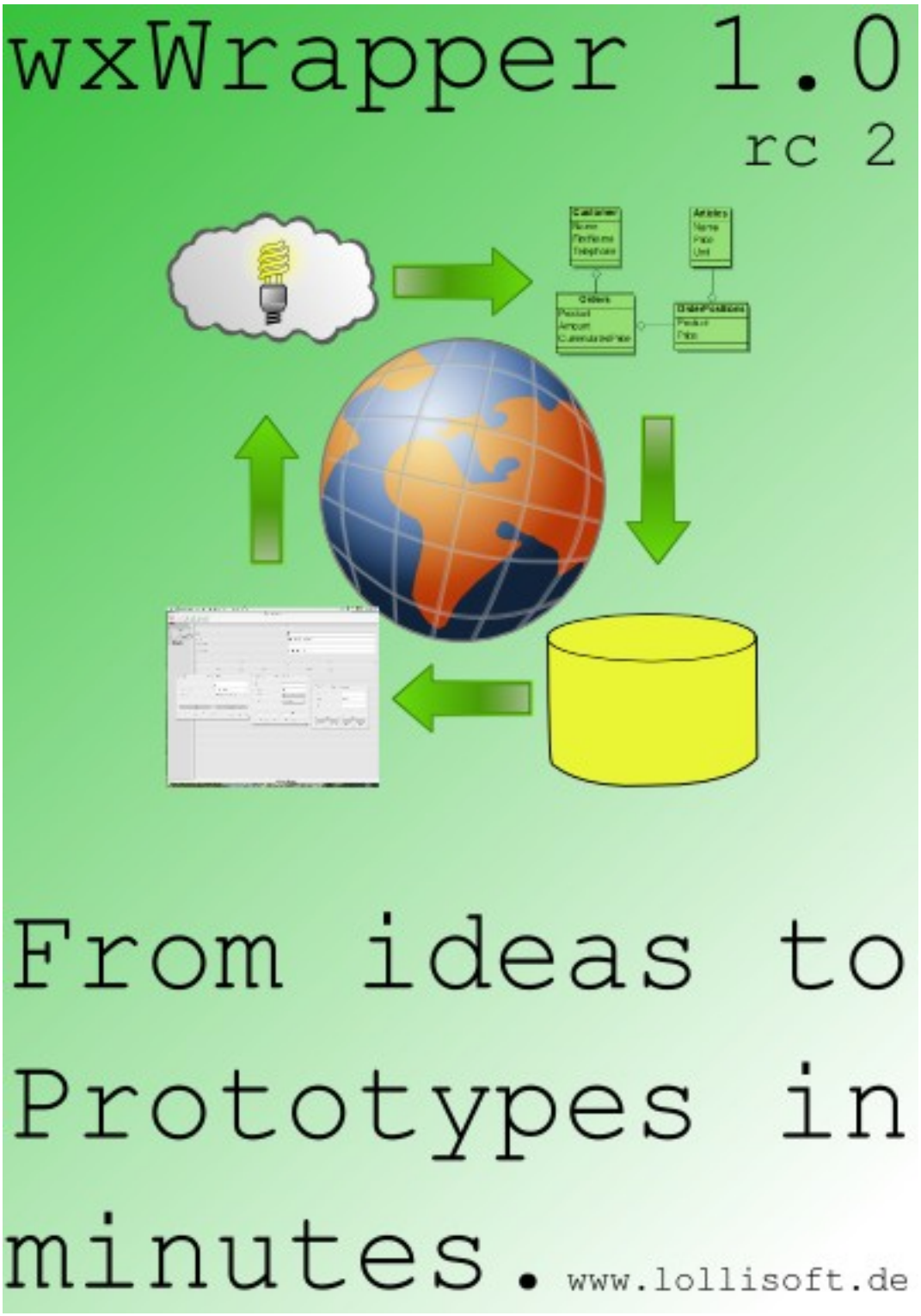


Fast Application Prototyping

driven by wxWrapper and its lbDMF framework in version 1.0rc2

Fast Application Prototyping

Documentation

Creating a database prototype in minutes

$Revision: 1.3 $

Table of contents

Introducion

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 exsists.

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

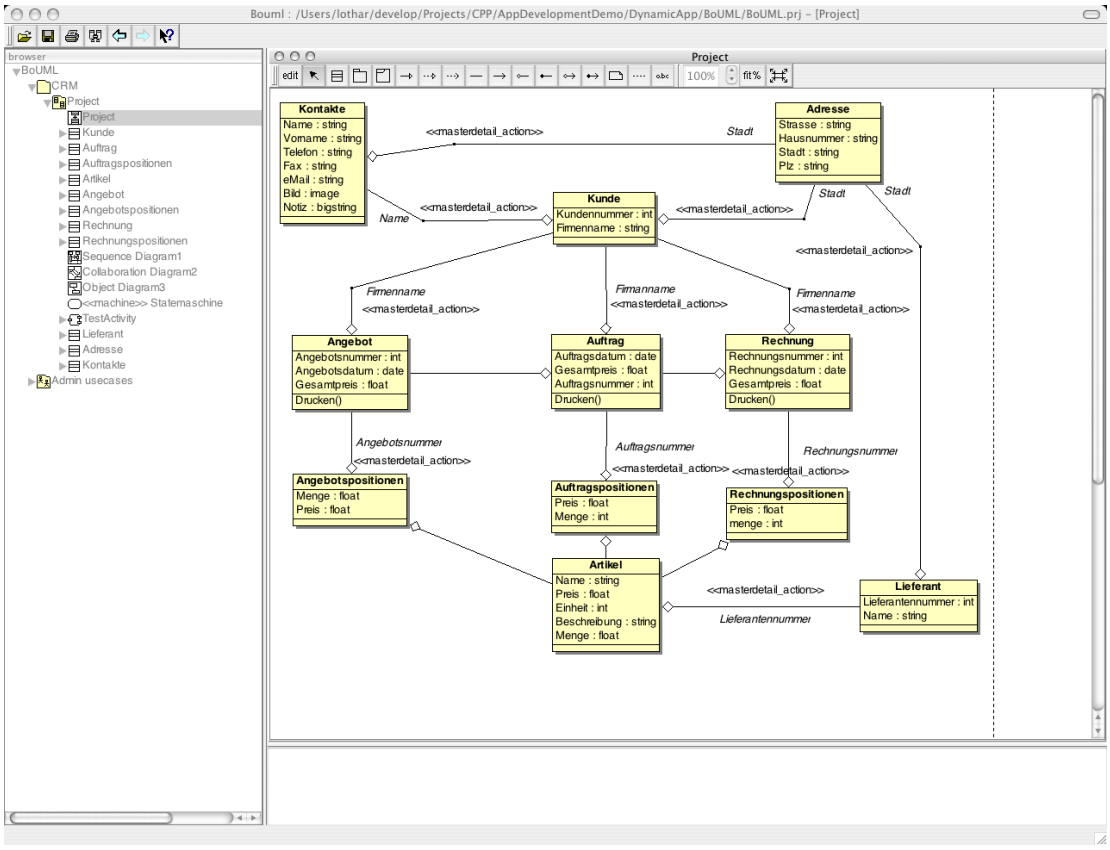
Quickstart

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.

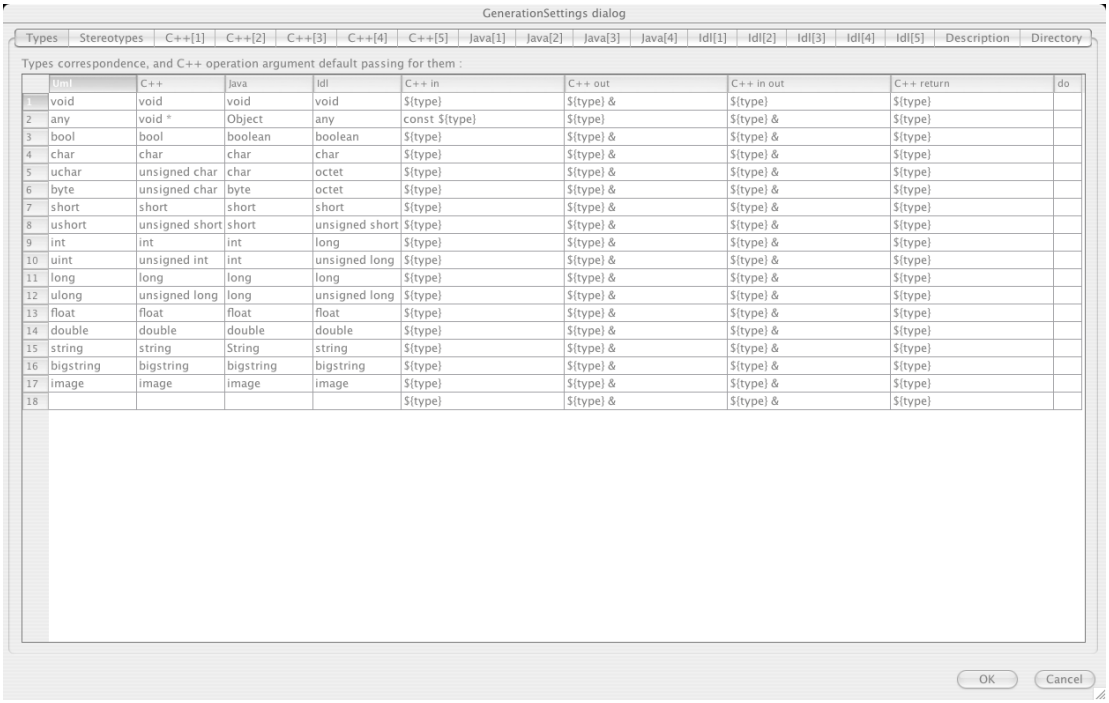
Prepare UML usage

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:

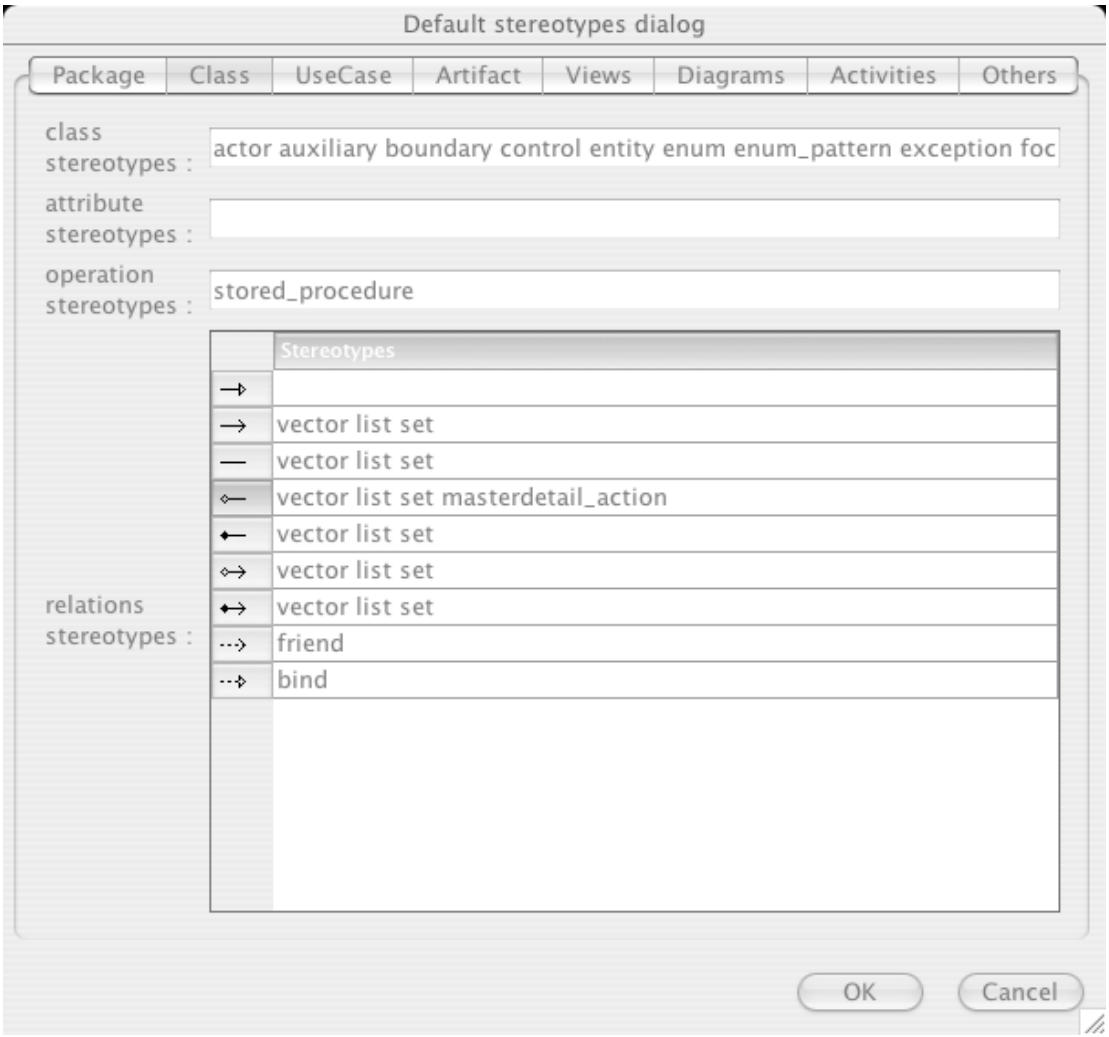
Type bigstring: The bigstring datatype is used to store large text into a database table. Use this for your memofields.

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.

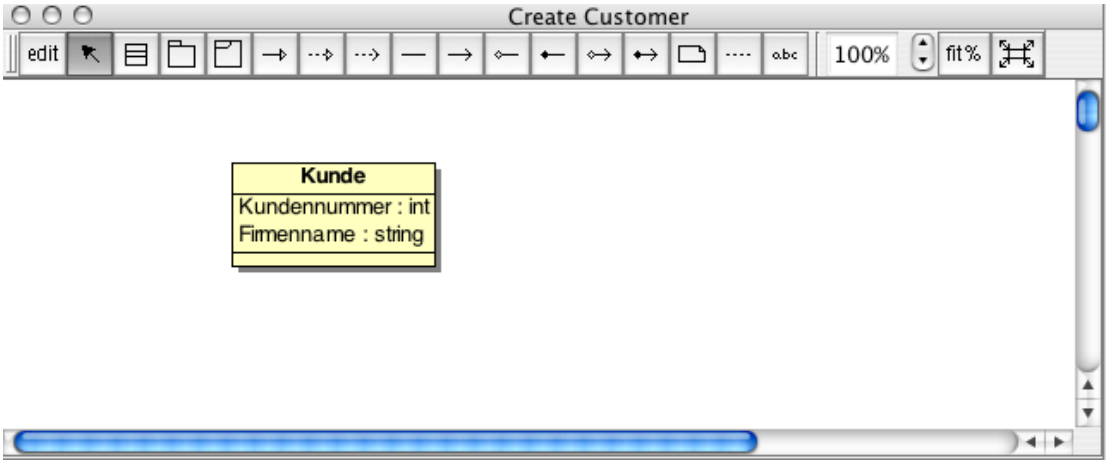
Creating the first UML classes

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.

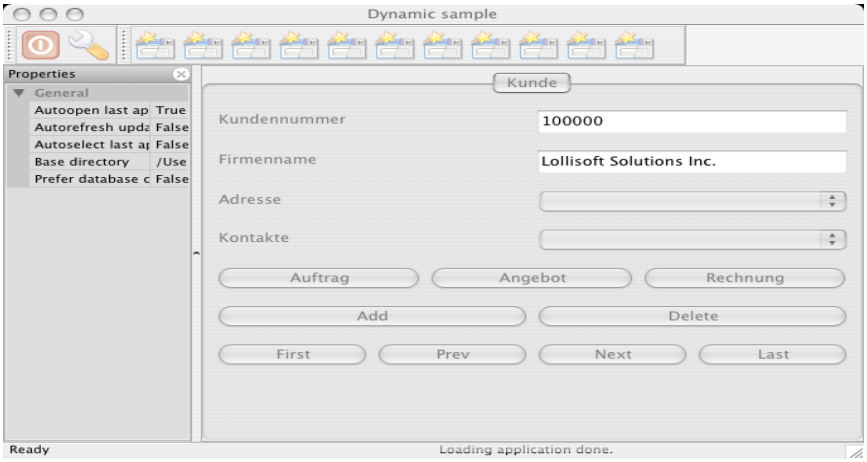
To show this, I will create separate class view, because I still have the model. You could use the main project's classview (Project). It takes no matter how the class view name looks like. I do not document the complete UML diagram because you will get it from the project.

Add the customer class to the model

The customer class looks like this (Picture 4):

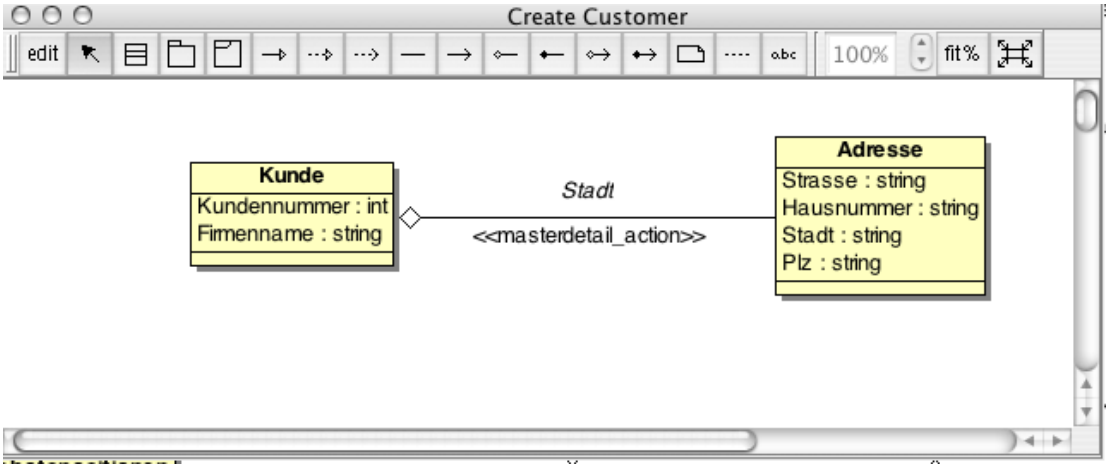


You see, that I do not add any address related data to this class. This is because an employee may also located at any address. Therefore we will add a separate class for the address (Adresse). The customer mask looks like here (picture 8, ignore the dropdown controls):



Add the address class, aggregate to customer

The address and an aggregate (Picture 5):



Here you see the new class Adresse. It is yet 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.

Currently I could only use aggregates for an application prototype. This is because the way I import the UML model described later.

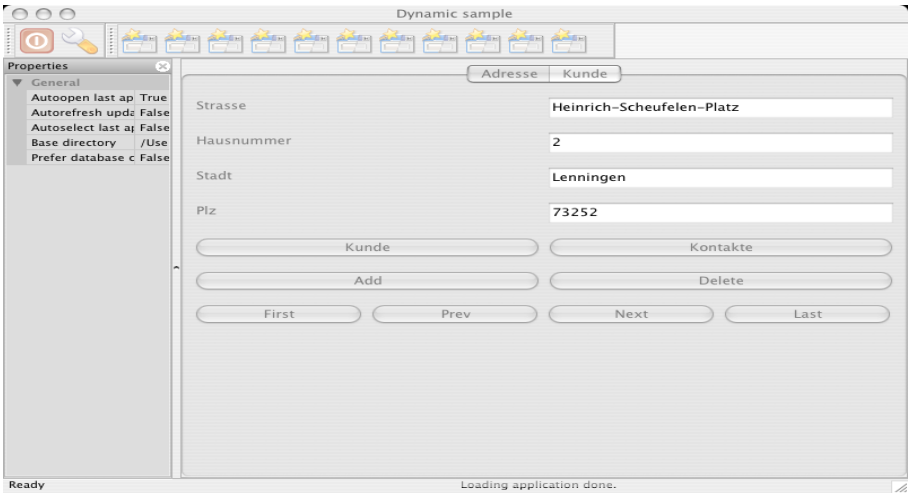
Here I will explain the details on aggregations. On the given sample you will see that the aggregate has an associated stereotype of masterdetail_action. This is important for the later navigation between the database forms. Here you will be able to open the customers located at a specific address.

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.

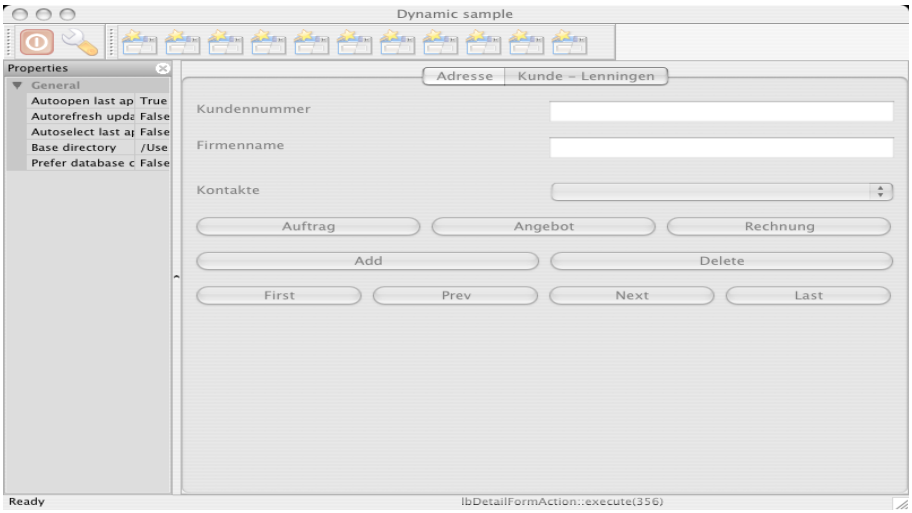
To enter a full customer entry with an address, you first need to enter the address and then save it by navigating once next and back. I do not have a save button yet. Then open the action 'Kunde' from the selected address.

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):

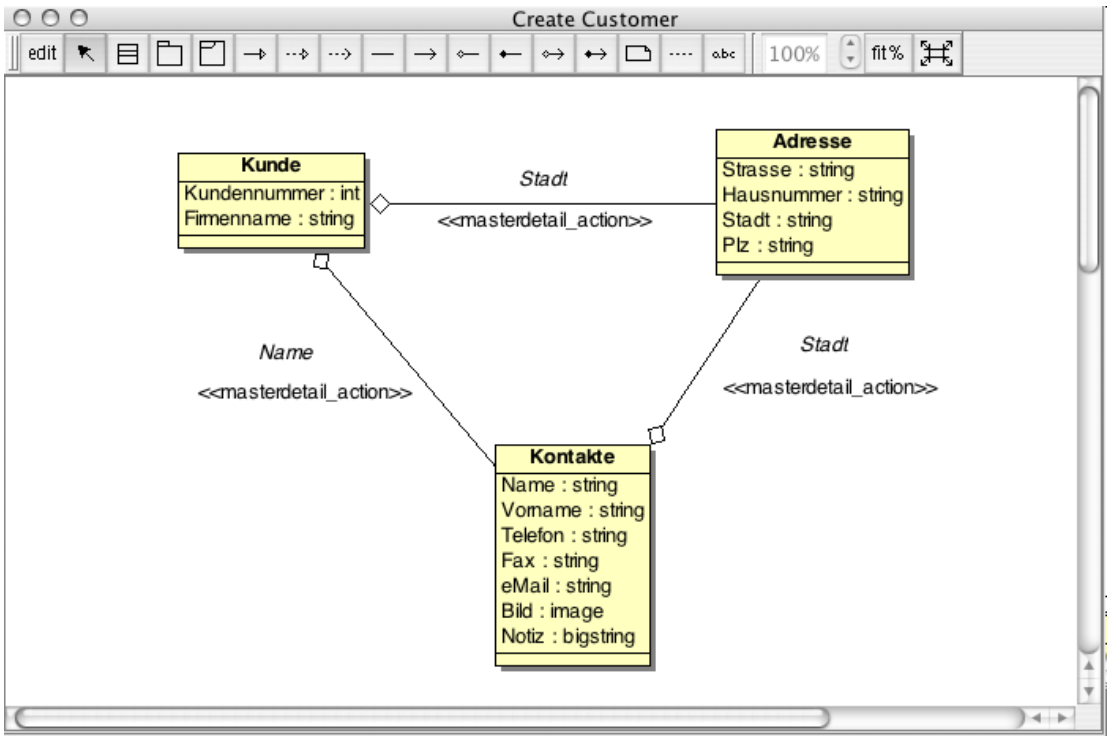


There is no data because you have newly created an address. You will see that there is no Address dropdown control. Compare to the picture in adding the customer class. By the way, you see the contacts control. This is there because I have added a contacts class and also assotiotiated it to the customer.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

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.

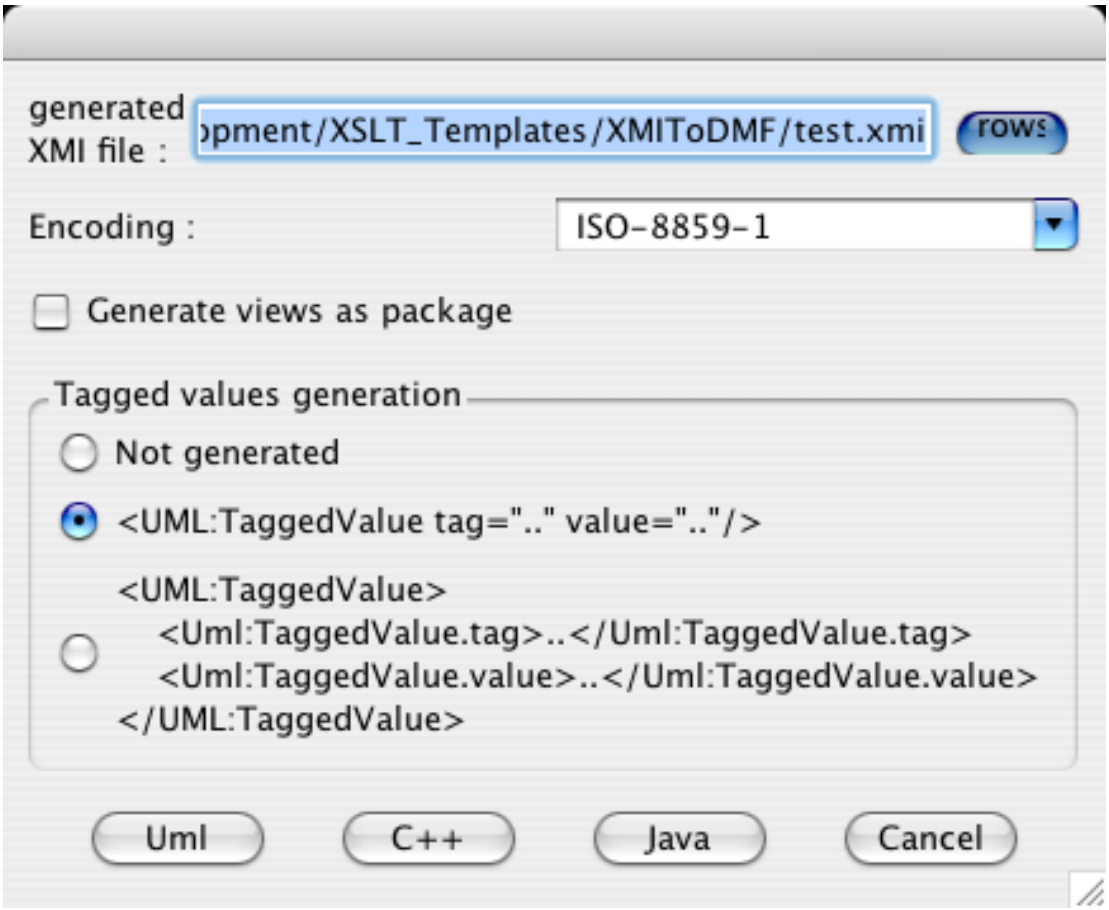
You may open contacts or customers from a given address. The aggregate has in both situations the master link on 'Adresse'. Now you will understand the picture Picture 6

What you also will see – after some customers have been entered – are the address and contact fields showing the town (Stadt) and surename (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)

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

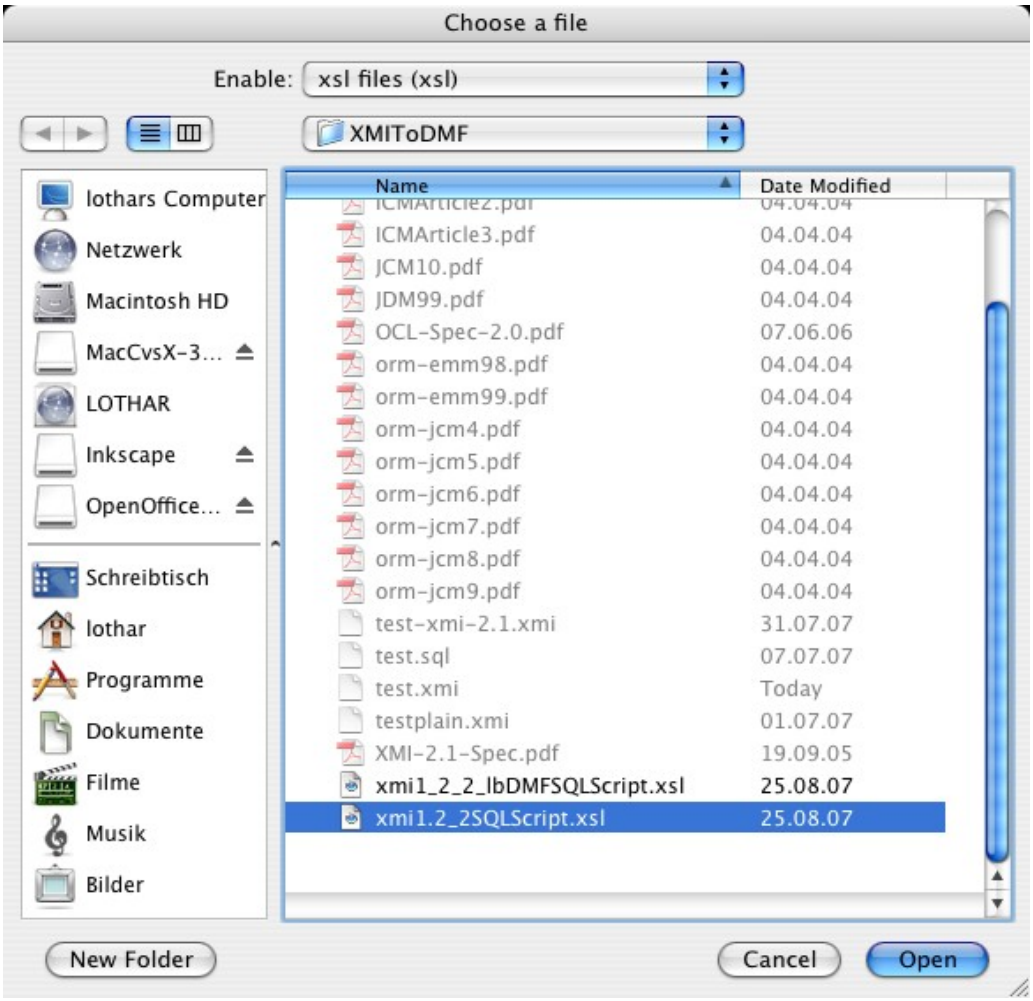
Before you import any XMI file, be sure you not yet have imported that file before. If so have a look at Deleting an application definition

To import an application definition, start wxWrapper, login with the default user 'user' using 'TestUser' as password. Then select the application 'lbDMF 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

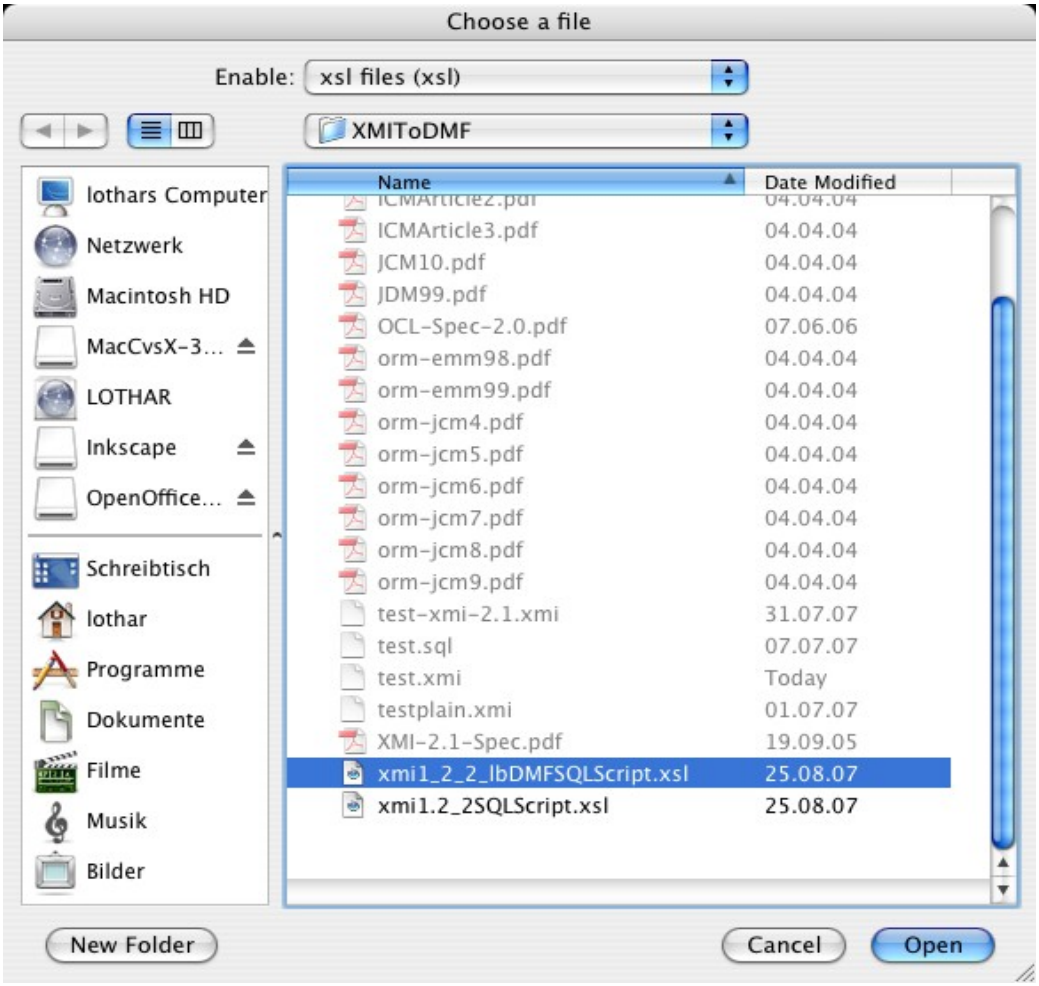
If 'Yes' was selected, you must choose my predefined template as of the picture 12:



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 logon to a different application.

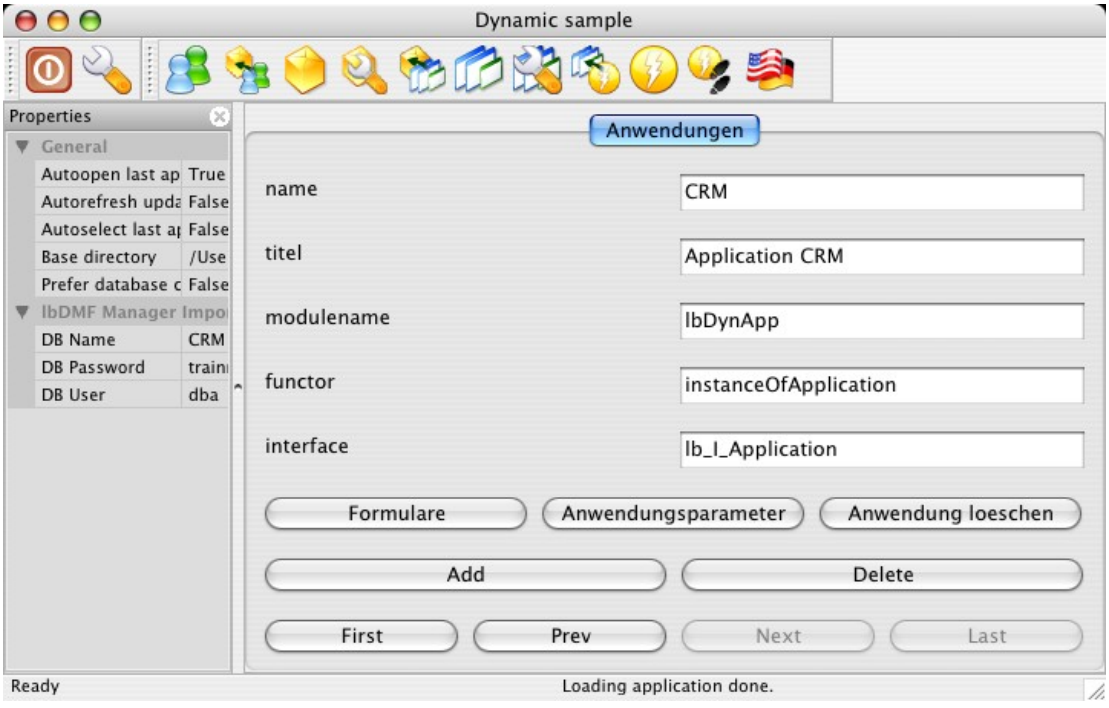
Deleting an application definition

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 'lbDMF Manager' and proceed with opening the form 'Anwendungen' in menu 'lbDMF Manager' or clicking the yellow box (third from left). Select the intented application and press 'Delete'.

Picture 11 shows how it looks like:



Installing the database

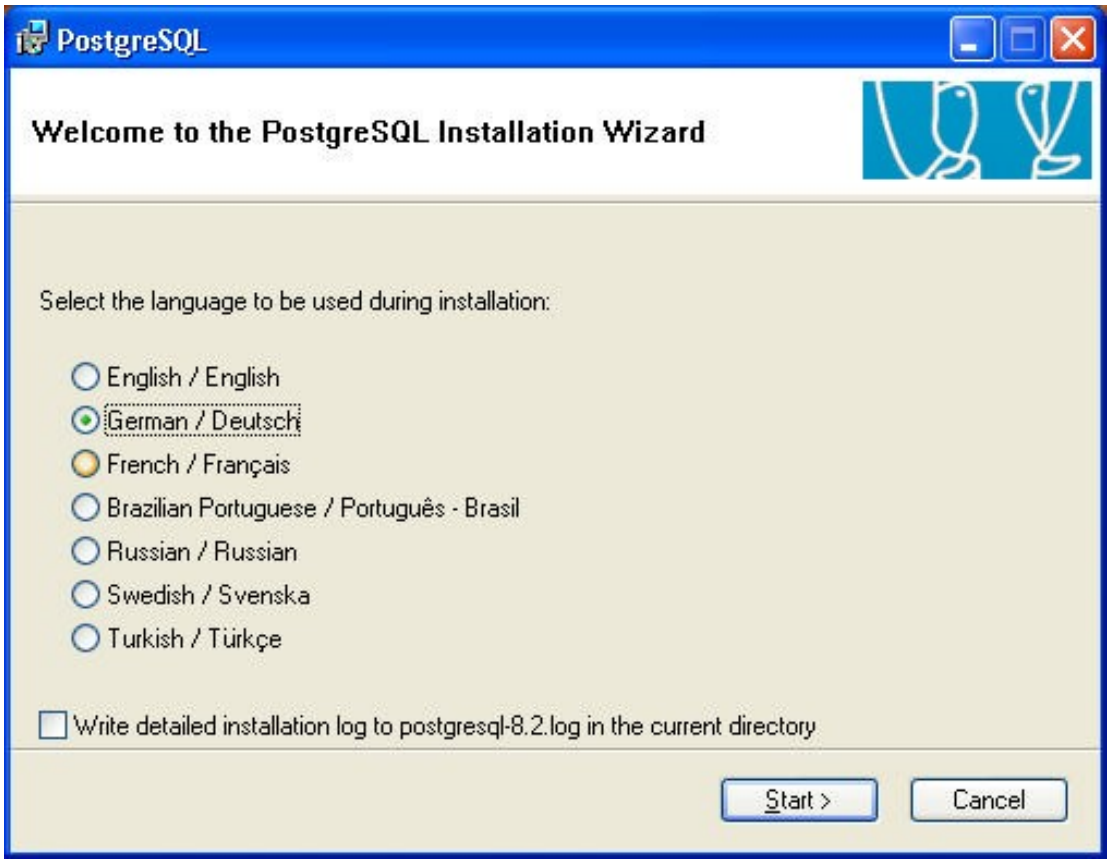
This chapter is divided to the different platforms I do currently support. Before you proceed, you have to decide what database you like to use.

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.

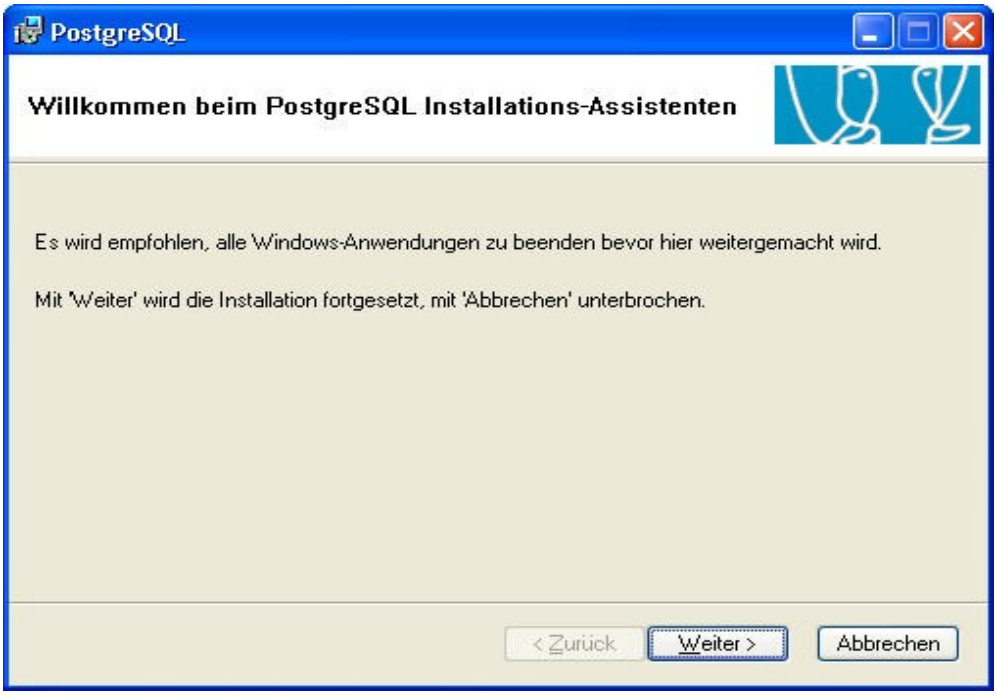
Installing on Windows

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):

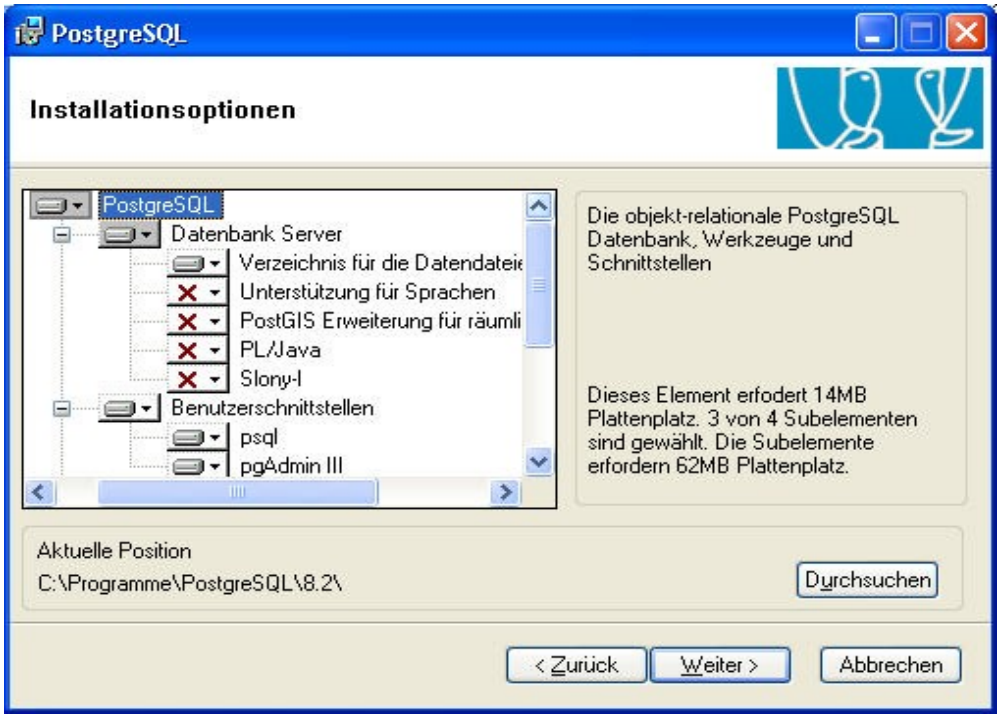


Close all other applications (Picture 15):



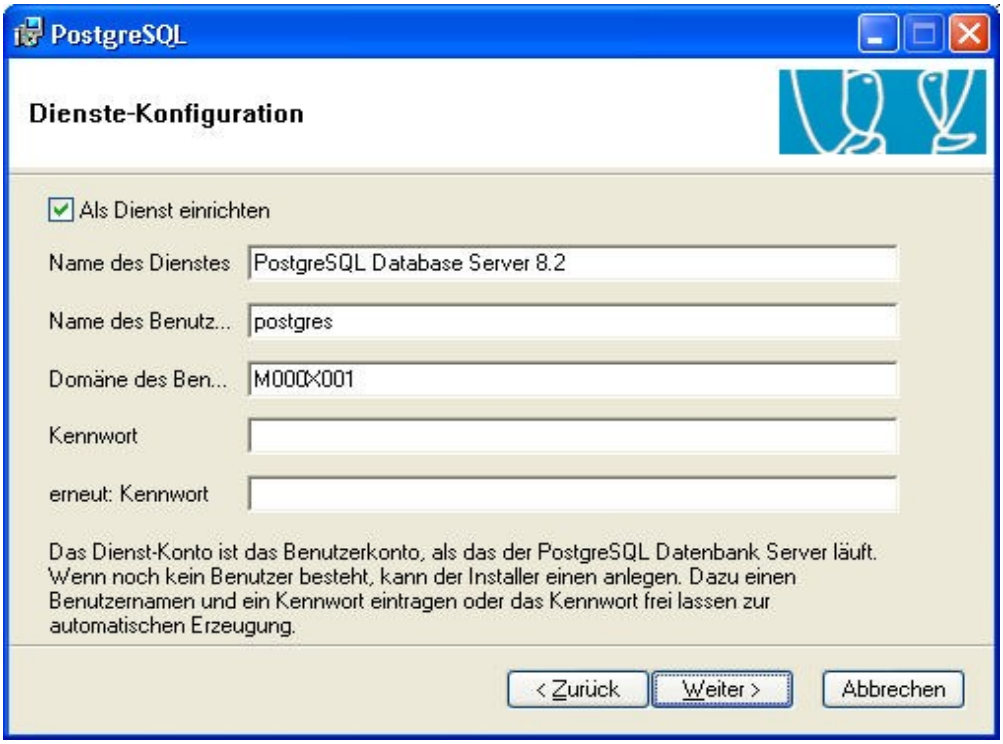
Some notes and licensing issues (Picture 16):

Leave the default settings (Picture 17):

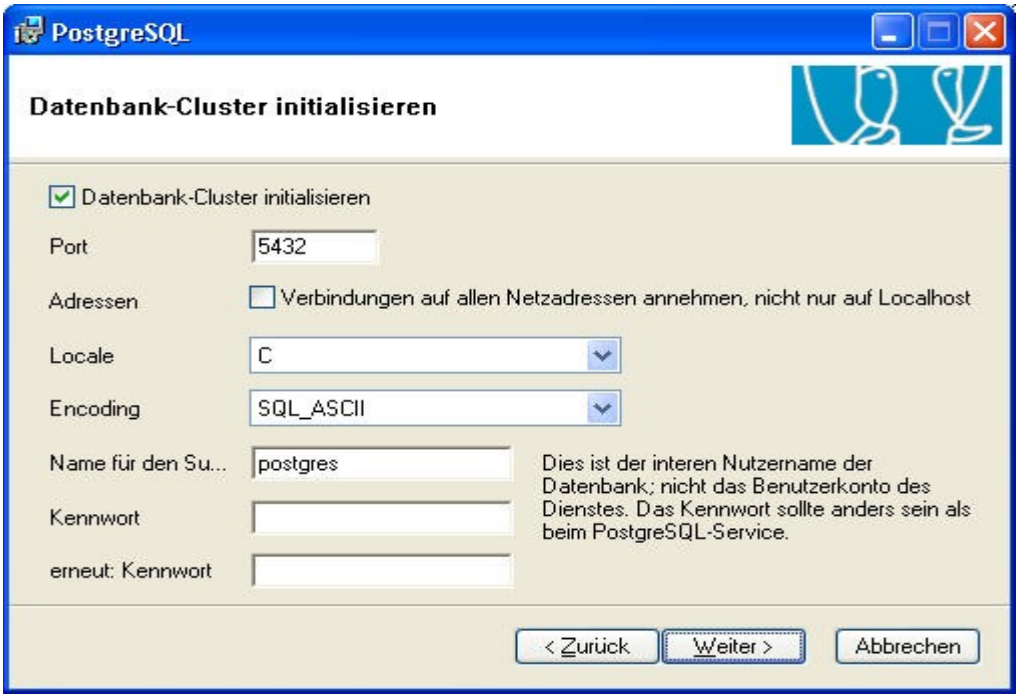


Note: You must at least install an ODBC driver and the database itself.

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



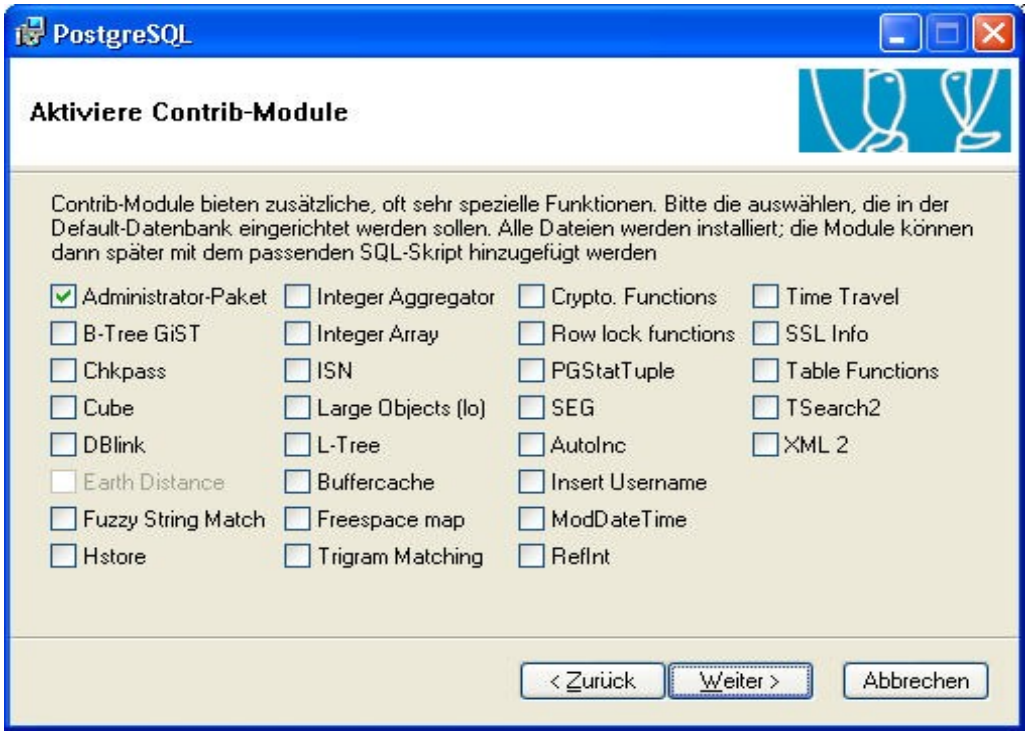
Setup some global database settings (Picture 19):



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

Leave these settings (Picture 20):

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

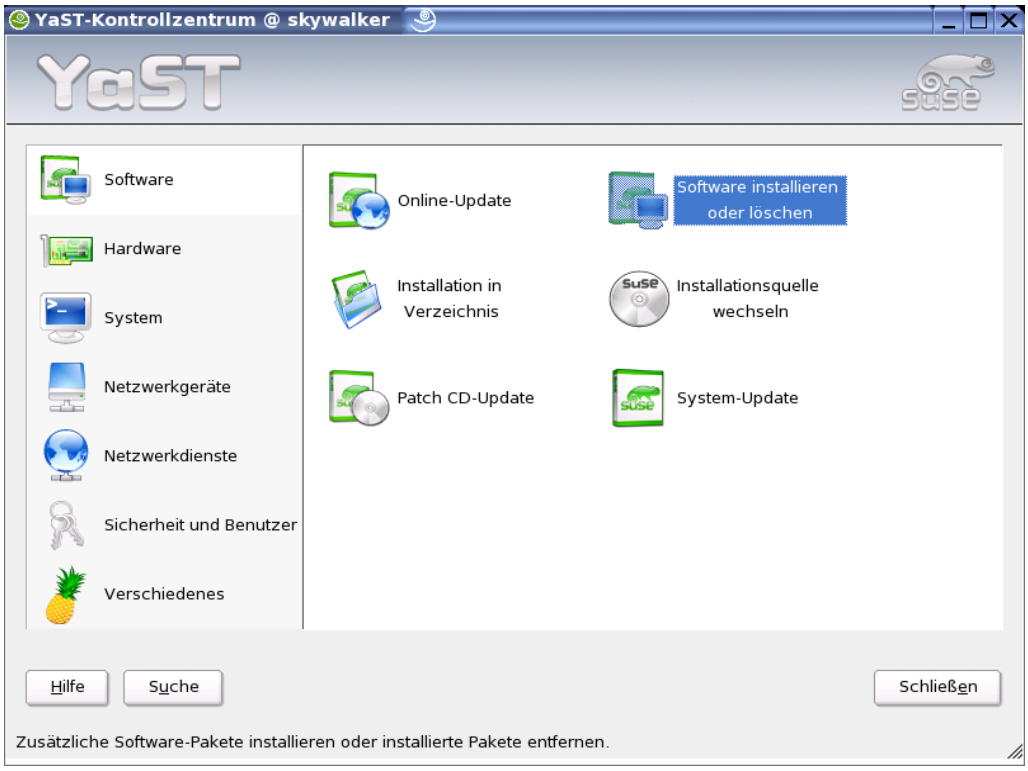


Installing on Linux

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

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

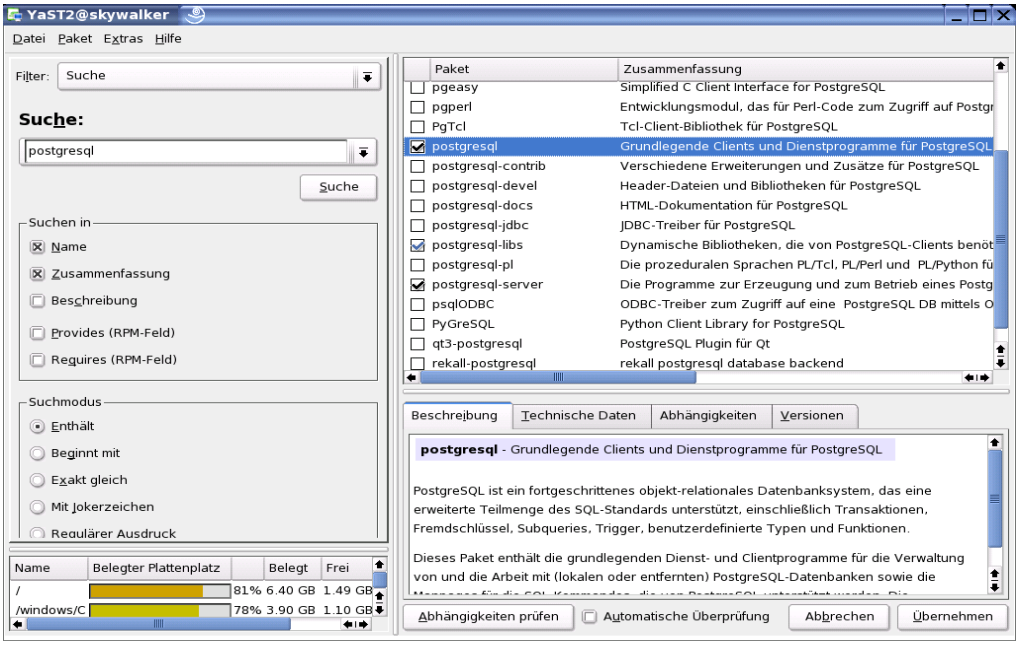
First of all you need to start Yast. I use the GUI variant. Select the following as shown:



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.

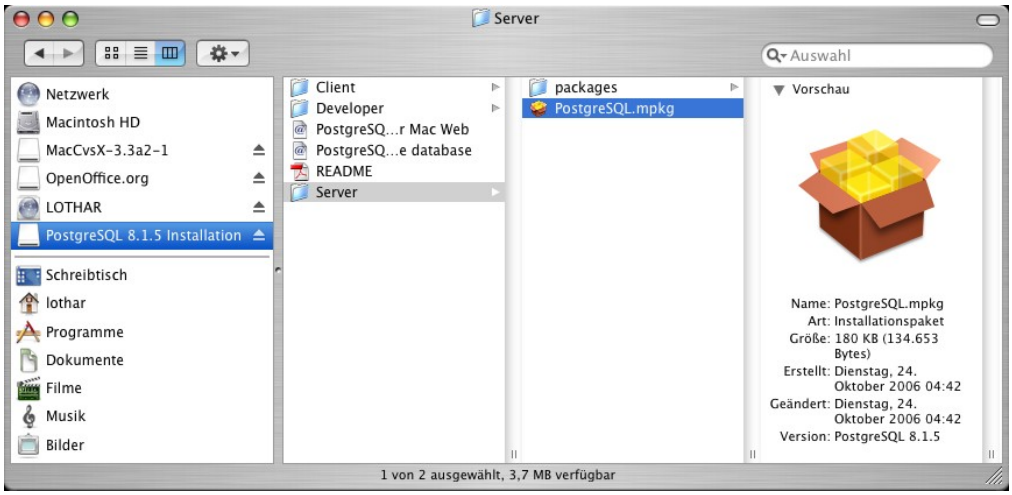
A sample may be this line:

host all all 192.168.0.0 255.255.255.0 trust

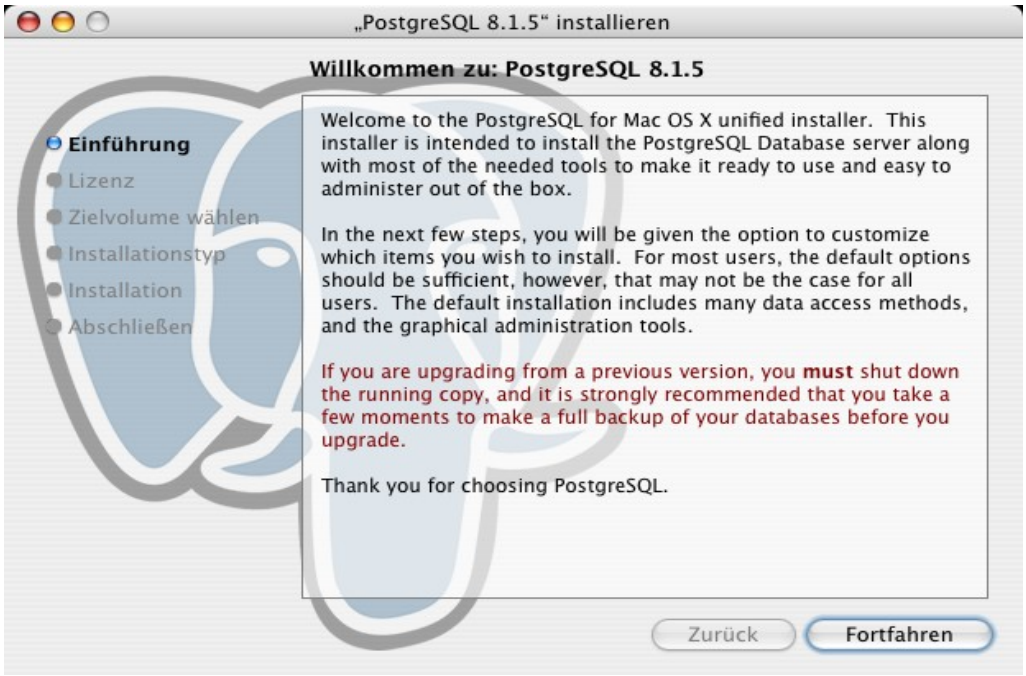
Installing on Mac OS X

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

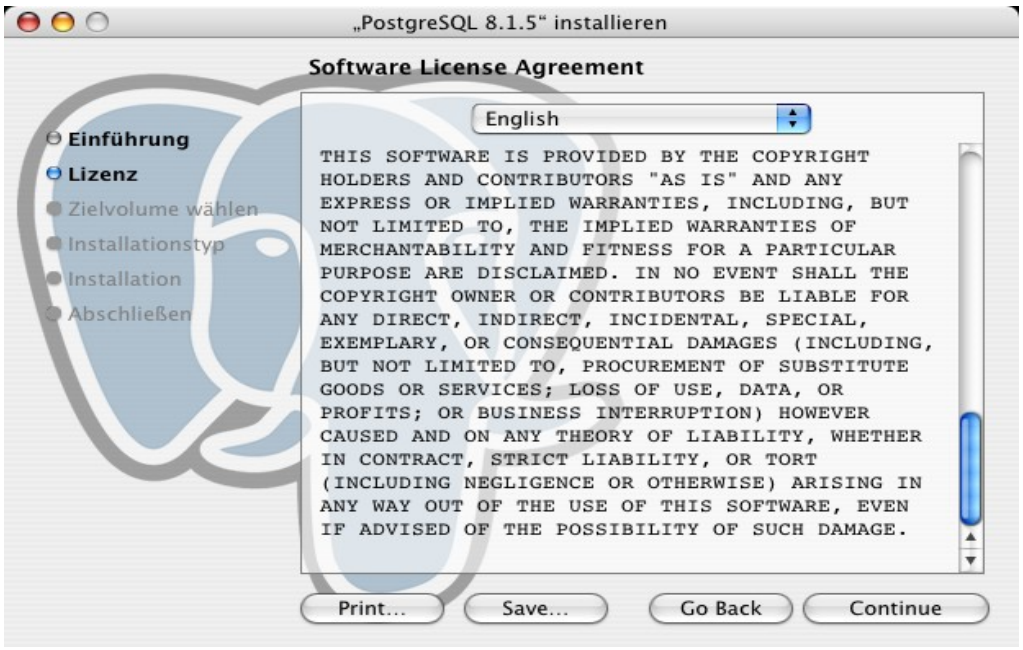
Mount the package if not yet done and execute the server installer:



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

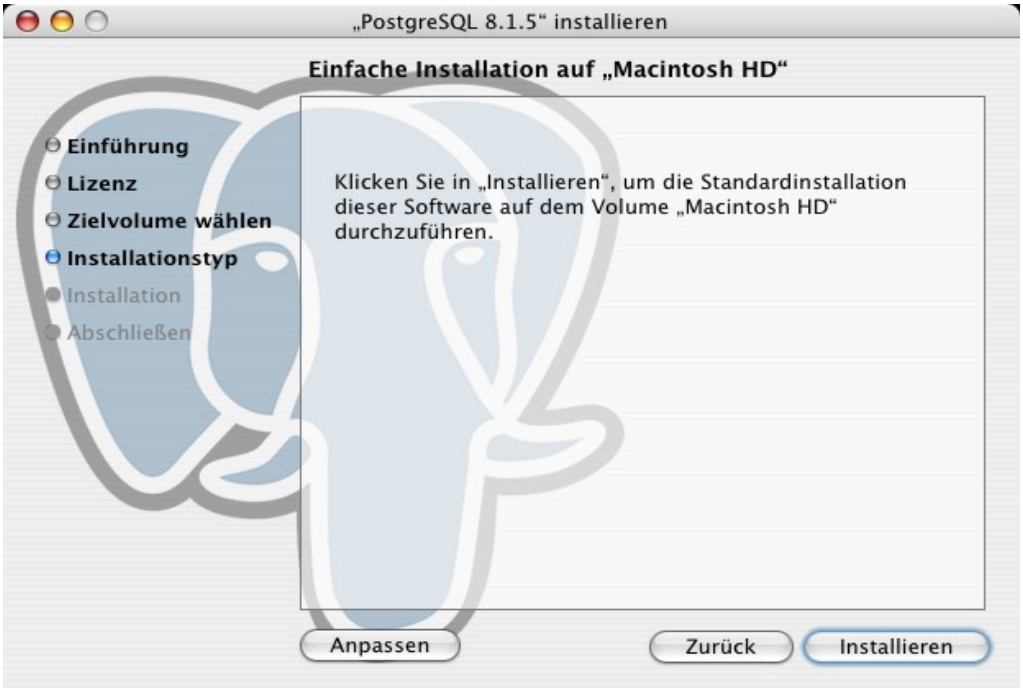


License stuff:



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 a source tgz file and follow the typical UNIX make commands.

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

Then follow the Linux installation guide. It may be the same here.

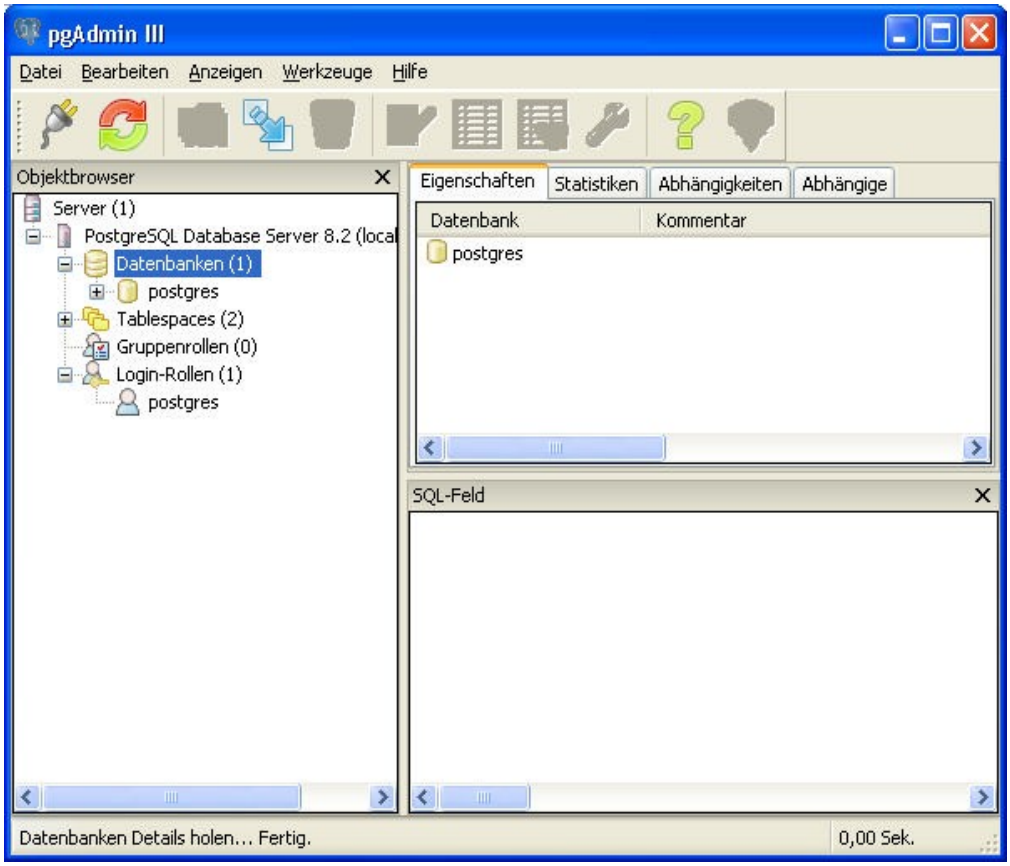
Configuring the database at first time

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.

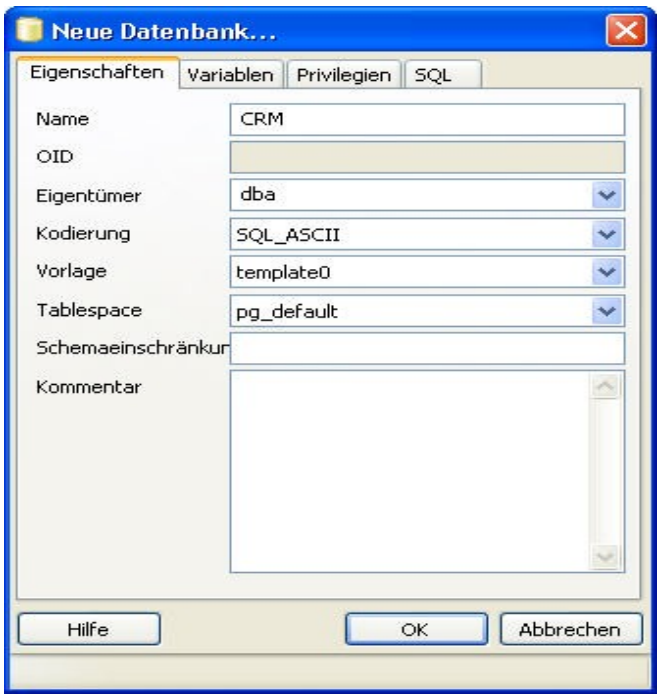
After you have the database user you create the following two databases:

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

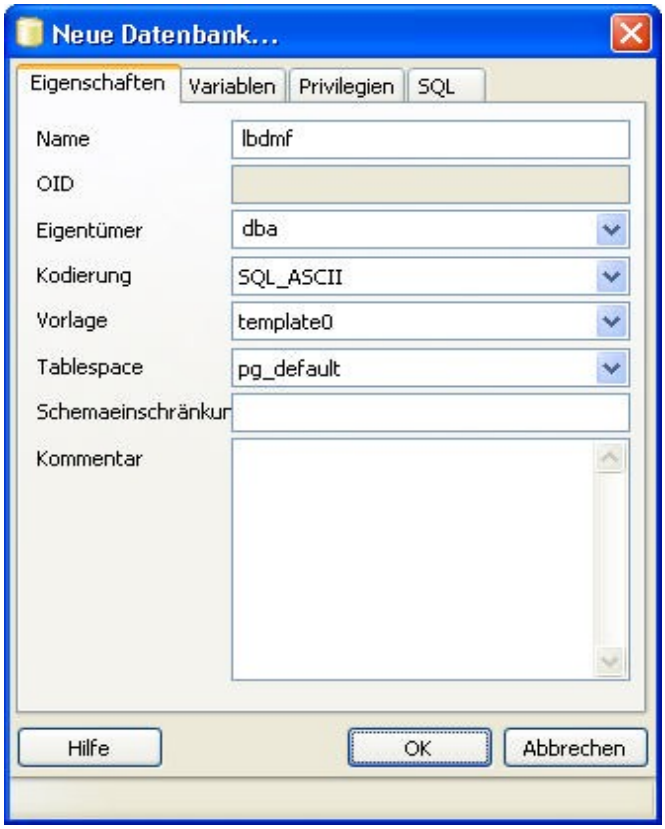
Database lbdmf: As the system database for the application configurations.

Enter the role name as shown (Picture 23):

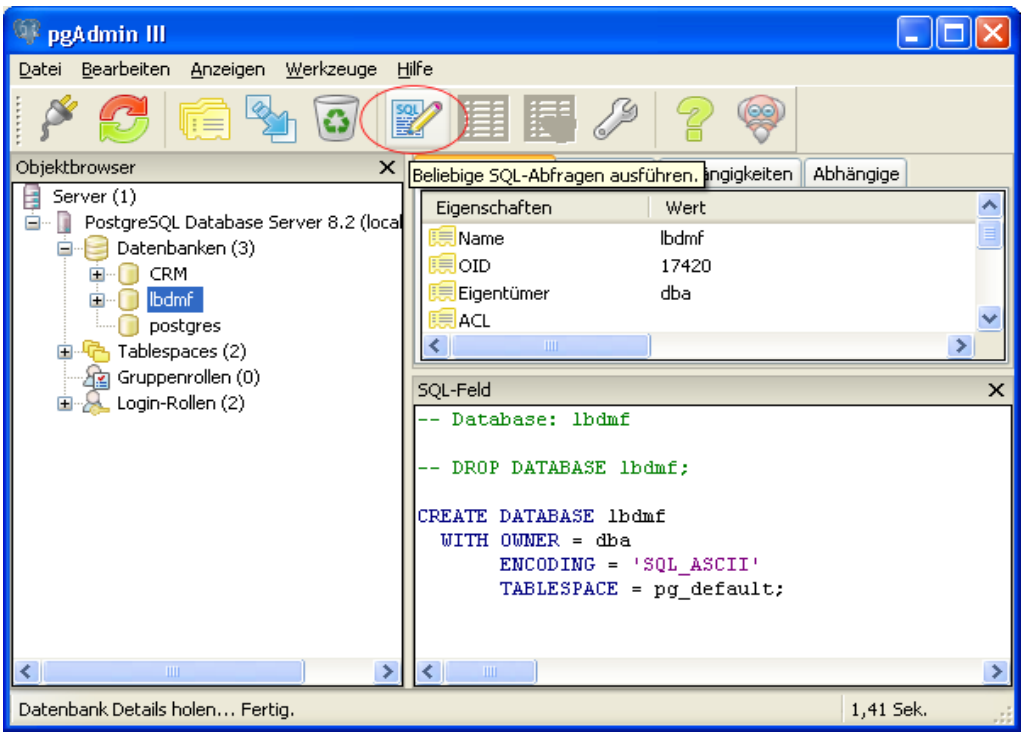
On the Datenbanken tree node (databases), you need to right click and select create new database. (Neue Datenbank...) (Picture 24):



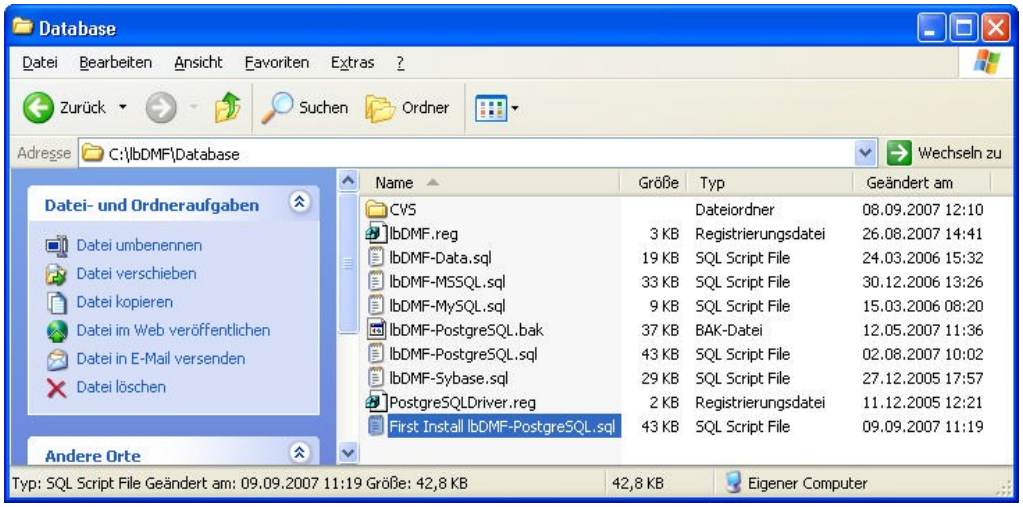
Settings for the system database (Picture 25):



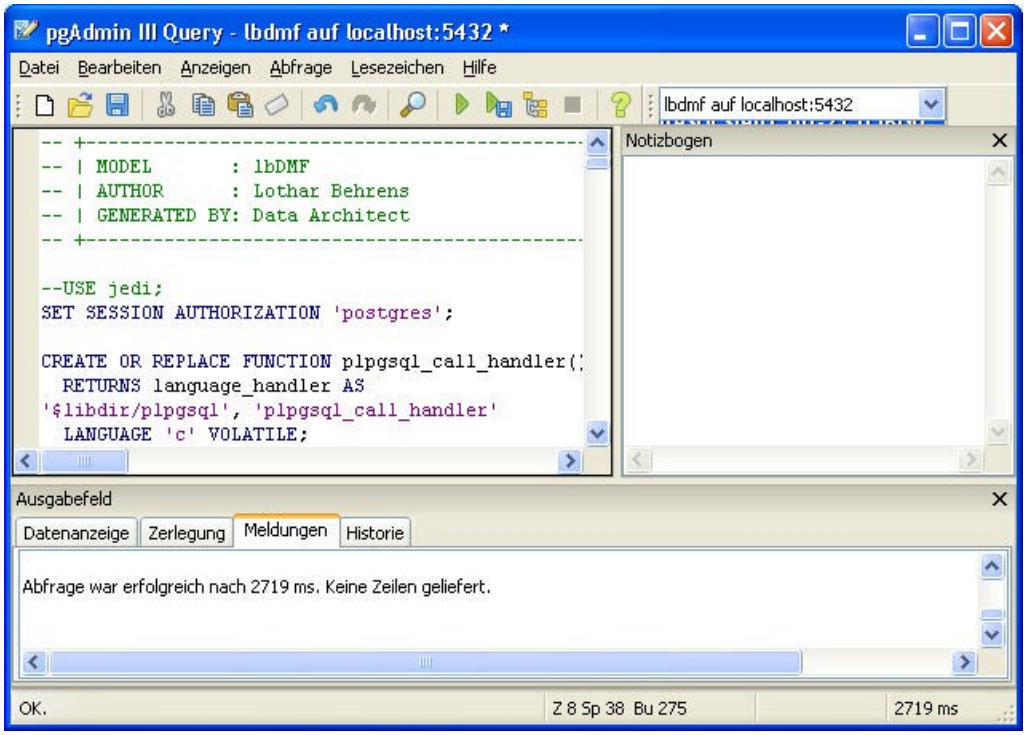
After you have created the databases, select the system database (lbdmf) and open the SQL window (Picture 26):



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 expect 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

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.

In my sample the driver is installed as seen in the 'Driver' line on the next page.

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        = trainres
Database        = lbdmf
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.

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.

Configuring on Solaris

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