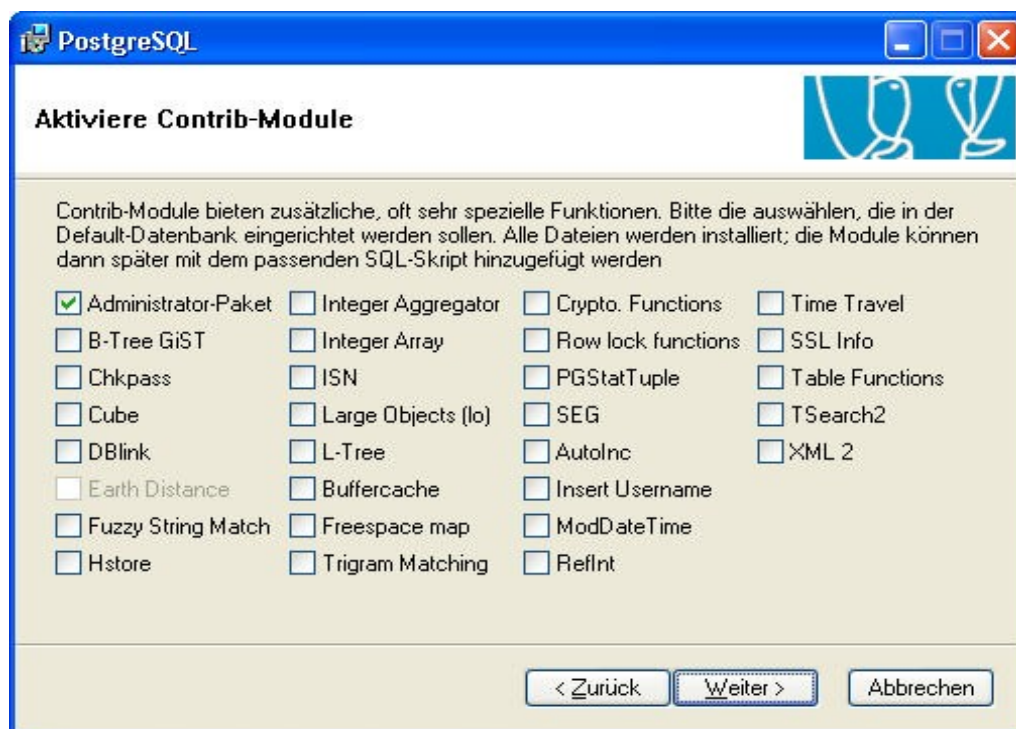
☐ PL/tcl

☐ PL/tcl (untrusted)

☐ PL/java (trusted und untrusted)

< Zurück Weiter > Abbrechen

Also leave these settings. The software didn't need more (Picture 21):



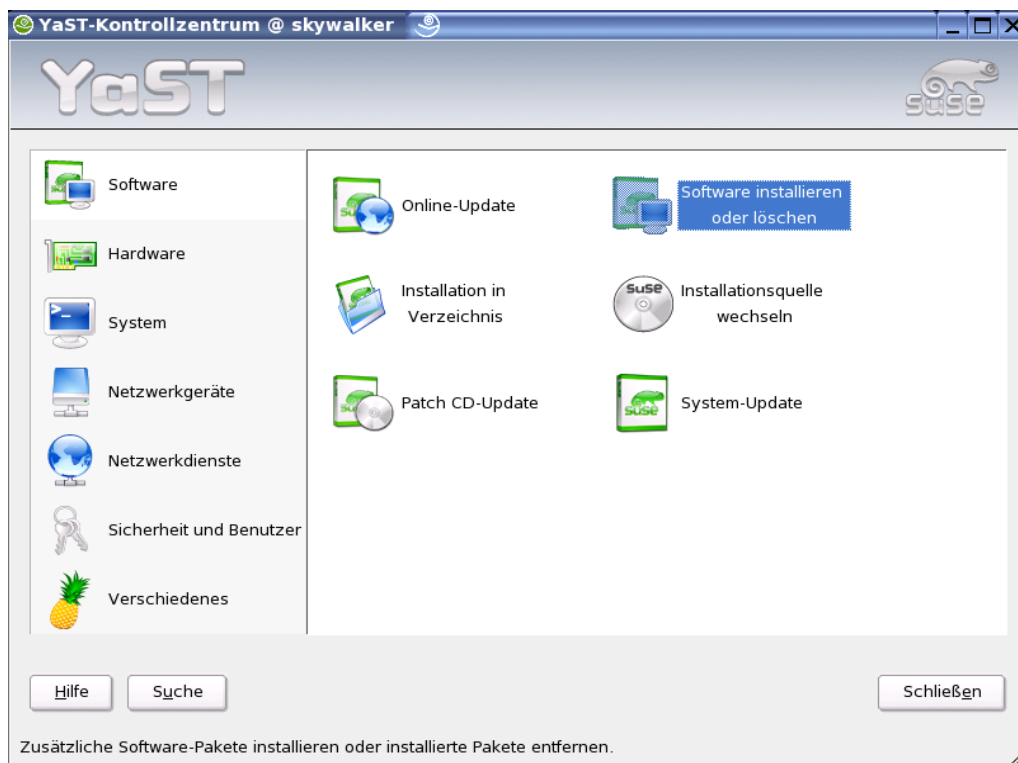


## Installing on Linux

Typically there are source distributions from the database vendor. If you like to install the database from these files you must be familiar with the UNIX source installation process.

Exemplarily I show how to install the database server on a SuSE Installation.

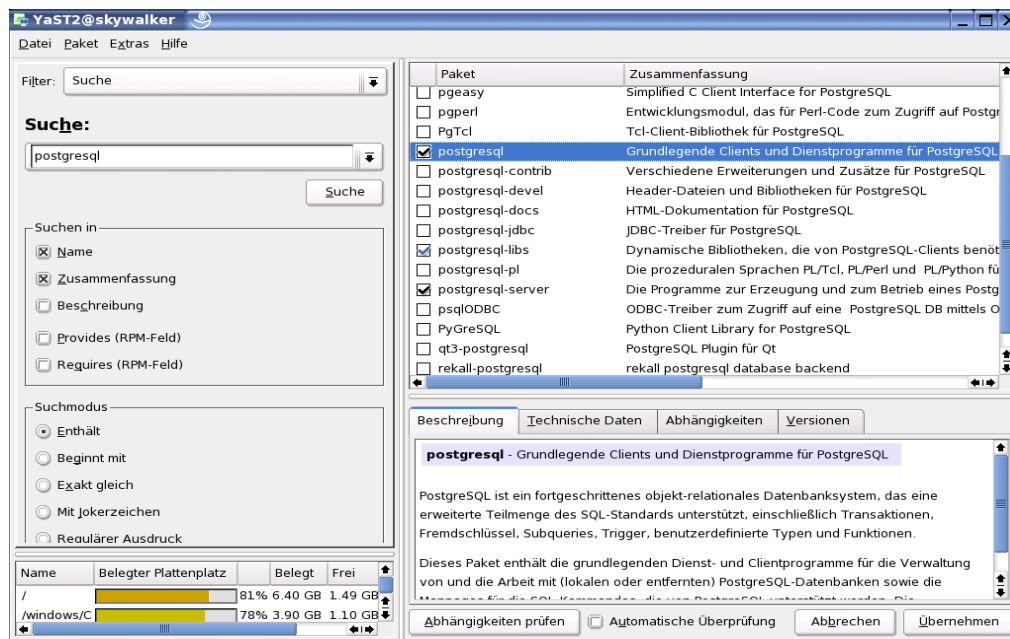
First of all you need to start Yast. I use the GUI variant. Select the following as shown:



This will start the software installation module. On the next page you will see what modules should be installed. Simply do a search for postgresql and you will get the list matching that word.

To install the server, you select postgresql, postgresql- server and the postgresql- libs if not yet selected.

Software selected to install the database server:



After you have installed the server, you need to do some steps on a console to finish the installation.

Start a console and enter the following commands as user postgres:

```
# /etc/init.d/postgresql start
```

Edit the database configuration file to allow tcp connections. The file is `/var/lib/pgsql/data/postgresql.conf`

The line, containing `tcpip_socket`, must be activated and 'false' replaced by 'true'

In the file `/var/lib/pgsql/data/pg_hba.conf`, you need to set permissions for those users and hosts to allow to connect from.

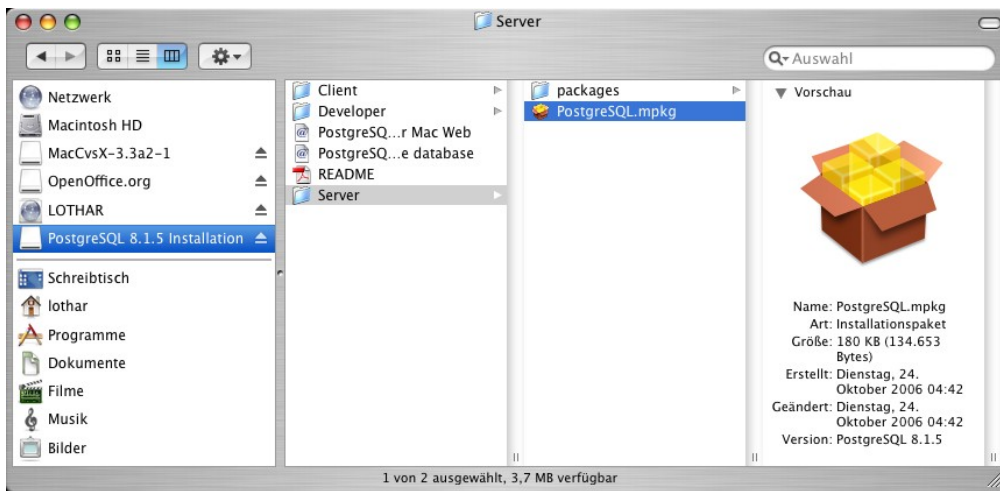
A sample may be this line:

```
host    all             all             192.168.0.0     255.255.255.0   trust
```

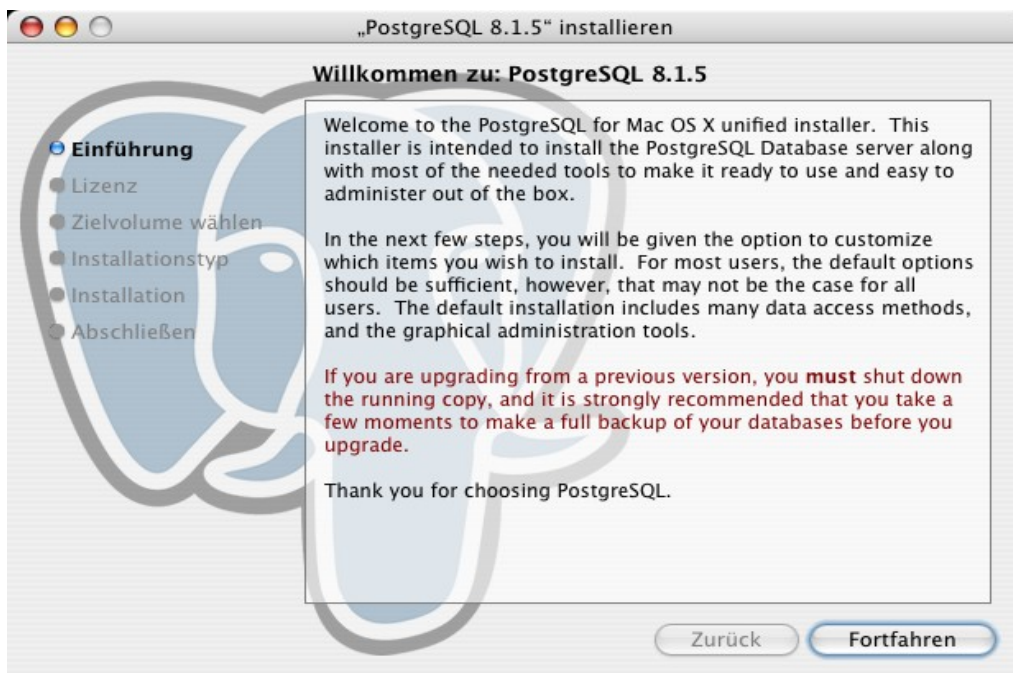
## Installing on Mac OS X

Download the [PostgreSQL](#) package for Mac OS X from any mirror in your area. Follow the installation steps as shown in the pictures.

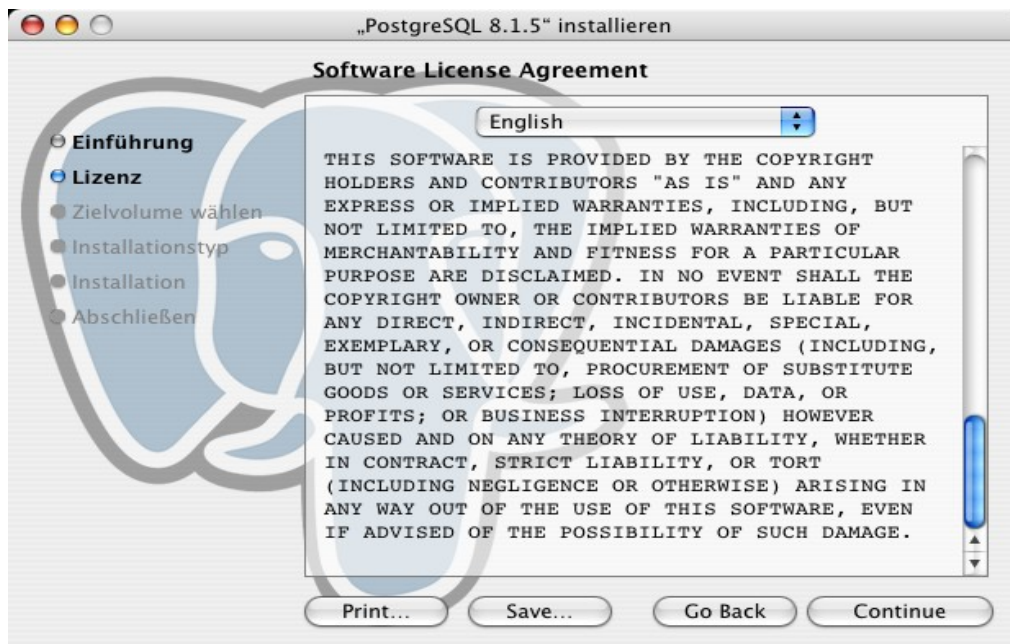
Mount the package if not yet done and execute the server installer:



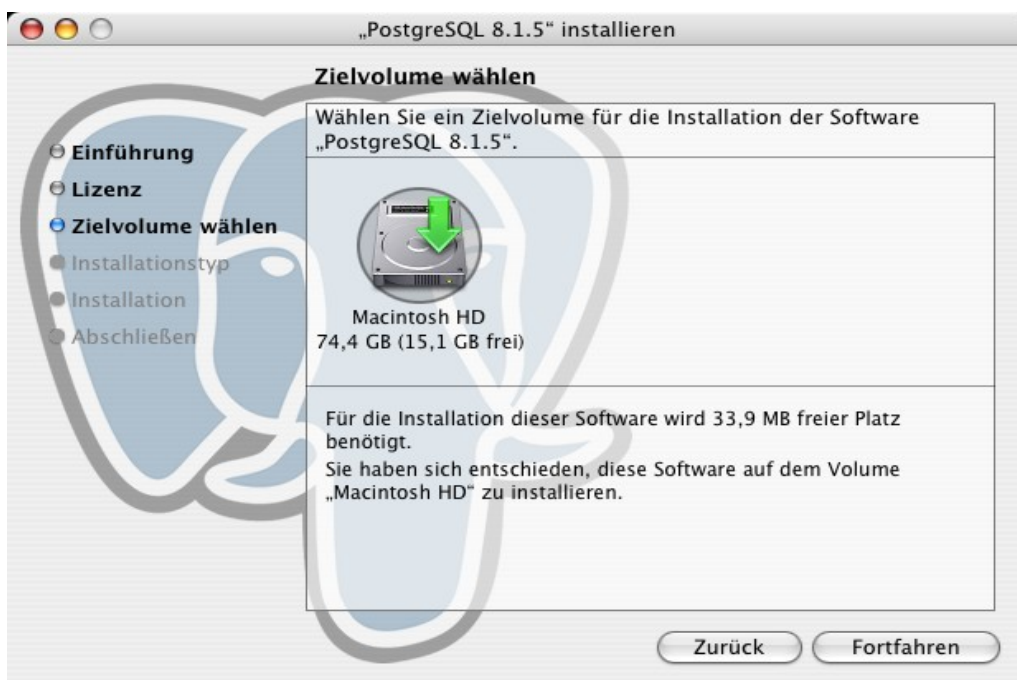
Do a backup and stop database service if you do upgrade:



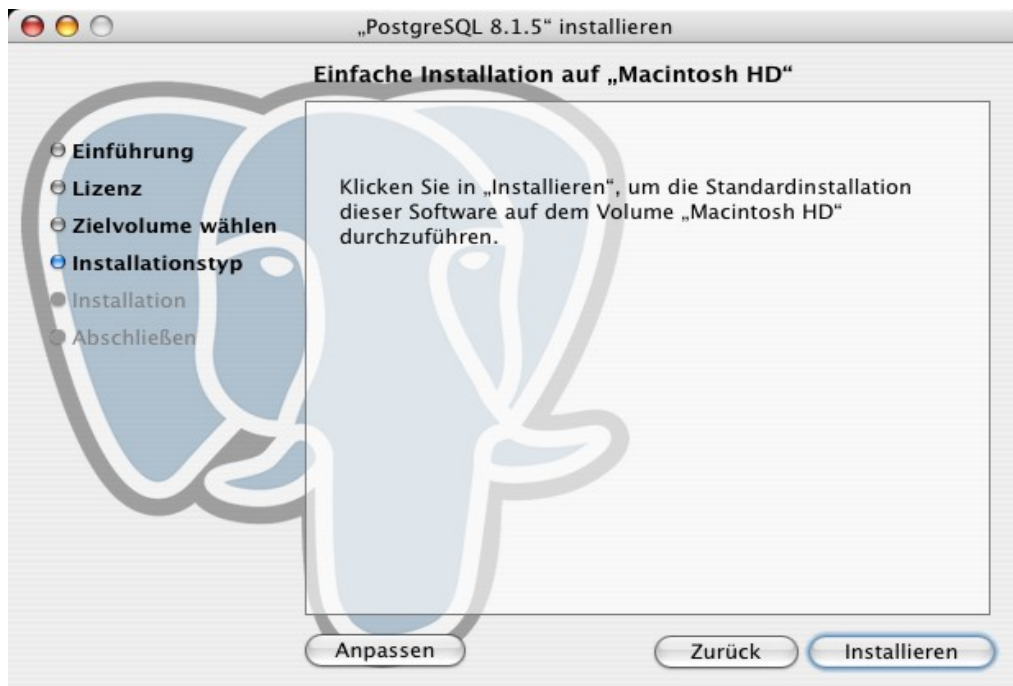
License stuff:



Select the installation target:



You are ready to install if you like to have all components:



Note: If you do only like the server, you need to click on 'Anpassen' or 'Customize' and select the packages you want to install.

## ***Installing on Solaris***

I have not seen any GUI installer yet for the database. Typically you get the database as source tgz file and follow the typical UNIX make commands.

```
# Configure & make & make install
```

You may follow the installation guide for Linux, but not the GUI part, only the database configuration itself.

Todo. Describe exact setup procedure, so that the database will start on system boot.

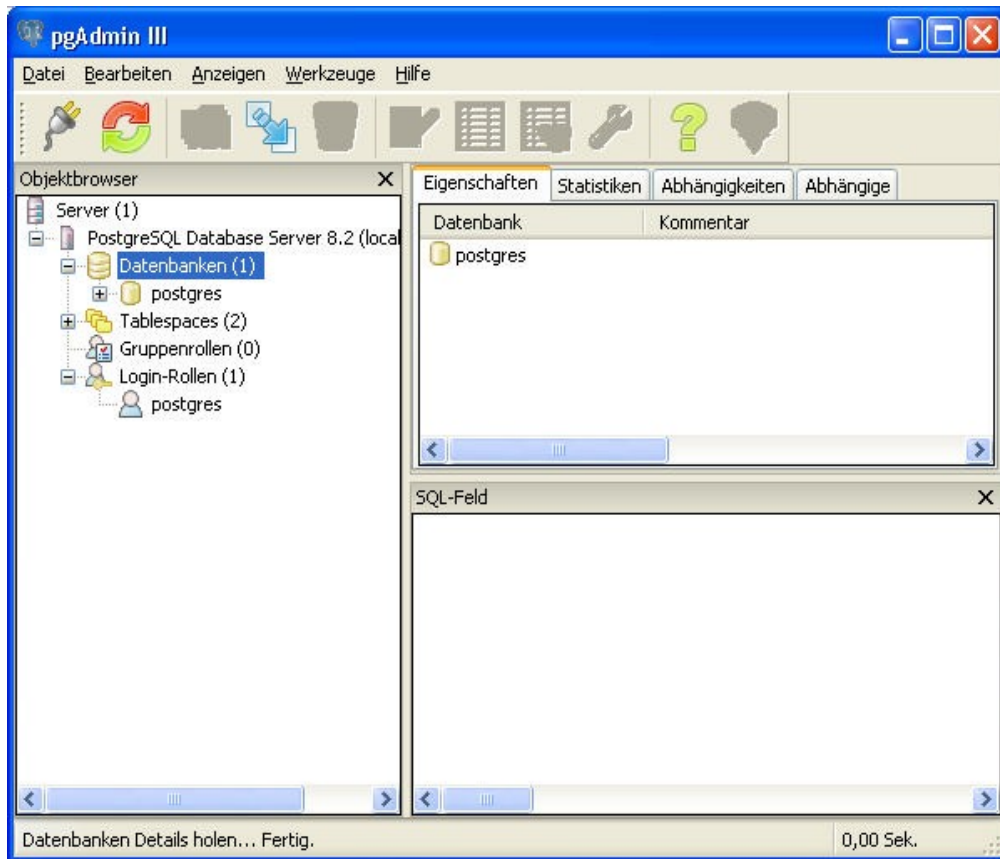
## **Configuring the database at first time**

This chapter is divided to the different platforms I do currently support.

Currently I support [PostgreSQL](#). Other databases are supported, but they are currently not as actual as PostgreSQL. Also the XMI import templates are only available for PostgreSQL yet. Configuring other databases may be similar and I expect you have some experience.

## Configuring on Windows

After you have installed the database you will get this (Picture 22):



On 'Login- Rollen' create a new login used in the application.

After you have the database user you create the following two databases:

Database CRM: For the application you will model based on this documentation.

Database Ibdmf: As the system database for the application configurations.

Enter the role name as shown (Picture 23):

**Neue Login-Rolle...**

Eigenschaften Rollenmitgliedschaft Variablen SQL

Rollenname: dba

OID:

Kann einloggen: ☒

Passwort: .....

Passwort (nochmal): .....

Konto erlischt: [ ] [ ]

**Rollenprivilegien**

- ☐ Vererbt Rechte von Vaterrollen
- ☐ Superuser
- ☐ Kann Datenbanken anlegen
- ☐ Kann weitere Rollen anlegen
- ☐ Kann Katalog direkt modifizieren

Replikation anwenden: [ ]

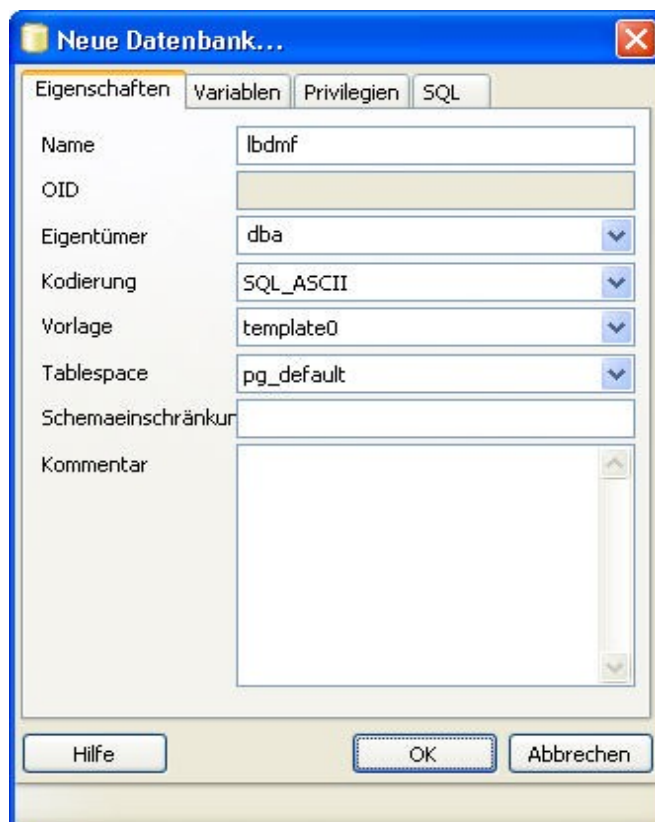
Hilfe OK Abbrechen



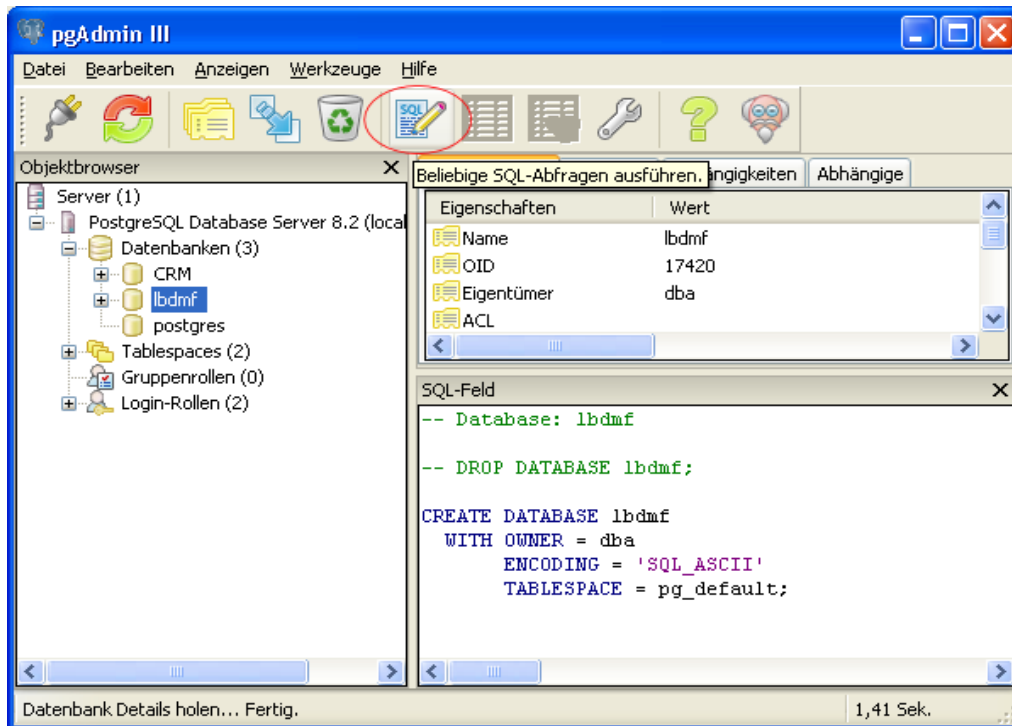
On the Datenbanken tree node (databases), you need to right click and select create new database. (Neue Datenbank...) (Picture 24):



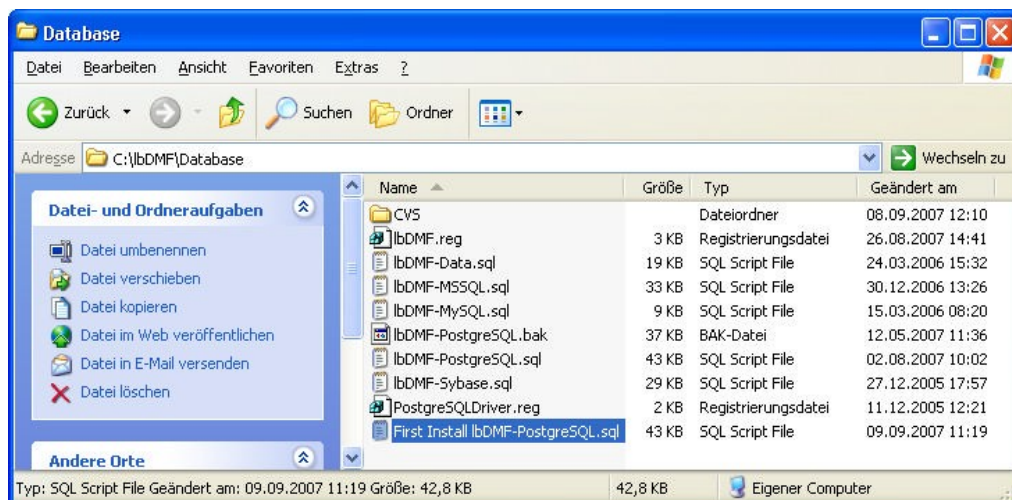
Settings for the system database (Picture 25):



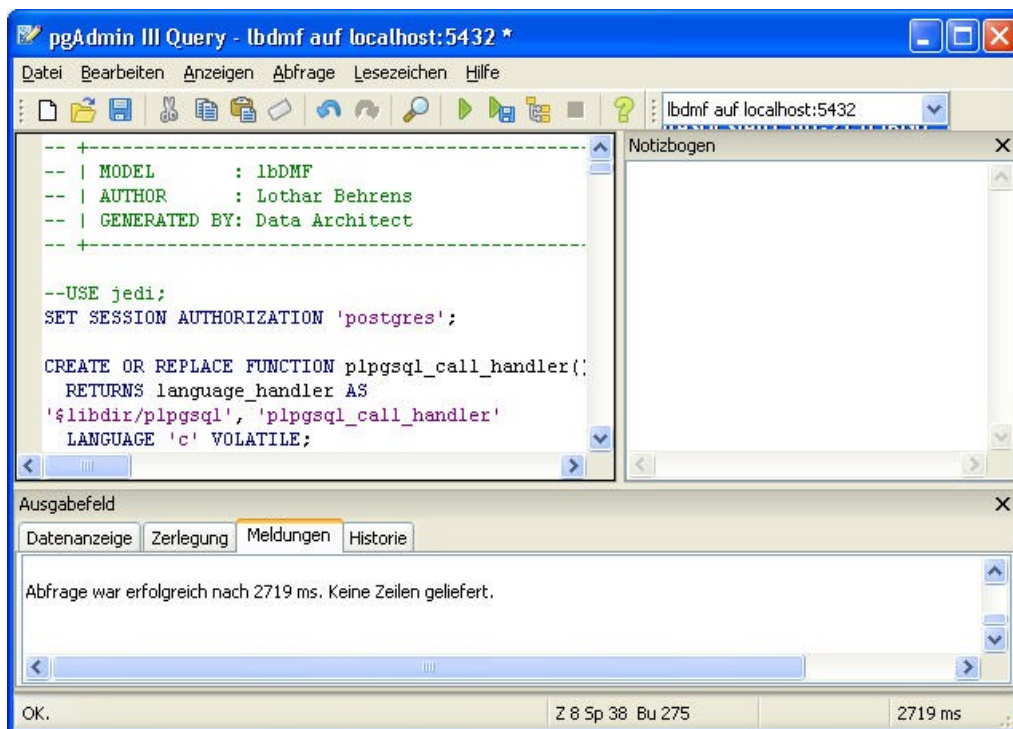
After you have created the databases, select the system database (lbdmf) and open the SQL window (Picture 26):



Copy and paste the contents of the 'First Install lbdmf- PostgreSQL.sql' file, so it looks like picture 28 (Picture 27):



Copied SQL query from file in last picture (Picture 28):



After executing the query (by the green triangle), you would be able to use the application you install in the next steps. Never use this file twice, it will fail, because it expects an empty database.

## ***Configuring on Linux***

To enable a working environment, you need to setup a database schema. To do this, login to the server and enter the following commands as user postgres:

```
# createuser dba -P -E
# createdb lbdmf
# createdb CRM
# psql -d lbdmf -f First\ Install\ lbDMF-PostgreSQL.sql
```

### **Note:**

On the user creation, you will be asked to enter a password and repeat it.

The SQL script may located in your installation of the lbDMF software. For the source distribution typically in

~/CPP/Database/First\ Install\ lbDMF- PostgreSQL.sql

## ***Configuring on Mac OS X***

The configuration on Mac OS X is similar to the Windows configuration process, because there is also the PGAdmin application available.

The only difference is the location of the file you need as database script. This depends on the location where you have installed the client software as of [Installing the client application](#).

## ***Configuring on Solaris***

Follow the Linux configuration guide. The PostgreSQL client tools may be available there too.

## **Setup ODBC configuration**

Now you need to setup the ODBC configuration. For this, you need to get an ODBC [driver](#) for PostgreSQL. Use a driver that supports full cursor capabilities, not only forward - only cursors. The 07.03.0200 version does that. Other versions and development states are currently unknown.

### ***Setup ODBC on Windows***

## Setup ODBC on Linux

Please download the TGZ archive from the [driver](#). Install it by unpacking it and build it with the typical commands like:

```
# ./configure
# make
# make install
```

In my sample the driver is installed as seen in the 'Driver' line. If the installation of the driver is in /usr/lib, please correct the 'Driver' line.

In the server installation sample I have used the subnet 192.168.0.0 with a mask of 255.255.255.0. Assuming a server address of 192.168.1.0, you will have to fill in your local (home directory) .odbc.ini file the following content:

The .odbc.ini file:

```
[lbDMF]
Driver           = /usr/local/lib/psqlodbc.so
Description      = Configuration for DMF
Servername       = 192.168.0.1
Username         = dba
Password         =
Database         =
Port             = 5432
Protocol         = 6.4
BoolsAsChar      = 0
ShowSystemTables = 1
TrueIsMinus1     = 1
UpdatableCursors = 1
Trace            = 1
ByteaAsLongVarBinary = 1
TraceFile        = /home/lothar/lbDMF.log
Debug            = 1
DebugFile        = /home/lothar/lbDMF.debug.log

[CRM]
Driver           = /usr/local/lib/psqlodbc.so
Description      = Configuration for CRM
Servername       = 192.168.0.1
Username         =
```

```
Password          =  
Database          = CRM  
Port              = 5432  
Protocol          = 6.4  
BoolsAsChar       = 0  
ShowSystemTables  = 1  
TrueIsMinus1      = 1  
UpdatableCursors  = 1  
Trace             = 1  
ByteaAsLongVarBinary = 1  
TraceFile         = /home/lothar/CRMDB.log  
Debug             = 1  
DebugFile         = /home/lothar/CRMDB.debug.log
```

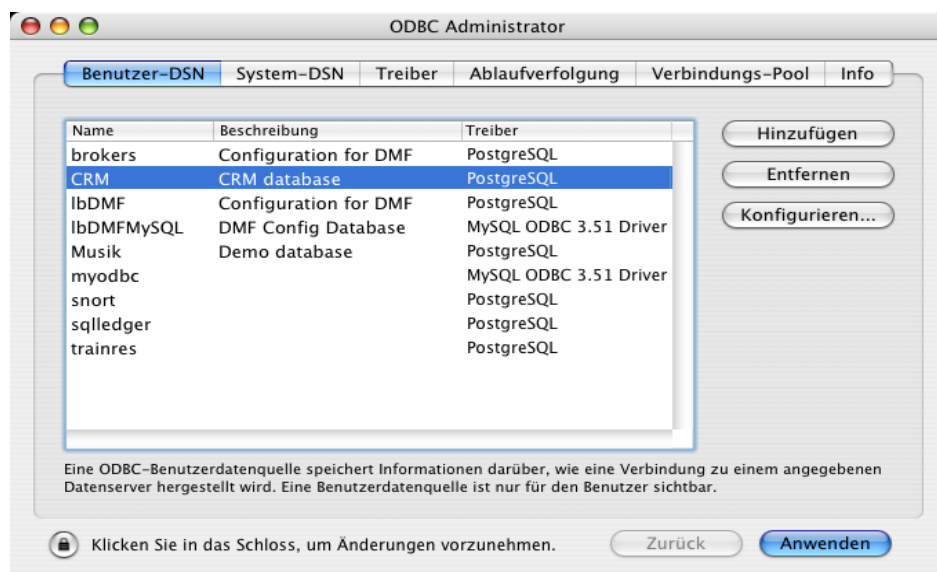
Note:

Replace the user 'lothar' with your user name.

Now you could install the software and enjoy.

## ***Setup ODBC on Mac OS X***

There are GUI applications available to setup ODBC configurations, but they are not better than doing it in the plain `odbc.ini` file. To install the driver, follow [Setup ODBC on Linux](#). There is also no difference in the TGZ file version to be downloaded and installed. But you may use 'ODBC Administrator' GUI to setup the database configuration.





Here you see the settings for lbDMF (system database):

The screenshot shows a dialog box titled 'Name der Datenquelle (DSN): lbDMF'. Below the title bar, there is a field for 'Beschreibung:' with the value 'Configuration for DMF'. The main area of the dialog contains a table with two columns: 'Schlüsselwort' and 'Wert'. The table lists various configuration parameters for the lbDMF system database. At the bottom of the dialog, there are four buttons: 'Hinzufügen', 'Entfernen', 'Abbrechen', and 'OK'.

Schlüsselwort	Wert
Servename	192.168.150.3
Username	dba
Password	trainres
Database	lbdmf
Port	5432
Protocol	6.4
BoolsAsChar	0
ShowSystemTables	1
UpdatableCursors	1
TruelsMinus1	1
Trace	0
TraceFile	/Users/lothar/lbDMF.log
Debug	0
DebugFile	/Users/lothar/lbDMF.debug.log
ByteaAsLongVarBinary	1

Here you see the settings for CRM (test database):

The screenshot shows a dialog box titled 'Name der Datenquelle (DSN): CRM'. Below the title bar, there is a field for 'Beschreibung:' with the value 'CRM database'. The main area of the dialog contains a table with two columns: 'Schlüsselwort' and 'Wert'. The table lists various configuration parameters for the CRM test database. At the bottom of the dialog, there are four buttons: 'Hinzufügen', 'Entfernen', 'Abbrechen', and 'OK'.

Schlüsselwort	Wert
Servename	192.168.150.3
Username	dba
Password	trainres
Database	CRM
Port	5432
Protocol	6.4
BoolsAsChar	0
ShowSystemTables	1
TruelsMinus1	1
UpdatableCursors	1
Trace	0
TraceFile	/Users/lothar/lbDMF.log
Debug	0
DebugFile	/Users/lothar/lbDMF.debug.log
ByteaAsLongVarBinary	1

## ***Setup ODBC on Solaris***

Simply follow the [Setup ODBC on Linux](#) chapter. Other variants would involve installation of GUI versions of ODBC setup software. I didn't have tested the GUI clients of [unixODBC](#).

# Installing the client application

The client application is either a binary package without sourcecode or it is available in the sourcecode package. In the sourcecode variant, you need to check if the prerequisites are fulfilled. Go to the chapter [Preparing to build the source distribution](#) and read the corresponding chapter for your OS. In the binary package variant, go to the chapter [Installing a binary package](#) and read the corresponding chapter for installing binary packages.

## ***Installing from source distribution***

There are different ways to get the source distribution. One may be SRPM file, TGZ file, ZIP file and also a Setup application for Windows. But all needs some prerequisites to build correctly.

## **Preparing to build the source distribution**

Building the sourcecode requires a proper installed software development tool. Currently I do support [Open Watcom 1.6](#) for the Windows platform and the GNU compilers on the other platforms.

Be sure having installed the compiler tools. Installing it is not discussed in this documentation.

### ***Prepare on Windows***

#### **Install wxWidgets package**

### ***Prepare on Linux***

You need to check if the following libraries are installed. This is done this way:

```
# xml2-config --version
2.6.7

# xslt-config --version
1.1.2
```

```
# wx-config --version  
2.6.2
```

If one of these commands fail, you may be missing the development installation of that package. **At least these version numbers must be installed.**

To install the missing packages, you may load Yast and search for the following:

XML2: libxml2

XSLT: libxslt

wxWidgets: wxGTK

Note: On openSUSE 10.1 I do not have found a wxGTK-devel package. To get a source distribution installed, you need to use the TGZ variant.

if found, also check their development versions and also install mandatory libraries. Yast looks similar to this [Image](#).

If wxWidgets is not available in the minimal version, download it [here](#). Then build it as follows:

## Building wxWidgets on Linux from source TGZ

If you build the project and thus installing the client that way, you either need a precompiled version of wxWidgets or here the TGZ version (wxGTK).

Unpack the source TGZ file and cd into the newly created directory. Issue the commands as shown in the sample below:

```
# ./configure --enable-monolithic --enable-shared  
# make  
# sudo make install
```

I don't like to tell you how to install wxWidgets in RPM format, because this is too simple. Either you find it in Yast or you download it from the above link, if the developers provide a SRPM package in

future releases.

### ***Prepare on Mac OS X***

You need to follow the steps from [Building wxWidgets on Linux from source TGZ](#) to get the wxWidgets package, but download the wxMac version.

Also you need to install an ODBC driver for the database. Please follow the [Setup ODBC on Mac OS X](#) chapter for that.

### ***Prepare on Solaris***

You need to follow the steps from [Building wxWidgets on Linux from source TGZ](#) to get the wxWidgets package, but download the wxGTK version.

Also you need to install an ODBC driver for the database. Please follow the [Setup ODBC on Solaris](#) chapter for that.

## **Installing SRPM distribution**

Download the [latest](#) source RPM file and install it as usual. If you are familiar with that, you may skip reading this chapter.

You install any given RPM file by issuing the following command:

```
# rpm -i <rpm file>
```

You build directly from source RPM via the following command:

```
# rpmbuild -rebuild <rpm file>
```

## **Installing TGZ distribution**

Download the [latest](#) source TGZ file and install it as usual. You install any given TGZ file by issuing the following command:

```
# tar xvzf <tgz file>
```

## **Installing from Setup executable**

### **Build the source**

Building the source is very easy, if all dependencies are resolved. Follow the instructions of that chapter that fit your needs.

### ***Build from source RPM file***

You may either install the rpm file and follow the next chapter expect the directory where the TGZ file is located. In this case the file is usually located at /usr/src/packages/SOURCES.

To get a binary RPM out of the source RPM, do the following:

```
# mkdir $HOME/bin
```

### ***Build from TGZ file***

### ***Installing a binary package***

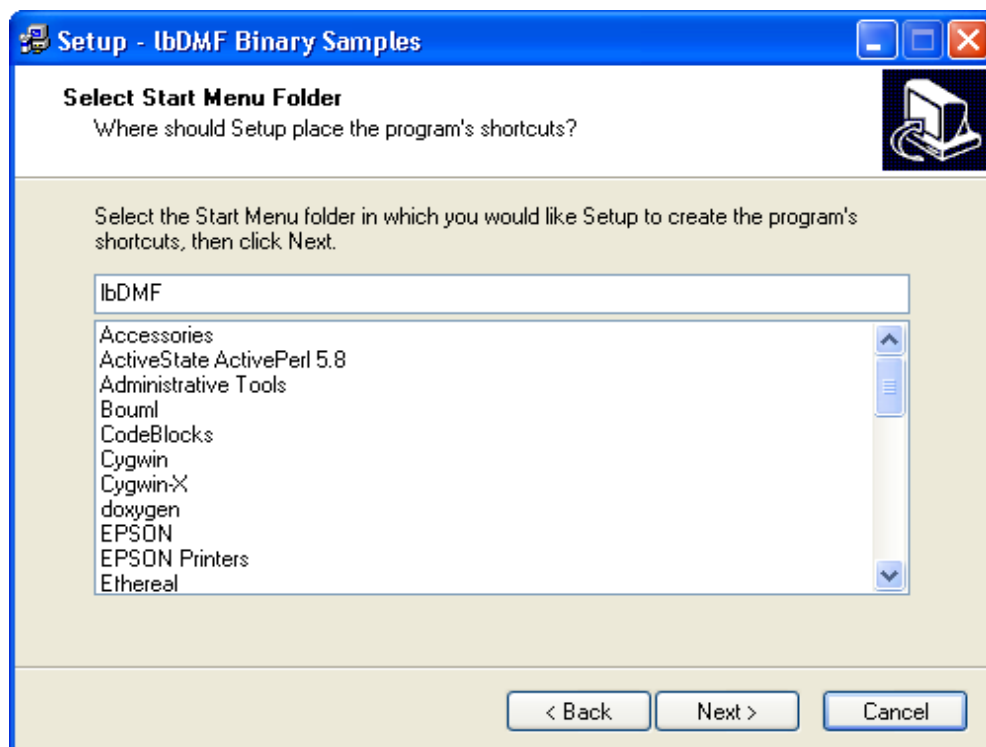
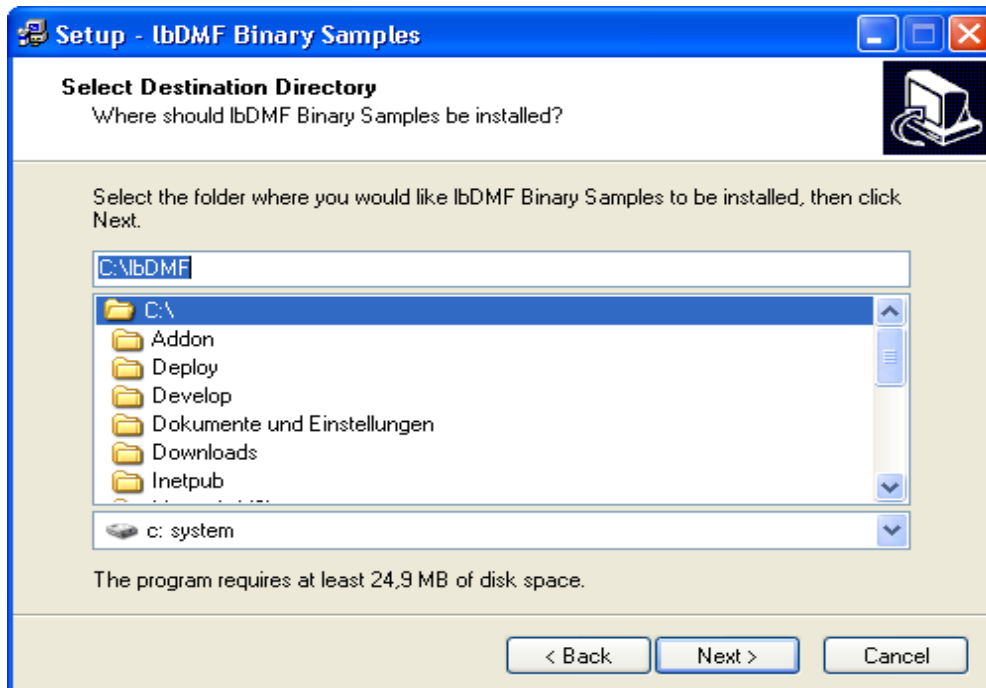
To install a binary package you also need to check the prerequisites on the intended OS. I have made some packages independent to other packages so you only need to install that package. But there is always a dependency to the database and ODBC drivers.

Before you install the client, please check [Installing the database](#), [Configuring the database at first time](#) and [Setup ODBC configuration](#) to see, if the database is available.

Download the latest binary version of the [software](#) for your OS and jump in the corresponding chapter.

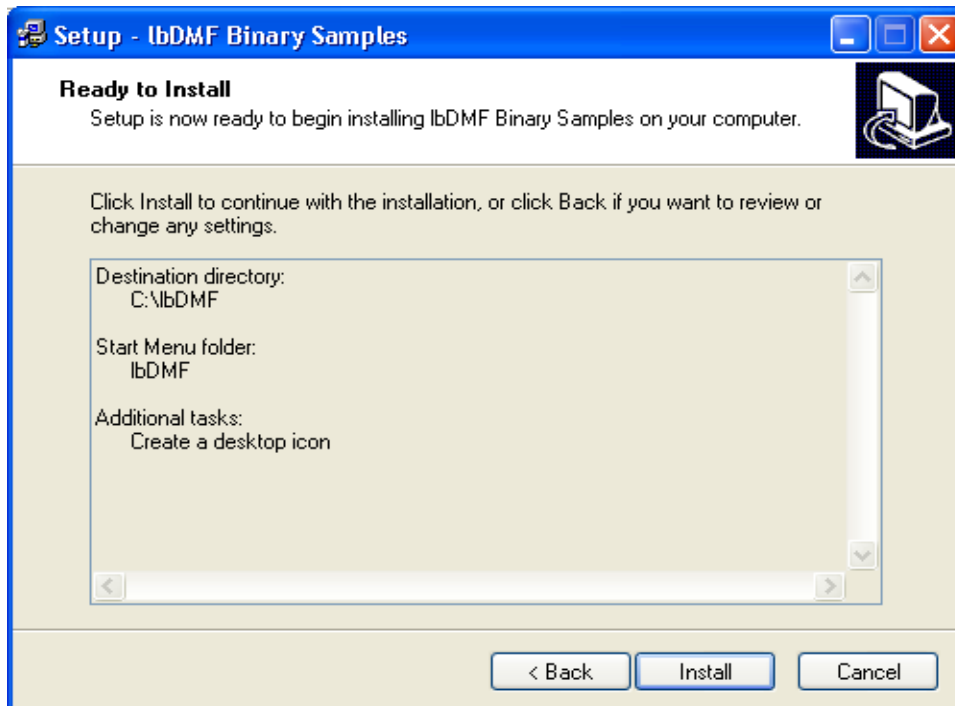
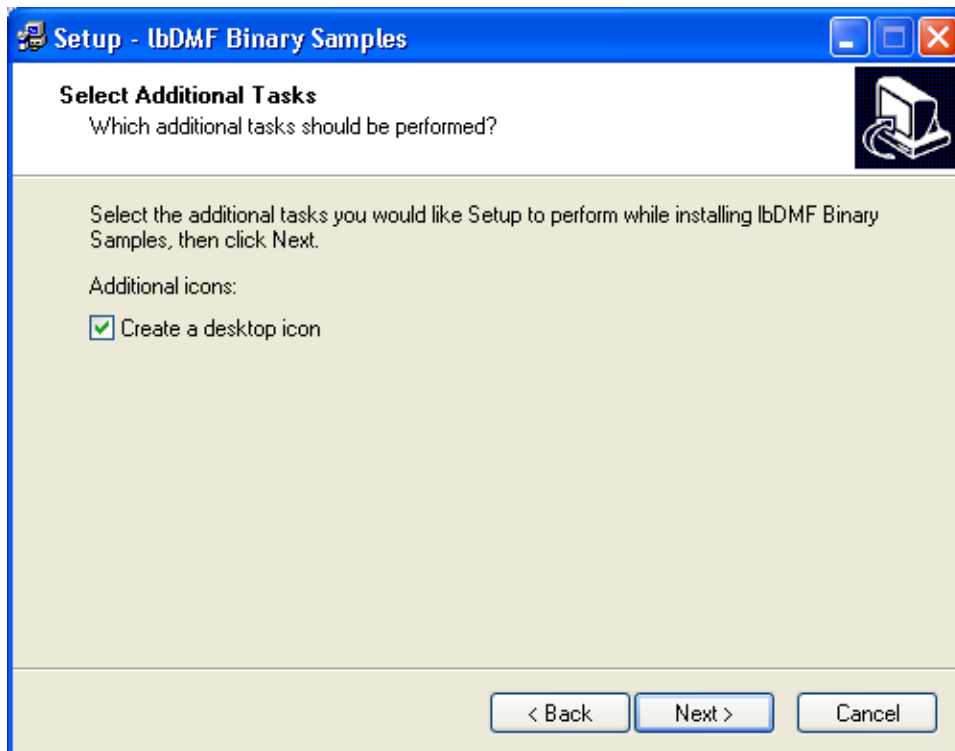
## Installing on Windows

Execute IbDMF-BinSamples- 1.0rc1.exe and click to 'Ja'. Then click on 'Next >'. Here you will be asked for the installation location. Leave this with the default value if you not have any reason to change this.



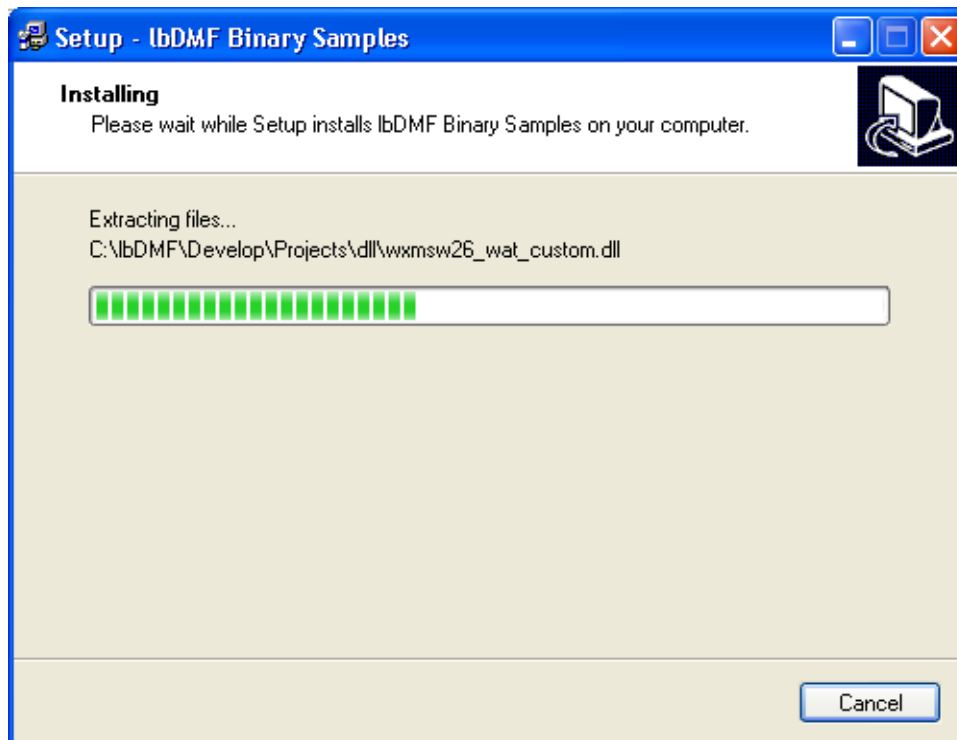


If you don't like the desktop icon, deactivate it.

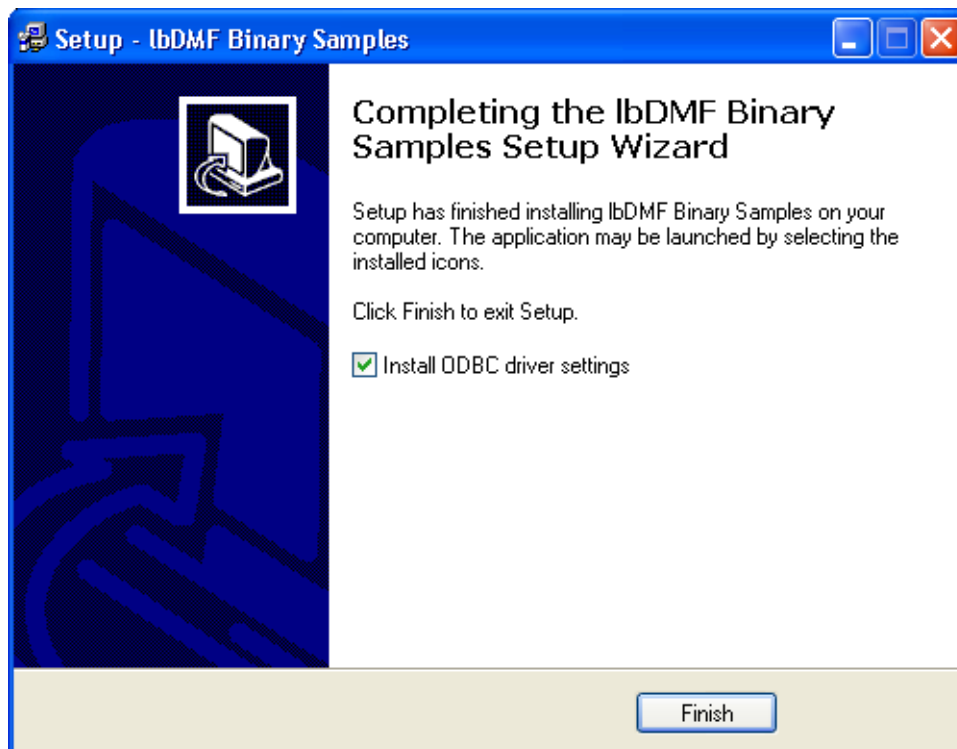


After you have installed the software, you will get asked to install the including ODBC driver. This creates ODBC settings for a database on the same machine. The driver files will always installed and the ODBC settings could be installed after this by executing the installODBC.bat file.


You see here, that the wxWidgets library is included in the package.

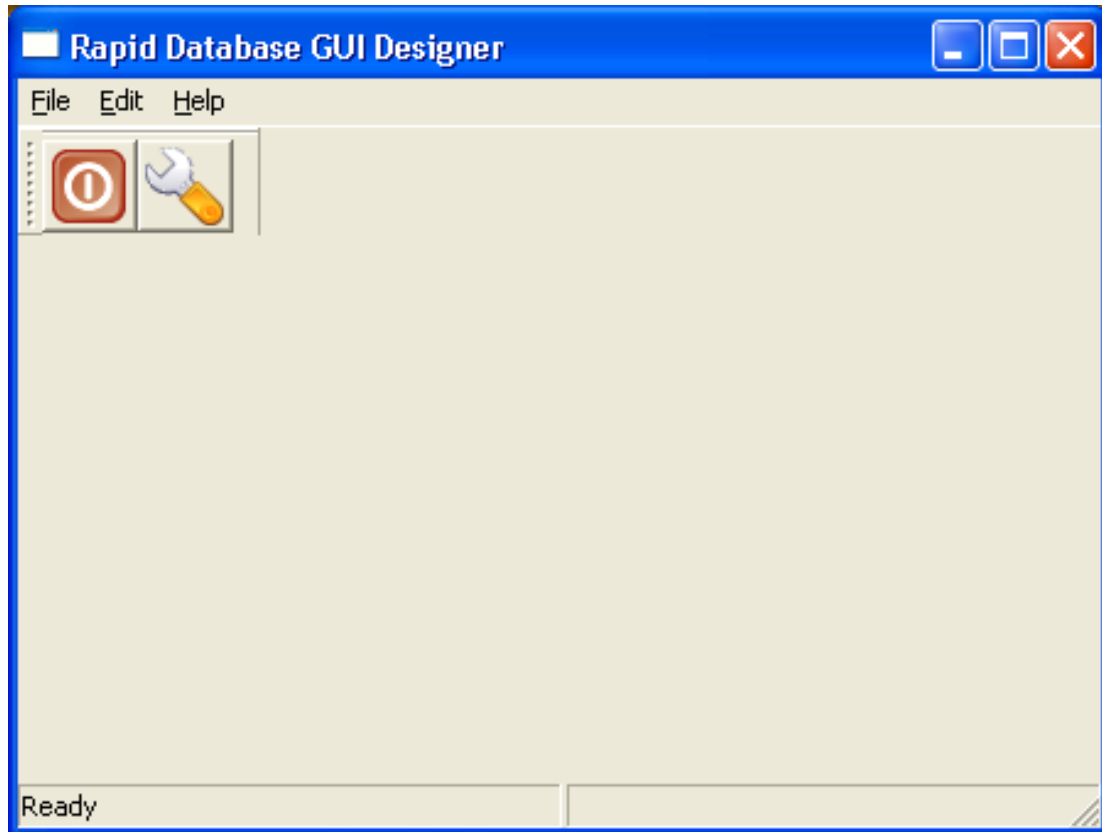


Deactivate install ODBC driver, if you already had an installation.



Now you may start the application by clicking on the following icon:

 At the first time you will see the following image where you not automatically be logged into an application:



## Installing on Linux

Installing on linux depends on the Linux distributor and the Version. I have tested SuSE Linux 9.1 yet. The same RPM package does not install on the openSuSE 10.1 version, because libexpat is newer and the package depends on the older one. So here you should prefer the source installation in TGZ format and follow the corresponding chapter for that. Therefore [Preparing to build the source distribution](#), [Building wxWidgets on Linux from source TGZ](#) and [Build from TGZ file](#) would be interesting for you.

After the installation you will see the application similar to the Windows version. Also you are not automatically logged into an application.

## Installing on Mac OS X

Todo.

## Installing on Solaris

There is not yet a binary installation package. Follow [Prepare on Solaris](#) before.

Please refer to [Installing TGZ distribution](#) to get a binary installation out of sourcecode.