Puzzle ITC GmbH
Maulbeerstrasse 15
CH-3011 Bern
Switzerland

Phone: +41 (0)31 398 44 77
Fax: +41 (0)31 398 44 78

www.puzzle.ch

openDBcopy - User Manual

http://opendbcopy.org

openDBcopy is an Open Source Project hosted at SourceForge
http://www.sourceforge.net/projects/opendbcopy/

Release	0.51
Date	19.07.04
Reference	user-manual-0.51

© 2004 by Anthony Smith, Puzzle ITC GmbH – all rights reserved.

ch20040728w9517

Document History

Date	Version	Person	Comments
19/07/04	0.51	A. Smith	First issue
22/07/04	0.51	A. Smith	Review and corrections, new screenshots

Table of Contents

1 Overview

openDBcopy must be understood as a framework to configure and execute plugins. A plugin can do ANYTHING. Some plugins are database specific, others provide features to do file manipulations and transfers etc. There is no limitation of what a plugin can or could do. To develop your own plugin, which is very simple, have a look at the developer's manual.

Executing a plugin or series of plugins can be done either via the Graphical User Interface or directly from the command line as a batch job providing an xml file with the required job configuration.

This document describes the core features and usage of available plugins.

openDBcopy is an open source project by Anthony Smith, Puzzle ITC GmbH. The project is hosted at Sourceforge and published under the terms of the GNU General Public License.

Please use the public forum to post questions. There are also links to request new features and track bugs.

Name	Website
openDBcopy at Sourceforge	http://sourceforge.net/projects/opendbcopy/
openDBcopy Download	http://sourceforge.net/project/showfiles.php?group_id=91406
openDBcopy Forum	http://sourceforge.net/forum/?group_id=91406
openDBcopy Website	http://opendbcopy.org http://opendbcopy.ch
GNU General Public License	http://www.gnu.org/licenses/gpl.txt
Puzzle ITC Website	http://www.puzzle.ch (Puzzle provides a complete Sourceforge mirror :-)

2 Software Requirements

openDBcopy requires a Java Runtime Environment. If not yet installed, please download an actual Java Runtime Environment or Developer Kit 1.4 from http://java.sun.com.

OpenDBcopy 0.5 has been tested using J2SDK 1.4.2_03.

An actual JDBC driver for each database system one plans to access.

3 Installing openDBcopy

openDBcopy is distributed using a set-up wizard (IzPack Installer – platform independent). Once successfully downloaded the `opendbcopy-xx-install.jar` can be started by entering the following command in a shell (Command Prompt for Windows users)

Go to the downloaded file's directory.

Enter the following command:

`java -jar opendbcopy-xx-install.jar` (replace xx with current version)

If Java is properly installed, a dialogue pops up asking to select a preferred language for the set-up wizard. Choose between English, German or French. Please note that openDBcopy is currently distributed in English and German only. If you are missing your language please do not hesitate to help translating openDBcopy into your language. All texts are stored in ASCII files and can easily be translated.

On the following screens read the README text and License Agreement. Then choose a file location for installing openDBcopy.

After installation the set-up wizard can create desktop shortcuts if openDBcopy is running on a Windows platform.

Non Windows users please note that **`bin/start.sh`** requires execution right.

Please note that your default URLs to databases are stored in a file `SQLDrivers.xml` within your home directory `<your_home>/.opendbcopy/conf`. This file is not overwritten when installing a new version of openDBcopy.

A note about file locations

openDBcopy, the programme itself, is installed by default in a directory called opendbcopy. Windows users may want to install this directory under their Program Files, Linux and friends under a directory such as `/usr/local/opendbcopy` – as you wish.

Personal configurations, such as default database URL for a certain JDBC driver, User Id, log files, personalised configurations of plugins and jobs are stored in separate folder and files. On first start up openDBcopy checks for a directory called

`.opendbcopy`

within your home directory. If it exists, it checks if the default directories already exist. If so, nothing further happens. If such a directory structure does not yet exist, openDBcopy creates such and copies a personal `SQLDrivers.xml` file into `.opendbcopy/conf`.

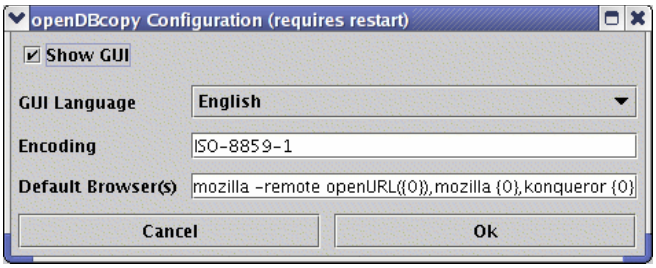
This separation has been done to avoid overwriting of personal and default settings. When updating openDBcopy, your personal directories and files for openDBcopy remain.

4 Before using openDBcopy

Make sure the required database drivers are set in the `CLASSPATH` environment variable or were copied into the lib directory of openDBcopy. When copying the database drivers into the lib directory of openDBcopy, one does not need to add those to the `CLASSPATH` any more, because those drivers are dynamically loaded at start-up of openDBcopy.

5 Configuring openDBcopy

Most configuration parameters can only be changed using the appropriate properties or XML files. Certain parameters, such as GUI language, file encoding etc. can be modified either by GUI or file. The following parameters can be changed by clicking on Show -> Configuration ... from within openDBcopy



When using this GUI to modify openDBcopy's configuration, please restart openDBcopy yourself so that changes can take effect.

The following table lists available configuration files and directories.

Name	Path	Description
conf	opendbcopy/conf	opendbcopy.properties - openDBcopy's main configuration log4j.properties – Logging configuration SQLTypeMapping.xml – SQL Type to Java Type Mapping SQLDrivers.xml – default JDBC driver pre-settings. On first launch a copy of this file is copied to the personalised conf directory
personalised conf	.opendbcopy/conf	SQLDrivers.xml – contains personal URLs, User Names and Passwords, default Source and Destination Connections

5.1 JDBC Drivers

In general one can say that JDBC drivers working with Squirrel-SQL should also work with openDBcopy.

Using an external XML file to configure driver pre-selections lets one change default database driver classes, names and connection URLs easily. Please note that openDBcopy has not been tested with all databases and drivers listed. Instead this list and parameters have been borrowed from Squirrel-SQL. The structure is similar, but not the same. If you find errors or missing JDBC drivers within the list provided, please report.

If a database driver is not listed within 'Driver Name' selection one can easily append a further entry into the SQLDrivers.xml file. Please note that updating the default SQLDrivers.xml file within openDBcopy's conf directory, not your personalised folder, does not have any effect as a copy of this default file is only copied once automatically to your personalised folder.

An entry consists of the following elements:

Element	Description
Driver Name	The driver name to show in the Driver Name selection
Class Name	The complete java package and class name of the driver to use
URL Value	Default URL (be aware to escape XML characters by appropriate characters)

When appending a new driver or modifying an existing one, please return a short feedback to the developer(s) of openDBcopy.

5.2 SQL Type Mapping

openDBcopy uses an XML file to map SQL Types (java.sql.Types constants) to the appropriate java types. Currently this feature is only required by the Migrate Database Schema Plugin. For further details see http://java.sun.com/j2se/1.3/docs/guide/jdbc/getstart/mapping.html.

An entry consists of the following element attributes:

Element	Description
Number	The java.sql.Types Constant (see JDBC Specification)
Name	A name which may change depending on the RDBMS. Therewith this parameter is less significant and not used by the schema generation plugin.
Mapping	The appropriate java type, e.g. java.lang.String, java.math.BigDecimal, byte, byte[], ...

I have noticed that for example Oracle uses additional java.sql.Types constants for Timestamp types. The JDBC specification default is 93.

But using Oracle, 1111 is also mapped to java.sql.Timestamp. To find out such mismatches, it is handy to use Squirrel-SQL, providing details about a certain JDBC driver. openDBcopy also captures this information when connecting to a database. When saving a captured database model as plugin or job, one gets a complete list of all supported database types. Open the XML file in an editor for details or use Squirrel-SQL.

5.3 Email Notification of Job Execution

To turn on email notification for plugin execution status (simple information or errors) open the log4j.properties file

change

```
log4j.logger.opendbcopy.plugin=INFO
```

into

```
log4j.logger.opendbcopy.plugin=INFO, MAIL
```

Then modify the following line with appropriate information

```
log4j.appender.MAIL.layout.ConversionPattern=
MAIL_SERVER&SENDER_EMAIL&RECIPIENT_EMAIL&%d&%5p&%c&(%F Line:%L)&%m%n
```

Replace

`MAIL_SERVER` with your mail server

`SENDER_EMAIL` with the email address that shall appear as sender

`RECIPIENT_EMAIL` with the email address of the email recipient

Save the changes and restart openDBcopy. Now all information and errors (hopefully not) generated by plugins are automatically sent to the email recipient as separate email showing the complete message and date / time of creation.

5.4 Language

openDBcopy uses language specific properties files. Currently, English and German are supported. Other languages can be simply added. If you like to translate openDBcopy into your language, please tell.

To change the default language of openDBcopy, change the following parameter in the opendbcopy.properties file

```
# English
default_language=en
```

into

```
# German
default_language=de
```

5.5 File Encoding

To change file encoding of all files used change the file encoding in opendbcopy.properties, save the changes and restart openDBcopy. The default encoding is UTF-8 (Unicode). Encoding becomes important when a database uses non ASCII characters for table or column names.

Change the following parameter in opendbcopy.properties when changing the XML file encoding

```
encoding=UTF-8
```

5.6 Log Level

Especially when developing new plugins further log information can be helpful. Each package of openDBcopy is configured with a log level. Change the appropriate package log level in log4j.properties. Available log levels are DEBUG, INFO, WARN, ERROR, FATAL.

5.7 Look and Feel – Screen Sizes

By default the look and feel is the operating system's look and feel. Future versions maybe implement other look and feel selections. openDBcopy's screen sizes can be adjusted using the following properties in opendbcopy.properties

```
frame_main_width=1024
frame_main_height=768

frame_console_width=500
frame_console_height=600
```

5.8 Directory Locations

Locations to directories such as log, personal plugins and jobs, SQL drivers etc. can be modified to point to other default locations as within the default distribution.

All directories, which can be modified, are specified within opendbcopy.properties. Change appropriate if required. One may use relative or absolute path names.

5.9 Disable GUIs in Batch Mode

Once a job is configured and tested, one can disable GUIs and let openDBcopy only run in batch mode, without any graphical user interfaces. Please note that an important factor for performance improvements is the network bandwidth between the database systems accessed. Another factor is turning off the Graphical User Interfaces to speed up execution. But this has less effect as when placing openDBcopy on a machine with a high – very high – network bandwidth.

To turn off GUIs change the following parameter in opendbcopy.properties to false

```
show_gui=false
```

Once you have changed this parameter to false, don't forget to turn it on again if you want to run openDBcopy again in interactive mode.

5.10 Default Browser(s)

openDBcopy uses a special Browser Opener library, com.Ostermiller.util (see http://ostermiller.org/utils/Browser.html for details).

On first start of openDBcopy this library tries to detect the operating system used and appropriate default browser(s). Certain operating systems provide several browsers by default, such as a Linux distribution.

Everything required to open the browser of your choice is the path and programme name to the appropriate browser. If the utility detects more than one browser, commands are separated by comma.

On Linux the list of commands automatically detected looks like:

```
mozilla –remote openURL({0}),mozilla {0},konqueror {0}
```

Please note that there are no spaces between commas and new commands. The parameter 0 in brackets is the placeholder for the URL to open.
Windows users please be aware that spaces within paths are not allowed or must be put in brackets, e.g.

```
"C:\Program Files\Firefox\firefox.exe {0}“
```

6 Starting openDBcopy

6.1 Friendly Operating System Users

Linux, Unix or Mac users please call the `bin/start.sh` script to launch openDBcopy. The setup wizard should have changed the permissions (`chmod`) to allow execution of `bin/start.sh`.

6.2 Microsoft Windows Users

When running openDBcopy on a Windows platform one can either start openDBcopy using the shortcut 'openDBcopy launcher' on the desktop or click on 'openDBcopy launcher' within the programme group openDBcopy under Start Menu, Programs.

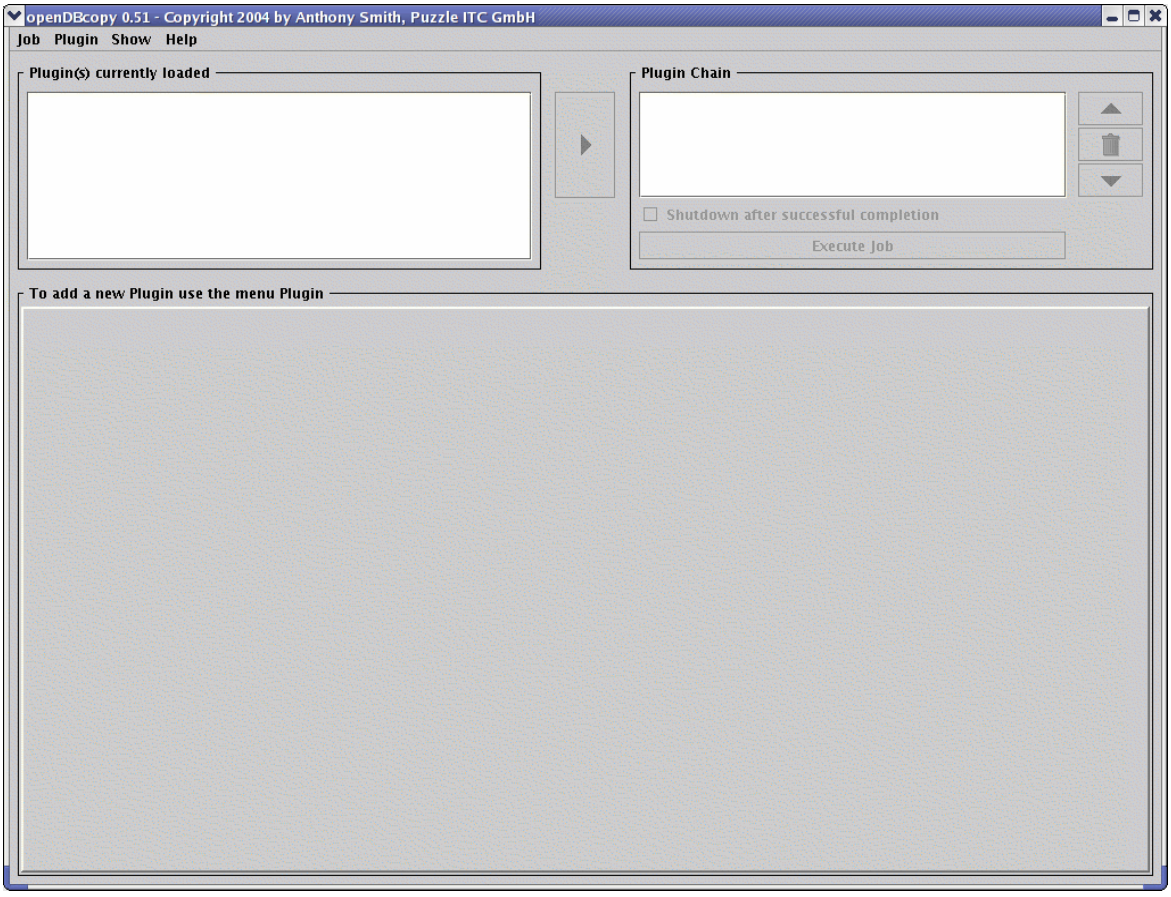
Windows users can also start openDBcopy by calling the `bin/start.bat` from a command line or Windows Explorer.

openDBcopy is launched via a shell and then opens a graphical user interface. Do not close the shell window if you do not want to close openDBcopy too. The shell window is automatically closed when closing openDBcopy via menu 'Project, Exit'.

7 Screens of openDBcopy

7.1 Main Screen

When launching openDBcopy the following screen is the main screen. To configure and execute a plugin, select a Plugin from the Plugin menu. To import or export a job configuration (serially executed plugins) use the Job menu. To exit openDBcopy use the "x" at the top right or "Exit openDBcopy" from the Job menu.



To display / hide other windows use the "Show" menu.

7.2 Console Log

The following screen shows all application log information. When having trouble launching openDBcopy, you normally see where it stuck. The progress bar and entries below provide more information.

Once openDBcopy is successfully loaded, you can close and reopen this screen at any time by using the appropriate entry under the "Show" menu. If you close this console screen window while openDBcopy is still loading, the whole openDBcopy launcher process is killed too.

All log information shown on this screen is also logged in your home directory

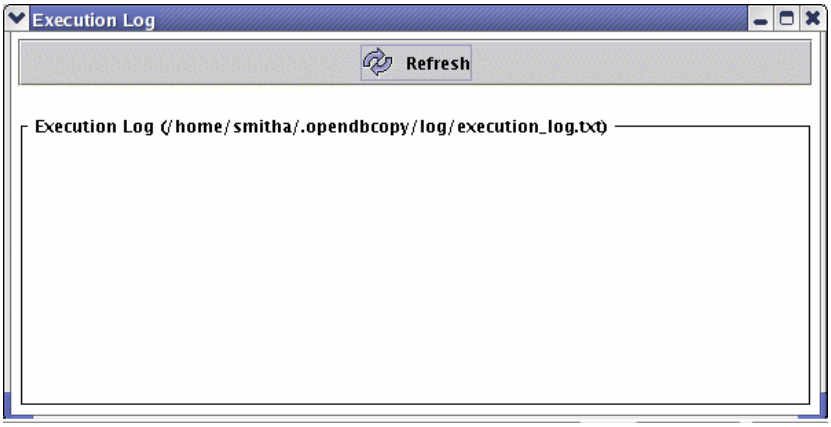
`/<your_home>/.opendbcopy/log/application_log.txt`

7.3 Execution Log

The following screen shows all job execution log information. The execution log screen is automatically opened when executing a job (when clicking on “Execute Job”). You can close and reopen this screen by using the appropriate entry in the “Show” menu. Please note that for better system performance of your job, you have to click the refresh button if you want to show the latest execution log information, while the job is still running. Once the job is done, the execution log is updated automatically.

All log information shown on this screen is also logged in your home directory

`/<your_home>/.opendbcopy/log/execution_log.txt`



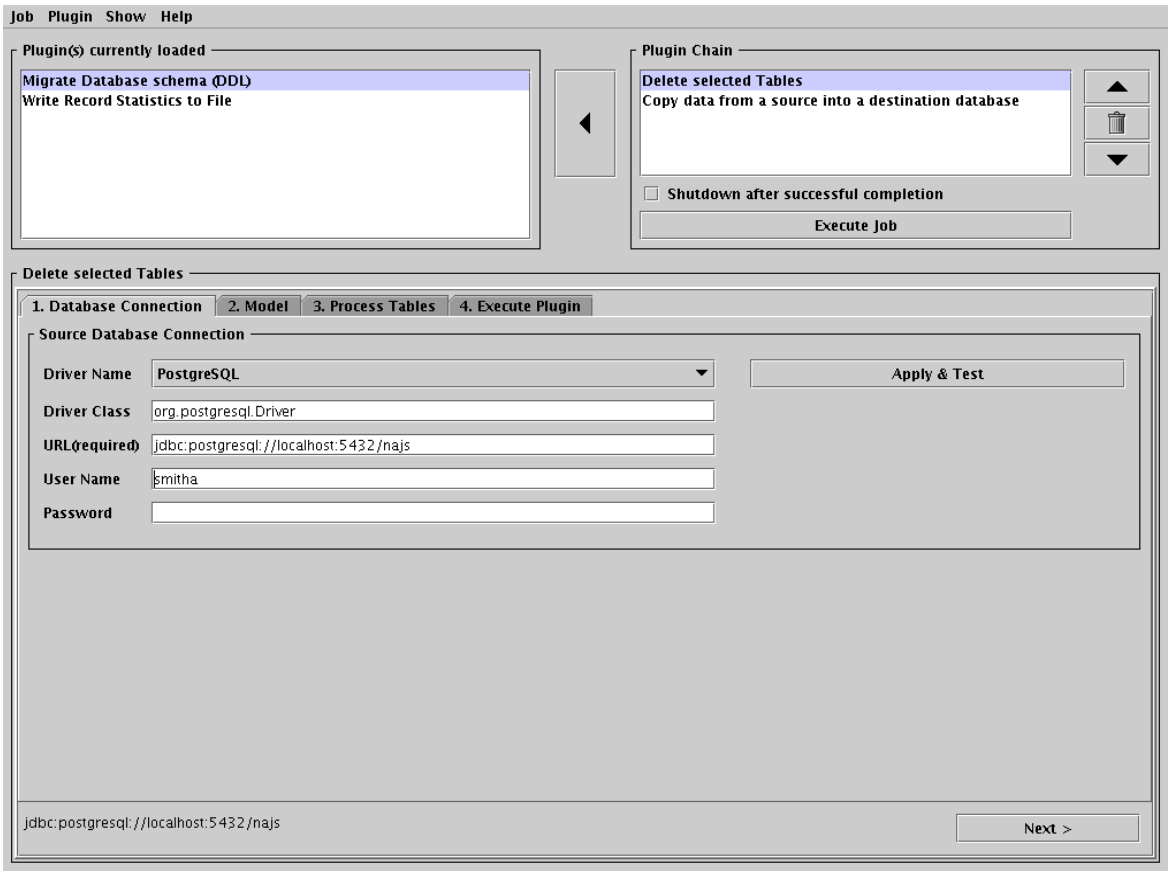
8 Executing a single job or series of jobs

OpenDBcopy is able to of execute several dependent or independent jobs serially – using the Graphical User Interface interactively or in batch mode. Serially means that before another job can be executed, the former must be completed successfully and so on. A job is done by a plugin. A series of jobs is called a job in openDBcopy.

In many cases it is sufficient to execute a single plugin interactively. But openDBcopy is also used in production environments to do daily batch jobs itself.

Please note the difference between plugins bound to a job and unbound plugins. Unbound plugins can only be executed interactively, using the graphical user interface. Plugins bound to a job can be launched interactively or in batch mode.

The following screen shows the difference between bound to a job and unbound plugins.



On the left hand side are unbound plugins, meaning those are not part of a job. On the right hand side are plugins bound to a job. The order of bound plugins is important. The screen shot shows a job which firsts deletes data. The second plugin copies table data from a source database into the previously deleted tables. Like this a slave table can be refilled from a master table. Note that this is brute force – there are nicer ways to update, respectively synchronise database tables. However, those two plugins allow you to do such a master / slave update database independent.

Clicking on a bound or unbound plugin brings up the appropriate configuration below.

The arrow between unbound and bound plugins allows to move unbound plugins to bounded plugins and vice versa. Only plugins bound to a job can be deleted. Unbound plugins are removed after restart of openDBcopy, unless you export and then import the appropriate plugin as a plugin (not a job).

Saving bound plugins stores all plugin configurations within one single job xml file.

9 Importing / Exporting a Job

As mentioned earlier in this document, a job is a series of individual plugins, which can be run interactively or as batch job.

To store a job or reopen it, use the appropriate Import Job ..., Export Job ... menu items under Job. A job is saved as plain XML file which can be edited either by reopening it using openDBcopy or by text editor.

However, until today I haven't implemented a DTD or schema for openDBcopy configuration files. So currently an invalid XML file is only recognised if a certain method tries to access its content. A DTD or schema is likely to being implemented soon for automatic XML file validation.

A job can be exported or imported from a file at any time.

The default location for job files is in your home directory

`/<your_home>/.opendbcopy/jobs`

10 Importing / Exporting a Plugin

Individual plugins can also be saved as XML files. The advantage here is that plugins with a different default configuration may make more sense in your daily business than the default plugin configurations in the appropriate

`opendbcopy/plugins/<plugin.dir>/plugin.xml`
directory. You can either save a plugin configuration under

`/<your_home>/.opendbcopy/plugins`
providing a new name for the file or overwrite the default plugin.xml file of the appropriate plugin. Decide yourself what suits you. Remember that when updating openDBcopy with a newer release, the default plugin directory may be overwritten, while the personal .opendbcopy folders are not.

11 Executing openDBcopy in Batch Mode

One goal of openDBcopy was always to keep it flexible. This also in terms of launching openDBcopy on different systems, with or without a graphical user interface.

Especially when you plan to use openDBcopy for a huge database migration, which has to be heavily planned and being executed once during a free time slot – somewhere between end of December and first of January, somewhere between 24:00 and 00:00 ... know what I mean? The deadline is fix, the day X has been fixed by the managers – and now it is up to you to make it working – successfully!

I have used openDBcopy for such a scenario. There was one night reserved to migrate all data from DB2 to Oracle – a live system migration! And it worked! During the previous months the migration process and involved applications have been tested about ten times on test environments.

Automatic Monitoring and intelligent exception handling were my insurance to still get some sleep during this night X.

For this migration I developed a special plugin which was able to handle all sorts of failures. Lost of network connection, lost of database session etc. And in certain cases the plugin had to redo the job itself; without having to begin from the very beginning.

Especially when working in batch mode, performance is important and you want to know how long jobs take to execute.

The most important performance factor for openDBcopy is the network bandwidth. When reading data using a JDBC driver, always over TCP / IP, I have noticed that having a 100MBit/s network connection against a 10MBit/s connection is almost proportional in speed – meaning your openDBcopy job is about 10 times faster using a 100Mbit/s network connection!

When using openDBcopy in batch mode, we launch it via a Linux Cron Job on a database nearby server, having a 100MBit/s network access.

Running openDBcopy in batch mode does not require any Graphical User Interface. You can turn all GUIs off, see chapter Configuring openDBcopy.

A repetitive job, launched via Cron Job, requires that openDBcopy exits itself after successful completion. Set the appropriate check box in the main window.

Use the E-Mail notification options for the status of job execution. See configuration chapter for details. On our night X project we used a Mobile Phone Gateway sending an email as text message to my phone in case of errors.

To launch openDBcopy in batch mode, use the appropriate `bin/start.sh` / `bin/start.bat` file and provide as parameter the path and location to your job file.

E.g. Linux users can use Cron Jobs

```
>./start.sh /home/smitha/.opendbcopy/jobs/myjob.xml
```

Windows users can use a big variety of tools to schedule and launch batch files. In above example replace `start.sh` by `start.bat` and the path to your job file accordingly.

Future version maybe will contain Quartz scheduler (see http://www.quartzscheduler.org) for simple and integrated job scheduling.

12 openDBcopy Plugins

12.1 The Plugin Connection

A plugin for openDBcopy is normally a small programme which can be launched and controlled by openDBcopy. A plugin is always executed in a separate thread. Plugins may require input, may produce output, but a plugin does not know what has to happen before it receives a thread for itself, nor does a plugin know, what is happening, after its thread has terminated.

Plugins can be executed stand-alone or as part of a job. Plugins bound to a job are executed serially in the order specified by the user. If a plugin of a job fails, by default the whole job is interrupted. A job or single plugin may also be interrupted by the user.

12.2 Plugin Life cycle

The plugin life cycle is controlled by the openDBcopy core framework. Certain parameters can or must be set by the user before a plugin can be executed. Once configured, the plugin may be executed stand-alone or as part of a job.

But before any plugin can run, its resources must be detected and loaded.

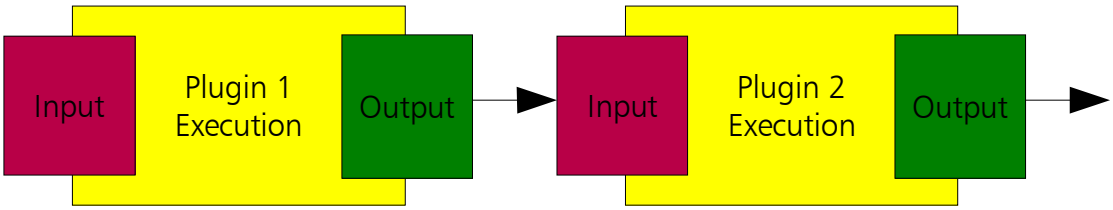
A plugin is always stored in a separate directory. The directory itself is stored in opendbcopy/plugins and consists of the following components:

- plugin.xml
- gui.xml
- 1 to n archives (*.jar or *.zip) within a sub-directory called lib (library)

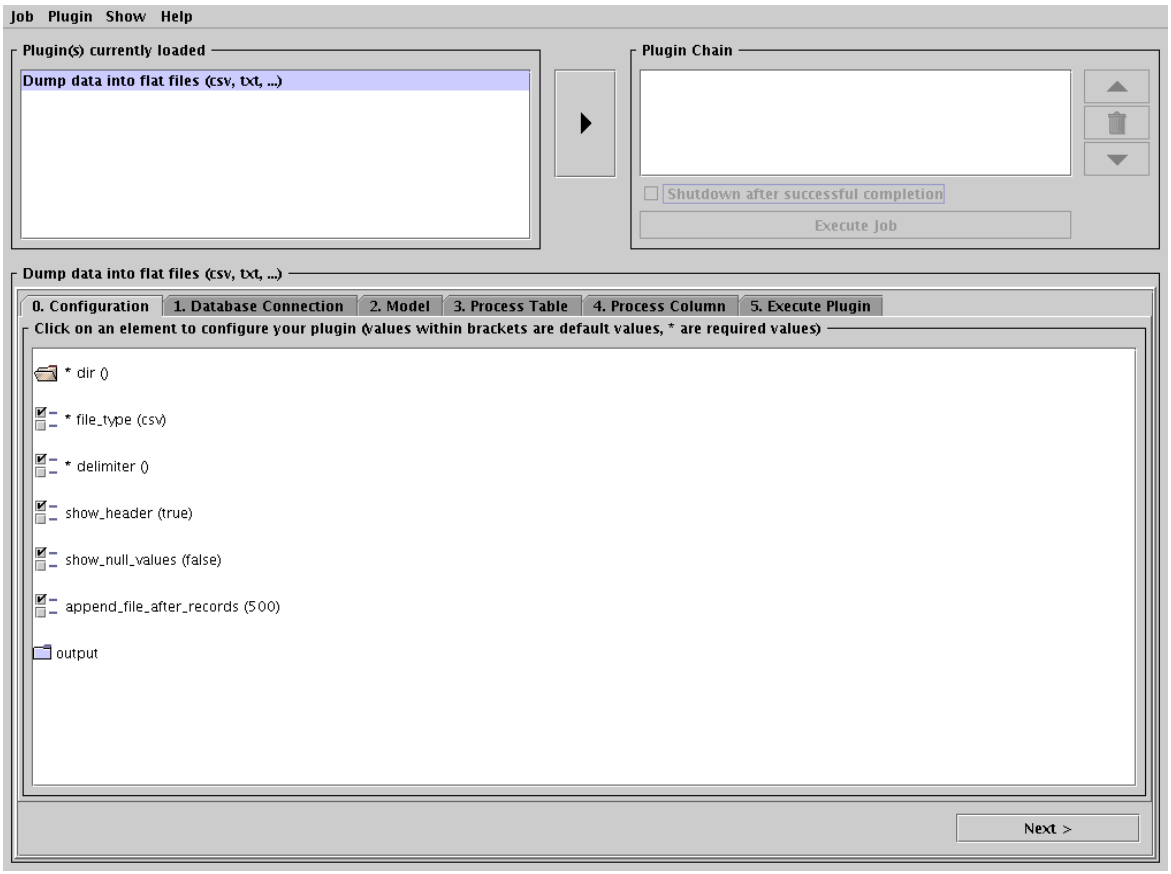
12.3 A Plugin's Backpack

The following drawing shall help to understand a plugin's backpack. A plugin may require input before it can execute any code. Once executed, it may provide output for other plugins, but not knowing what other plugins follow. The openDBcopy core framework, as the circus director, is responsible to handover the arena to the next artist. Only the circus director knows the programme. In case of problems the circus director has to react accordingly. It is not the artist's job nor responsibility to watch across his/her show. He/she may produce output, but not knowing if a following artist requires or can use this as his/hers input for his/her show.

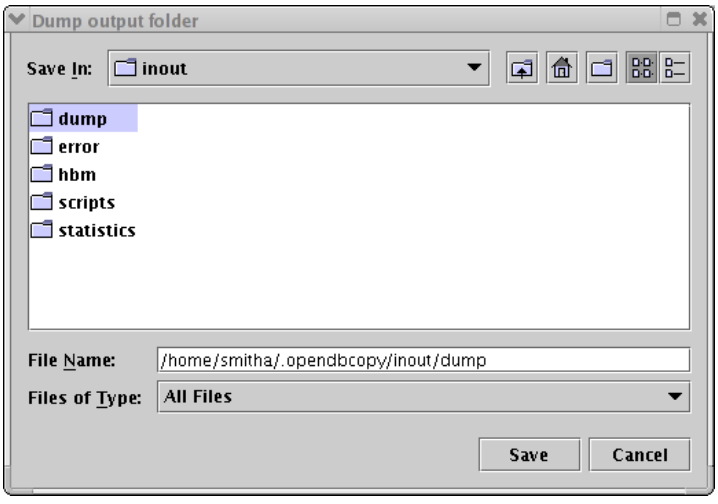
Certain input already exists before the circus director starts the whole show. All those static resources, such as fences for the lions etc. must be configured before the lions enter their arena, but just before they enter their arena. On the other hand lions may require food. What if the food is not ready before the show begins? What if the food is delivered just in time? Before the lions show? Then the lion's artist can configure food as a dynamic parameter, giving it a name and type, but knowing that the food is delivered just in time.



To configure plugin parameters, openDBcopy provides a standard Graphical User Interface. A plugin developer may implement a different GUI, but in many cases the following GUI is sufficient.



Given an icon, a description and a value, a user can specify plugin parameters. Click on any item to change a value. Certain parameters provide default values. Certain parameters are required. Required parameters have an * asterix in front. Depending on the parameter type, a different dialogue pops up when clicking on an item. E.g. When clicking on the item dir(), the following screen appears:



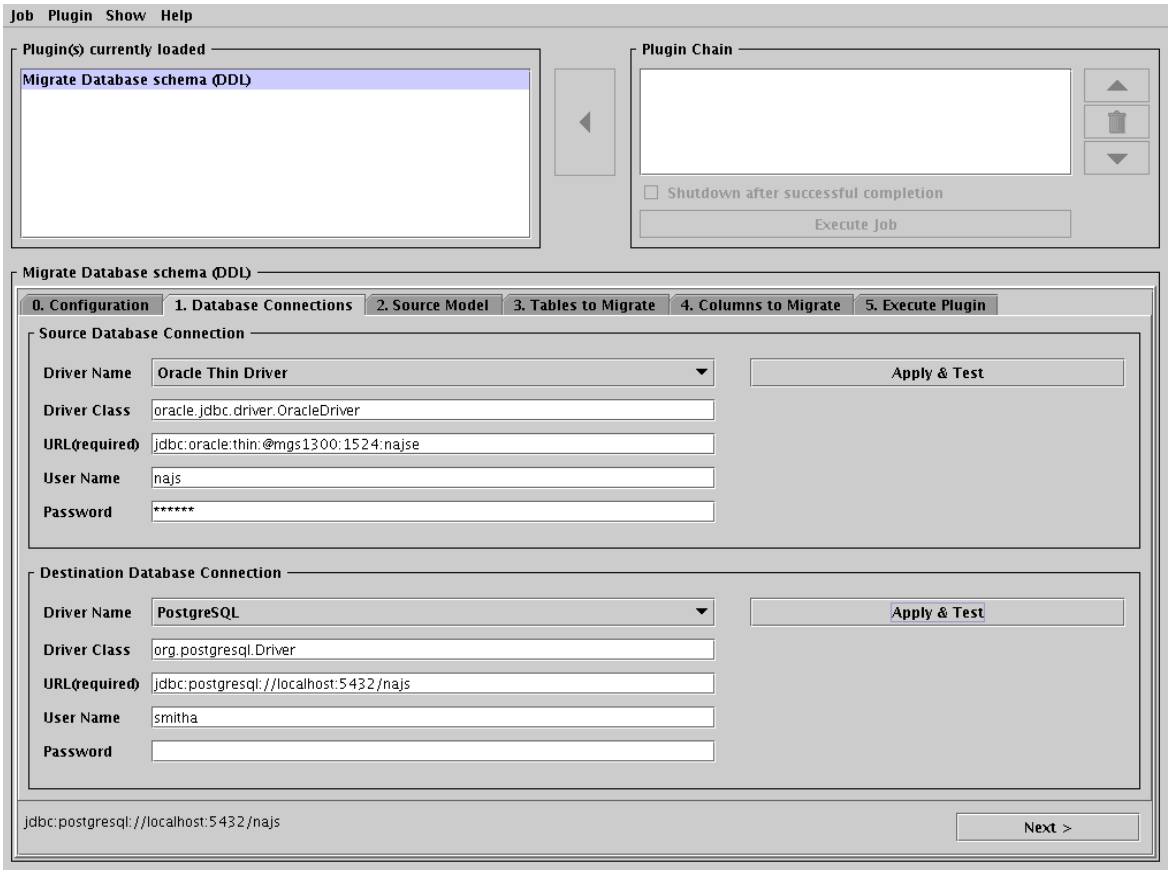
Please note that further information is displayed when you place your mouse over a parameter or in the window title of the parameter modification window.

12.4 Specifying Database Connections

All database plugins provide the ability to select database connection details. If the JDBC driver is available form the pre-configured list, a default URL and JDBC driver class name are automatically filled into the appropriate text fields. If not, you can still enter the details yourself. To have those details available after a restart of openDBcopy, enter the new JDBC driver information into the SQLDrivers.xml file. See configuration section for details.

Please note that the URL must be further configured, if one has not yet done so in a previous session, which saves source and destination details automatically – including passwords (plain text!). Replace <server> by a server URL, <dbname> by a database name etc.

In the user field enter the user which has the required rights to execute a certain plugin. To read database meta data no special user rights are required! This is defined in the JDBC specification.



Once done, click on “Apply & Test”. When receiving errors please have a look at the error message. Normally the error messages thrown by JDBC drivers tell what needs to be changed, what is wrong etc., e.g. invalid user / password, missing database connection etc.

12.5 Source and Destination Model

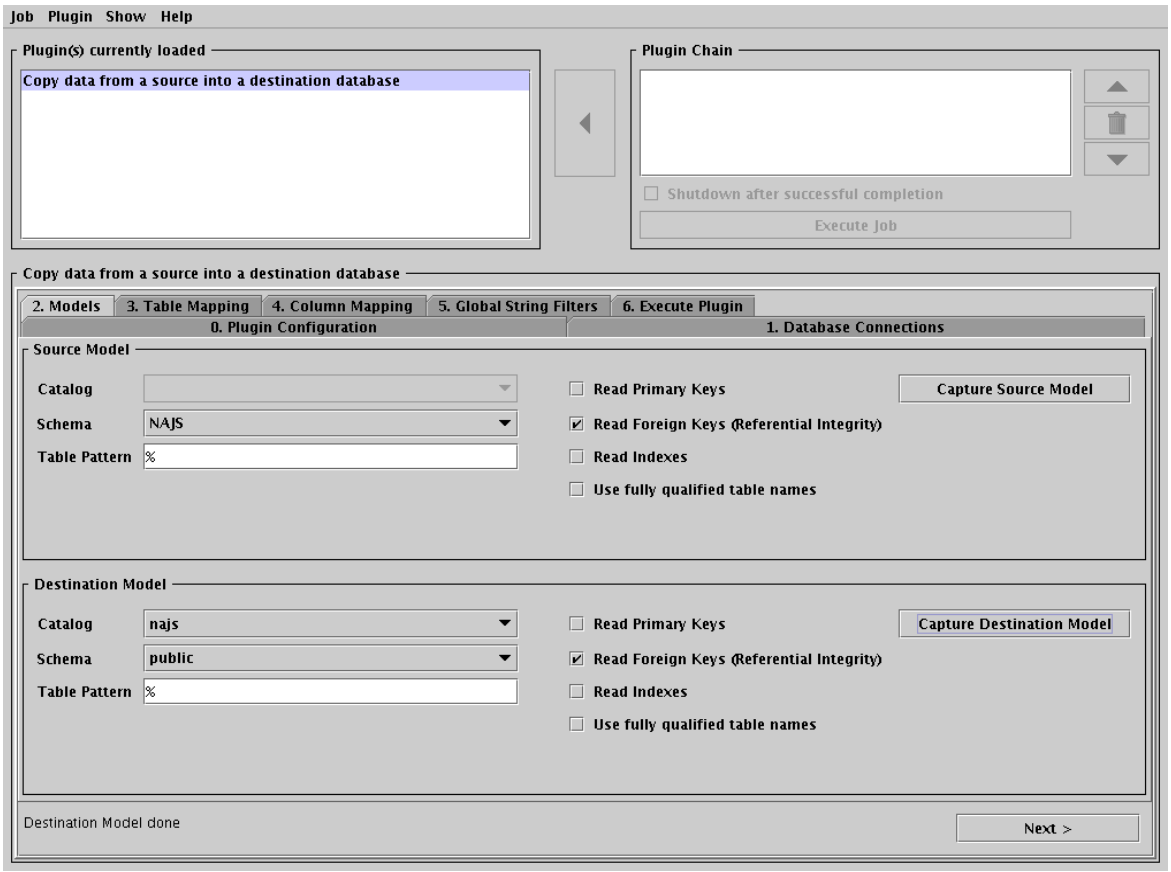
All database plugins provide the ability to select a source and / or destination schema / catalogue.

The following options allow you to specify what schema / catalogue shall be captured. The table filter, default % (SQL Syntax), allows to filter table names being captured. E.g. when capturing a database with 10'000 tables, you maybe do not want to capture and migrate all these tables. Use a table filter, e.g. TABC% to capture only tables beginning with TABC and any following character(s).

On the right hand side specify options for referential integrity, indexes and qualified table names. When the JDBC driver supports referential integrity, enable reading Foreign Keys! Reading Primary Key model information as well as Index information does not make sense for all plugins.

Qualified Table Names means how a table shall be accessed. Qualified is using the schema / catalogue name as prefix, normally a dot (.) as separator and finally the table name itself. As example, najs.tadresse to access table tadresse from the schema najs.

Certain JDBC drivers and / or RDBMS do not support qualified table names. Therefore disable the check box qualified table names for the appropriate database.



13 Plugin: Migrate Database Schema (DDL)

13.1 Description

A general problem of database migrations is to migrate a database schema. Certain tools, commercial and non-commercial, provide solutions to this problem. Some tools are able to migrate table and column definitions. But what about referential integrity constraints? Indexes? Compound Keys?

I have been looking for a solution being able to do all this – database independent. A database SQL syntax to create or update a table definition script – DDL (Data Definition Language) is standardised, but not all systems use the same standard syntax.

Instead of implementing database specific DDL generators, I decided to use a different technology. Given a JDBC driver and Relational Database Management System, the Object / Relational Mapping Framework Hibernate (http://www.hibernate.org) can generate and directly implement a database specific DDL.

For using Hibernate a Java class representing a persistent object and appropriate Hibernate Mapping File (XML) is required. To generate such files, Hibernate and available tools provide several mechanisms. The most flexible way is to generate the Hibernate Mapping Files based upon the tables, referential integrity constraints, indexes etc. This is what this plugin mainly does. Nothing more. The rest, creating persistent Java objects (by the way you can use those files for your persistence layer), generating and implementing the DDL is done by Hibernate tools. Therefore you can use the Apache Ant targets provided with this plugin.

13.2 Parameters

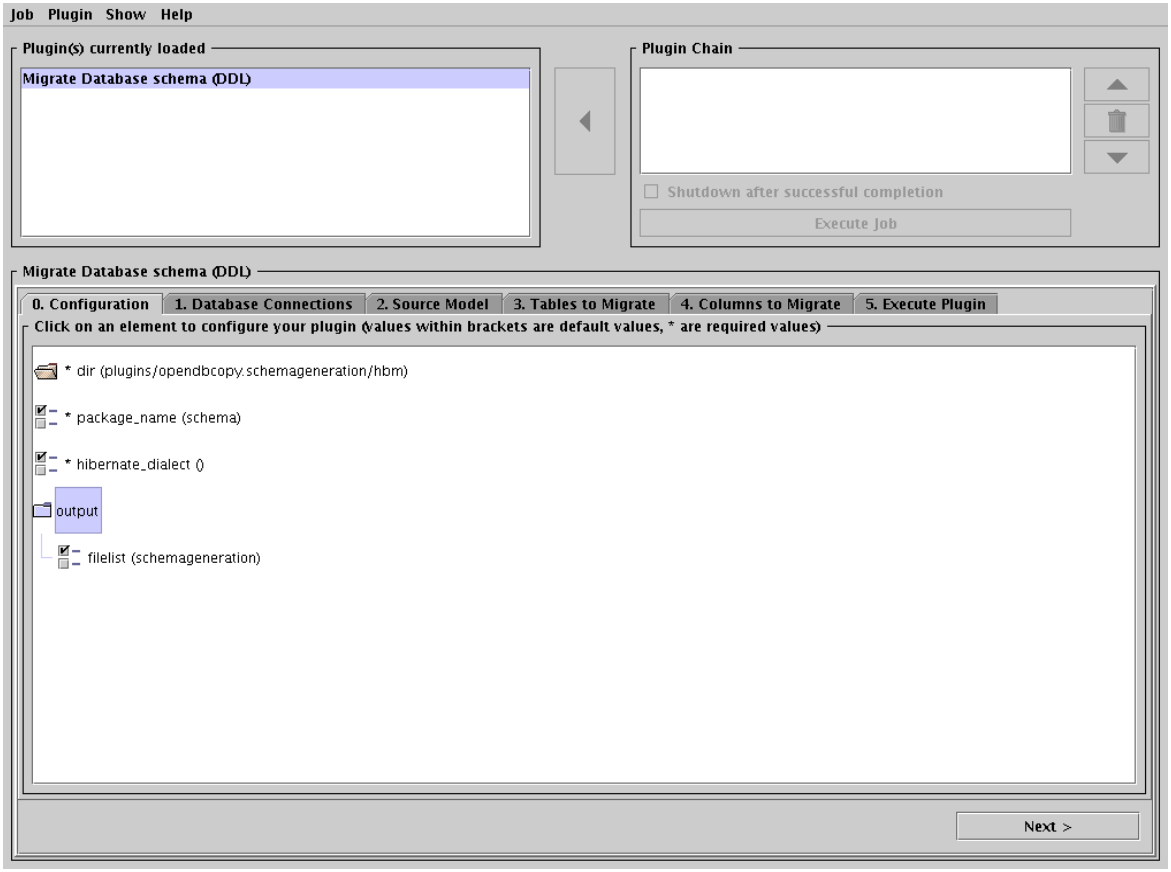
The following parameters can or must be specified.

Name	Value	Required	Description
dir	path	yes	path (relative or absolute) to store Hibernate Mapping Files
package_name	java package name	yes	Java Package prefix. E.g. schema is the default package name.
hibernate_dialect	DB2400, DB2, Firebird, FrontBase, HSQL, Informix9, Informix, Ingres, Interbase, Mckoi, MySQL, Oracle9, Oracle, Pointbase, PostgreSQL, Progress, SAPDB, SQLServer, Sybase11_9_2, SybaseAnywhere, Sybase, Generic	yes	These are the Hibernate dialects provided by Hibernate at the time of this documentation. It is very easy to add additional Hibernate Dialects. If your destination database is not listed, try using Generic.
filelist	schemageneration	no	Output of this plugin is a filelist containing path / file names of files generated

13.3 Prerequisite

- An existing source database schema / catalogue to migrate
- An existing – empty - destination database schema / catalogue

13.4 Configuration



In general, all plugin input and output is read and stored outside from the openDBcopy directory structure. In general use the personalised .opendbcopy directories for input and output.

- dir
  This plugin is special as it requires Apache Ant to generate Java Persistence Classes (source and compiled files) and finally the DDL. Therefore all Hibernate Mapping Files are stored within the hbm directory of the schema generation plugin itself. Keep the default directory value if you do not want to change the Ant build file afterwards.

- package_name
  schema is the default value. All java classes are then stored within the package called schema. When changing this parameter take care of the package naming conventions.

- hibernate_dialect
  This is the most important parameter. If your destination database system is not listed, try using Generic or develop your own Hibernate Dialect, which is fairly simple. Oracle users must take care that Oracle 9 is different than the Oracle dialect itself. The selection of the dialect looks as

- filelist
  As long as no other plugins following the execution of this plugin require input from this plugin, one can leave this parameter as it is.

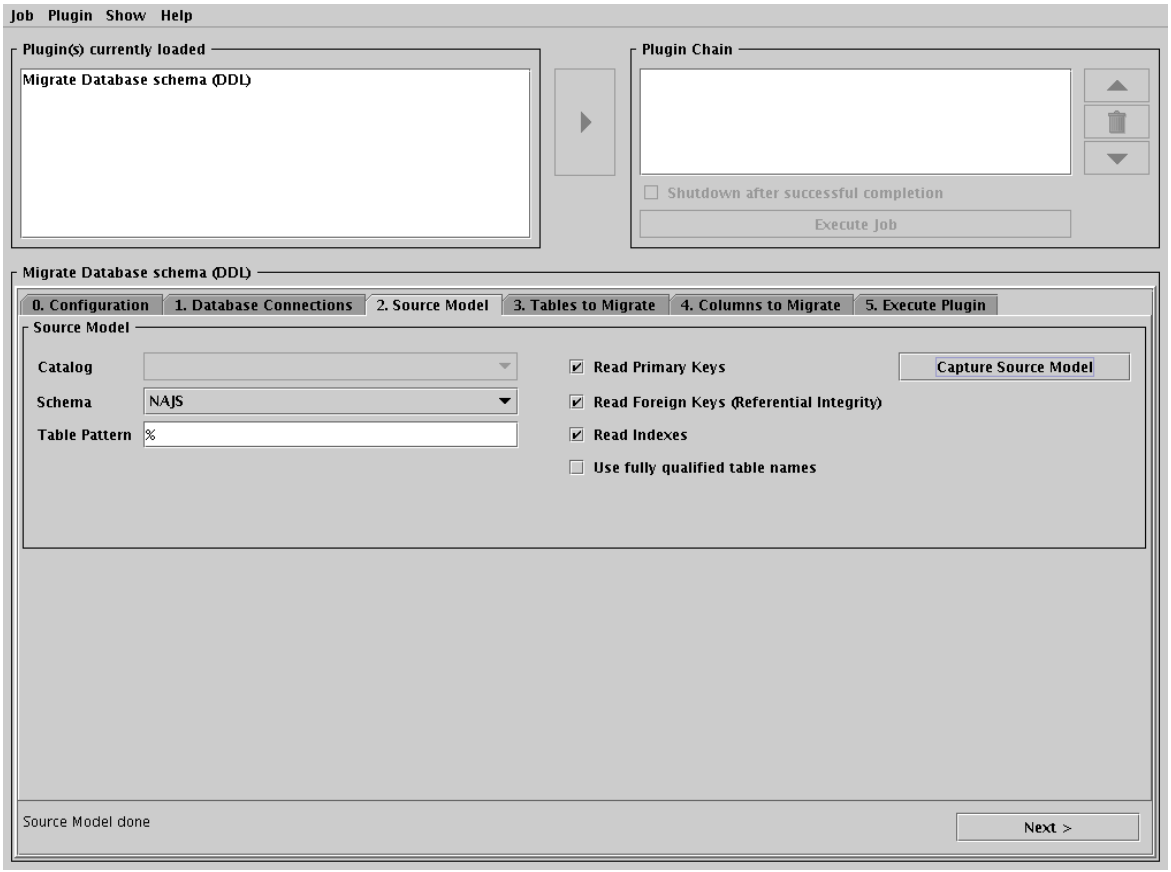
13.5 Database Connections

Specify your source and destination database connection details. A valid destination connection is required to complete the job of this plugin, even if there are no tables available yet. Click on Apply & Test.

13.6 Source Database Model

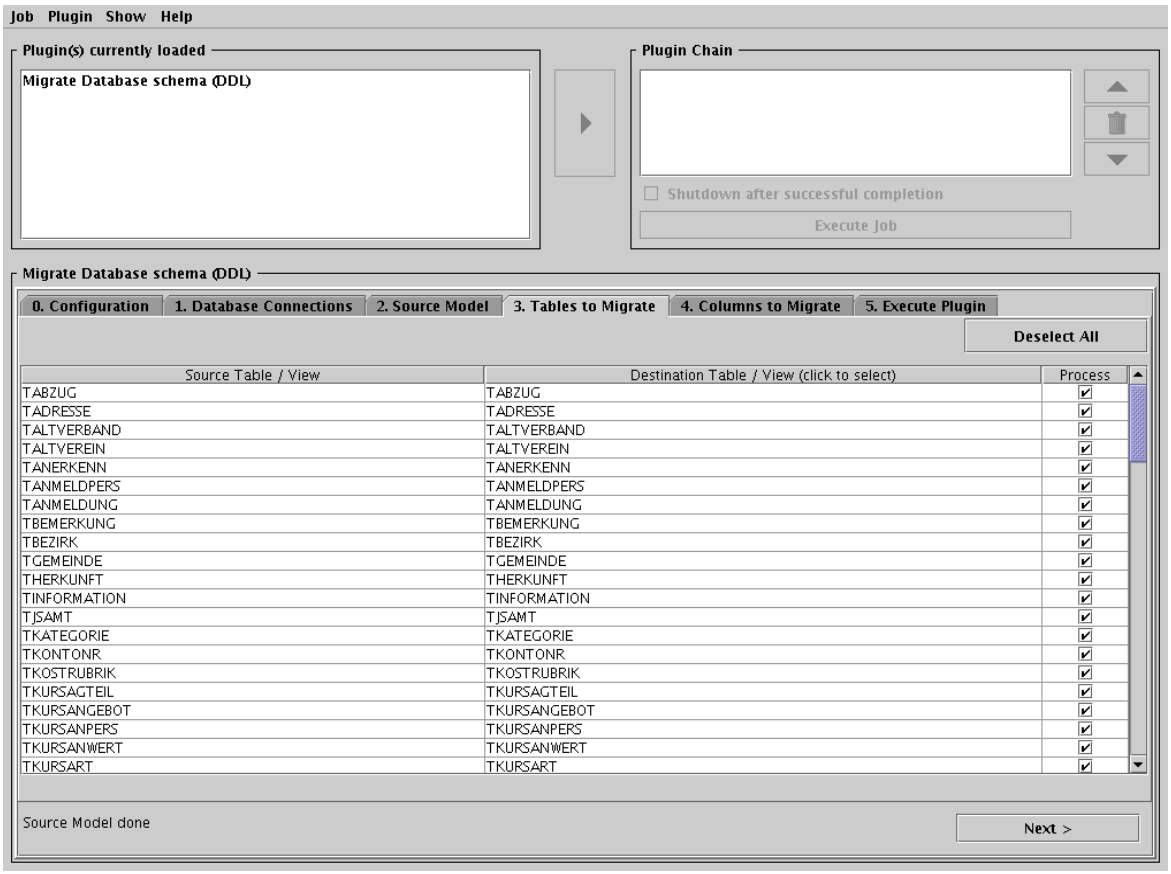
Please note that not all JDBC drivers and RDBMS support referential integrity and / or indexes. If such features are not supported by the JDBC driver used, you receive an error message. But if it is supported, it is very important that primary keys, referential integrity and index information are captured and migrated too. This is one of the big differences compared to other migration utilities.

In case of Oracle using the user NAJS, the Schema NAJS is selected by default. Certain RDBMS use catalogues, others catalogues and schemas. To capture the source model click on Capture Source Model. This may take a while, depending on the size of the model to capture, network bandwidth and RDBMS bandwidth.



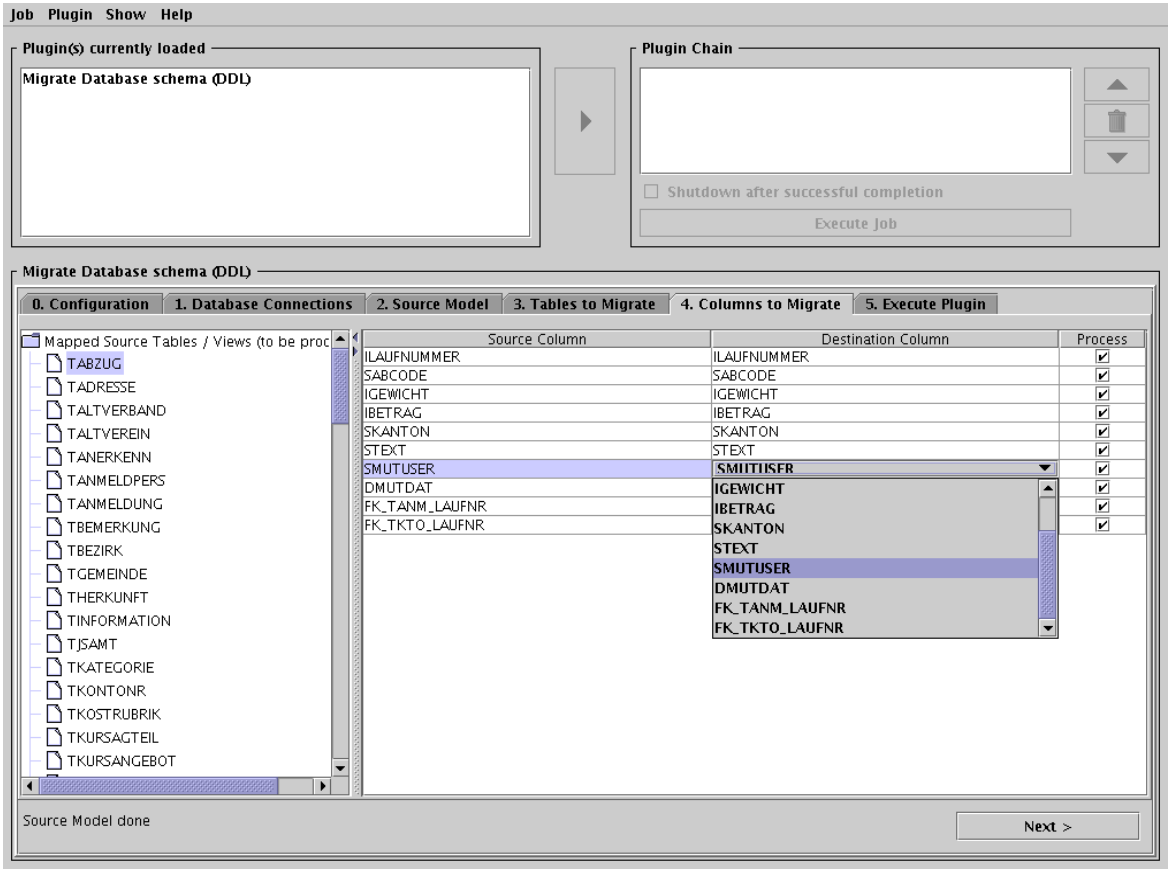
13.7 Tables to Migrate

Select the tables to migrate. Note that when referential integrity is used, certain child tables cannot exist without their parent table(s). In general when using this plugin to migrate a database schema it is expected that the user knows what to migrate. Please note that this plugin is still under quite some development. One planned feature is the ability to change table and column names, column types, referential integrity etc. Currently one can only select / deselect tables to migrate. If one wants to change other details, those must be modified on the generated Hibernate Mapping Files, which requires Hibernate Know-How.



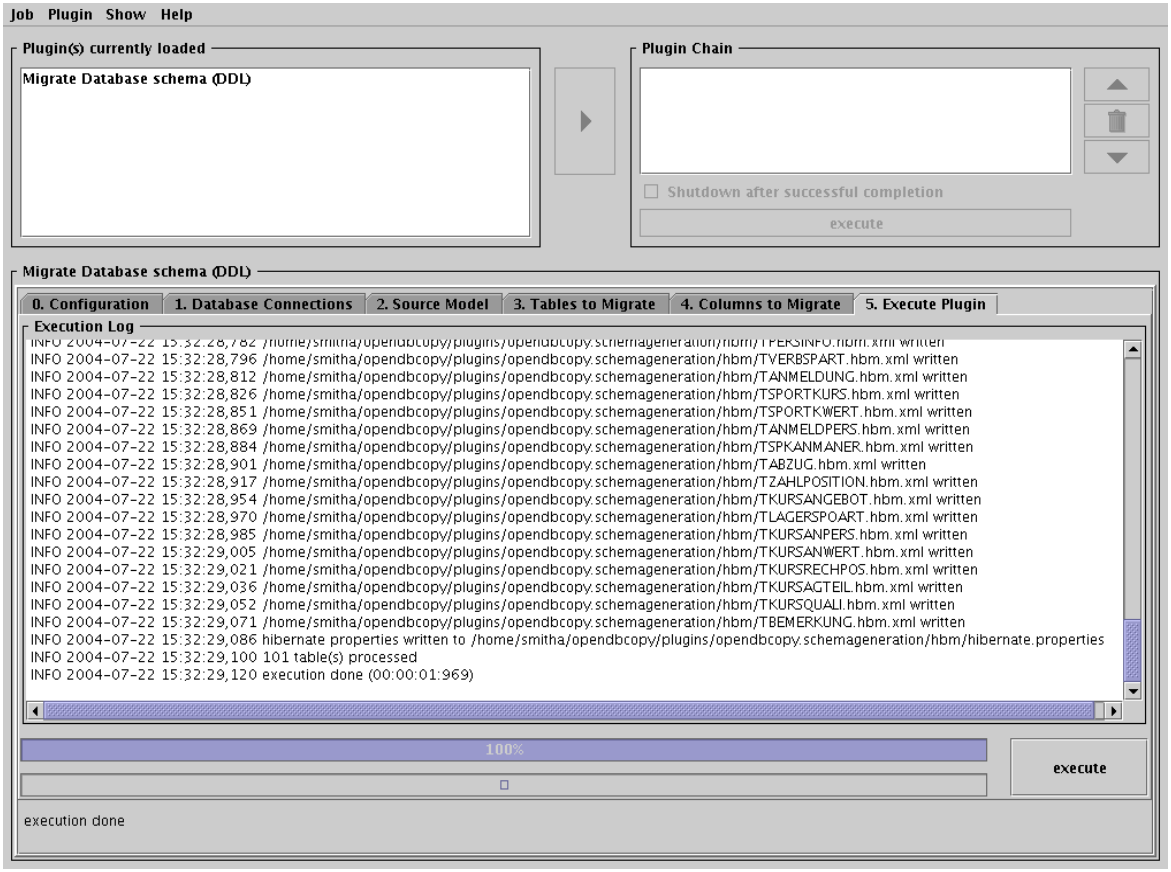
13.8 Columns to Migrate

On this screen one can deselect individual table columns selected for migration. Please note that changing column names requires editing of the Hibernate Mapping Files generated by this plugin in the next step. Future releases will provide a more user-friendly alternative to edit and create database schemas.



13.9 Execute Plugin

Click Execute to generate the Hibernate Mapping Files.



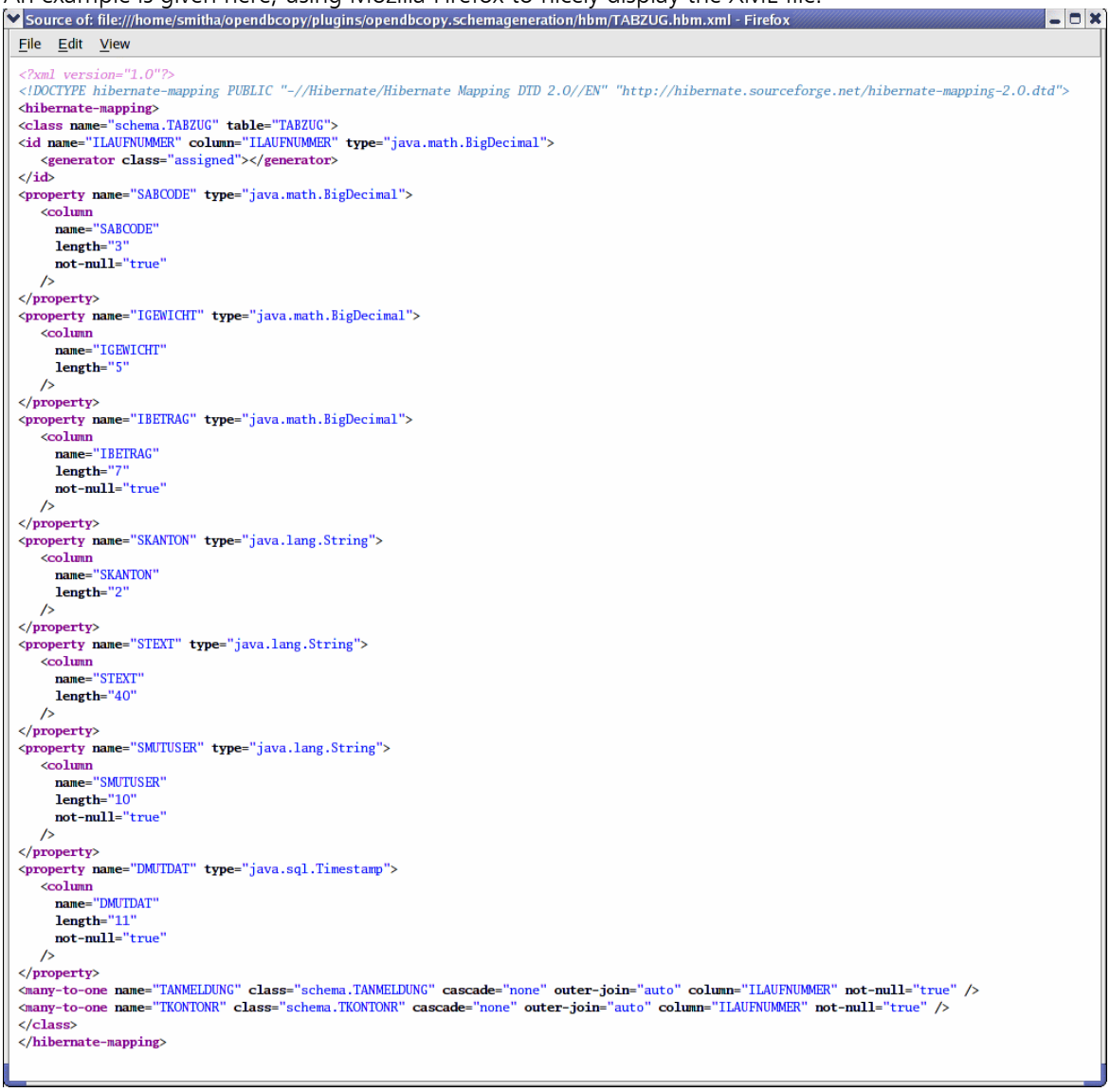
13.10 Modify Table / Column Names and other Properties

The current version of this plugin is not yet very user-friendly regarding modifying table and column names, referential integrity etc.

On the other hand splitting the process of migrating a schema into several individual steps, one reaches higher flexibility and receives the ability to influence the process at certain points. Other tools migrate a schema / catalogue in one click. Normally those programmes work, but the output is normally not what is expected, e.g. missing referential integrity, missing indexes, missing compound keys, etc. Without the ability to change a database model during migration a tool is not flexible enough. In general a migration requires slight modifications.

Therefore, using the current version of this migration plugin, one has to edit the Hibernate Mapping Files if one wants to change anything on table / column naming, insert new tables, columns, change referential integrity etc. To do such one requires the Hibernate Documentation, available online at http://www.hibernate.org

An example is given here, using Mozilla Firefox to nicely display the XML file.



13.11 Generating Java Classes from Hibernate Mapping Files

The remaining steps are done using Apache Ant (http://ant.apache.org).

If you have not installed Ant yet, do so first.

The Ant build.xml distributed with this plugin requires that the JDBC driver required to generate the DDL for the new database is stored within the lib directory of this opendbcopy.schemageneration plugin. Otherwise add the path to your JDBC driver manually within the Ant build.xml.

If your JDBC driver is within the lib directory of this plugin, enter the following command to generate the Java Persistent classes. This creates the source files and compiles those immediately. Compilation errors may occur in case the Hibernate Mapping Files contain keywords in Table and Column names which are reserved Java keywords. Please also note that the Hibernate Mapping Files have not been validated before. A future version of this plugin will have to check for valid table and column names.

```
>ant hbm2java
```

If successful your persistent classes (source and compiled versions) are now ready for the final schema generation.

13.12 Generating the DDL

Being successful so far, the remaining step is easy. Please note the Ant build.xml file provides targets to create, drop and even alter an existing database schema. Altering is only possible if the underlying JDBC driver and RDBMS supports altering.

To create the schema use Ant and enter

```
>ant schema-create
```

If the Hibernate Mapping Files, the compiled persistent classes and the hibernate.properties file generated by this openDBcopy plugin are valid, the schema is created and directly implemented into the schema / catalogue specified at the beginning of this migration. By default the new schema is also stored as SQL file. You can find this file in the sql directory of this plugin. The file name consists of the database product name and the ending _schema_create.sql, e.g. PostgreSQL_schema_create.sql.

To drop the schema use Ant and enter

```
>ant schema-drop
```

If you were able to successfully run this target, all tables of this schema are dropped! The sql directory of this plugin contains a drop sql script containing the actions just executed.

To update the schema use Ant and enter

```
>ant schema-update
```

If have never tested this plugin Ant target. What it cannot do is generate a SQL script. But, however, if the JDBC driver and RDBMS support altering table definitions, maybe this target is of use.

13.13 Alternatives for Migrating a Database Schema

Migrating a database schema can be easy or complex, depending on the differences between source and destination database system, new requirements etc.

Several tools are available for different jobs. For example native programmes exist to migrate Oracle to PostgreSQL. The advantage of this utility is, that it can also migrate stored procedures.

Another great open source tool is Middlegen (http://boss.bekk.no/boss/middlegen/) combined with Hibernate. It provides a generic way to generate code based upon an existing database schema / catalogue. The way Middlegen works with Hibernate is very similar to this plugin. Middlegen with Hibernate has some bugs, which this plugin does not have. One is missing support for unknown java.sql.Types. Another bug is that the order of columns of a compound key is not properly translated.

The future plans for this plugin also depend on you! If you like this plugin and see potential for extensions, don't hesitate to tell us or to help implementing such. One idea is to implement a graphical editor to create complete database schemas.

14 Plugin: Copy Data from a Source into Destination Database

14.1 Description

Originally this was the reason for developing openDBcopy – to copy table data from a source into a destination database. The source and destination database only need to provide a JDBC driver. If no JDBC driver is available and the underlying RDBMS supports ODBC, one can try to use the JDBC:ODBC bridge from Sun Microsystems. This JDBC:ODBC bridge is of type 1 and does not support referential integrity, indexes etc. This JDBC:ODBC bridge does not require an additional driver to be installed. One can directly select it from the list of available JDBC drivers.

One item when migrating data is referential integrity between tables. openDBcopy is able to read and respect referential integrity. Normally referential integrity can be analysed. An ordered list is the output of this analysis. This requires that there are no infinite loops in the schemas / catalogues analysed. In case of infinite loops, an exception occurs.

If referential integrity of the destination database schema / catalogue cannot be directly respected, try disabling or dropping those foreign keys temporarily. My experience is that such problems must be solved by try and fail, especially if the source database has different referential integrity constraints set, or none (most difficult, because you will mostly have crap data in the source schema / catalogue), and the destination database schema / catalogue has a well planned set of such rules already implemented.

Another problem often seen are NOT NULL constraints in a new destination schema / catalogue. The old database, source database, contains less or no constraints at all. One of the reasons for migrating is almost always combined with cleaning and extensions to the model. If one tries to copy records containing NULL values into destination columns, specified as NOT NULL, errors occur. All errors are logged properly, if enabled, see parameters below for further details about logging errors.

In general all “well known” relational data types can be copied from any database to any database using this plugin. If one wants to copy custom data types, one can develop special plugins. This implementation uses getObject() and setObject() methods, which, in general, should be able to detect automatically the source type and destination type. Conversion is done automatically. If for example a destination column is declared as NUMERIC and the source contains VARCHAR – but only numeric values are within this VARCHAR column – data can be copied without any additional manual steps required. If a conversion cannot be done, an exception occurs and is logged.

Until today there was no migration project using Binary Large Objects or other complex data types. openDBcopy has not yet been tested on such types yet. It may well be, that such types cannot be copied with the current release. But finding a way that solves it should not be a problem.

If one requires to merge fields during migration one has to develop an own plugin to migrate such data. But believe me, it is simple – 2 or 3 lines of additional code to the standard copy plugin described here.

14.2 Parameters

The following parameters can or must be specified.

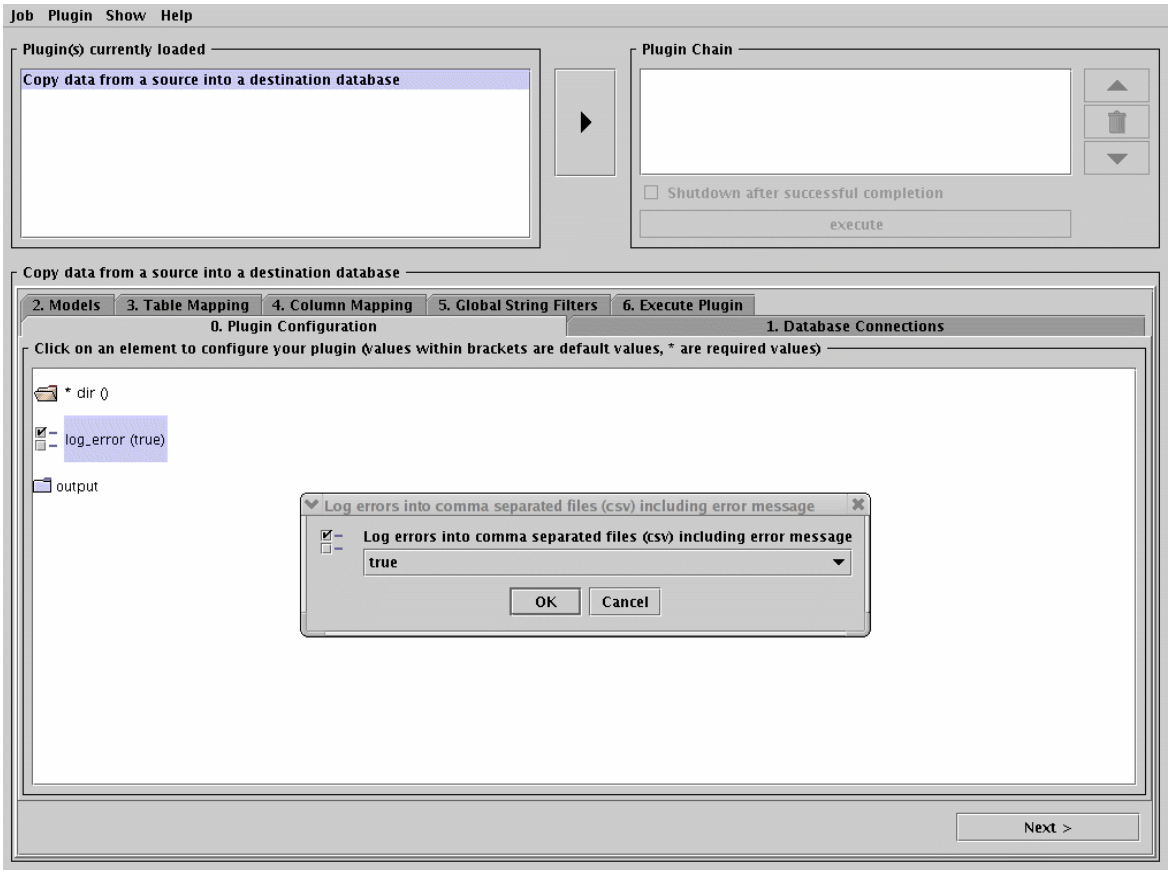
Name	Value	Required	Description
dir	path	yes	path (relative or absolute) to store records which could not be copied. For each table, an individual Comma Separated Value File (CSV) is created, if errors occurred, containing the record data and failure message
log_error	true / false	no	Default is true. If false, invalid records are not logged.
filelist	error_logs	no	If errors occurred, this filelist is created as plugin output.

14.3 Prerequisite

- An existing source database schema / catalogue to copy data from
- An existing destination database schema / catalogue to copy data to

14.4 Configuration

The only configuration to change is the output folder for possible error reports. The parameter is called dir.

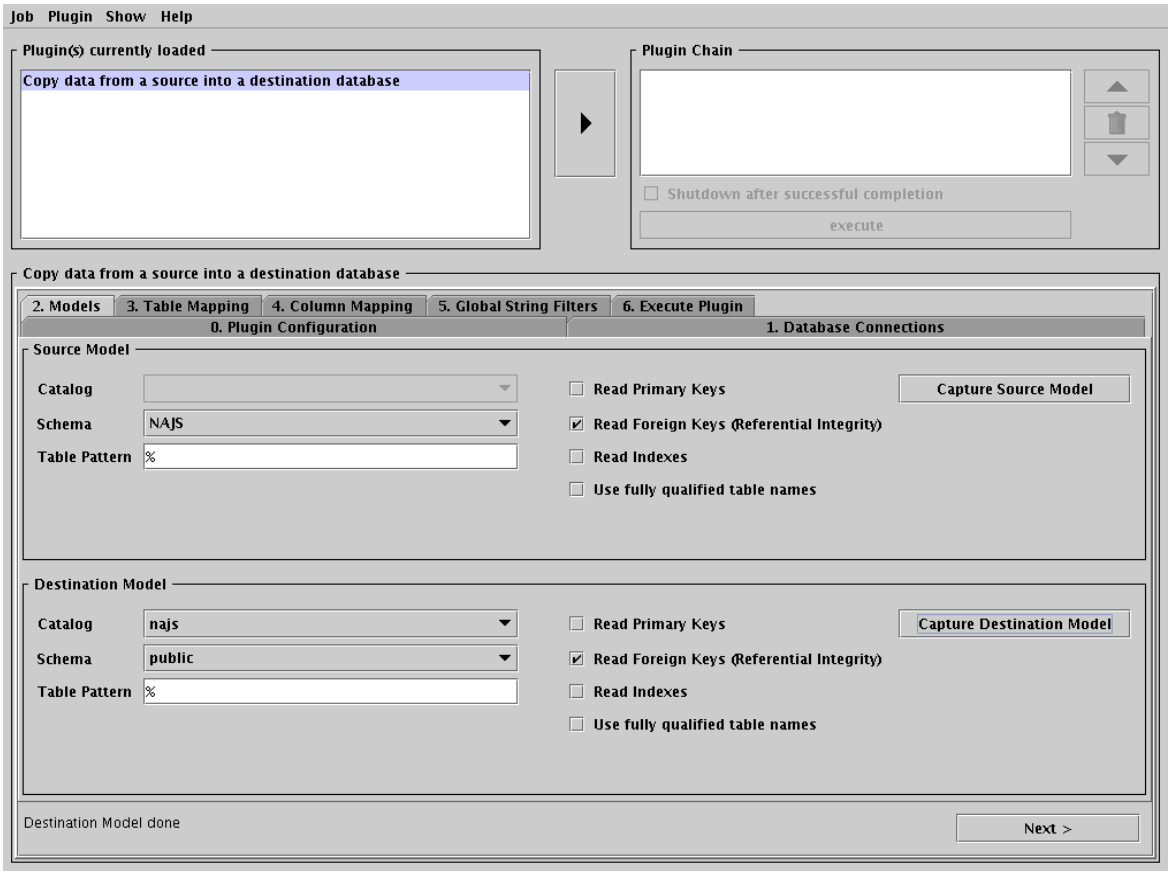


14.5 Database Connections

Specify the source and destination database connection details. Click on Apply & Test to save your settings and test the connections. Please note that you cannot continue if one or both connection test (s) was / were unsuccessful.

14.6 Database Models

The following options allow you to specify what schema / catalogue shall be captured. See the general documentation about available options and descriptions. The example used for this plugin looks like:



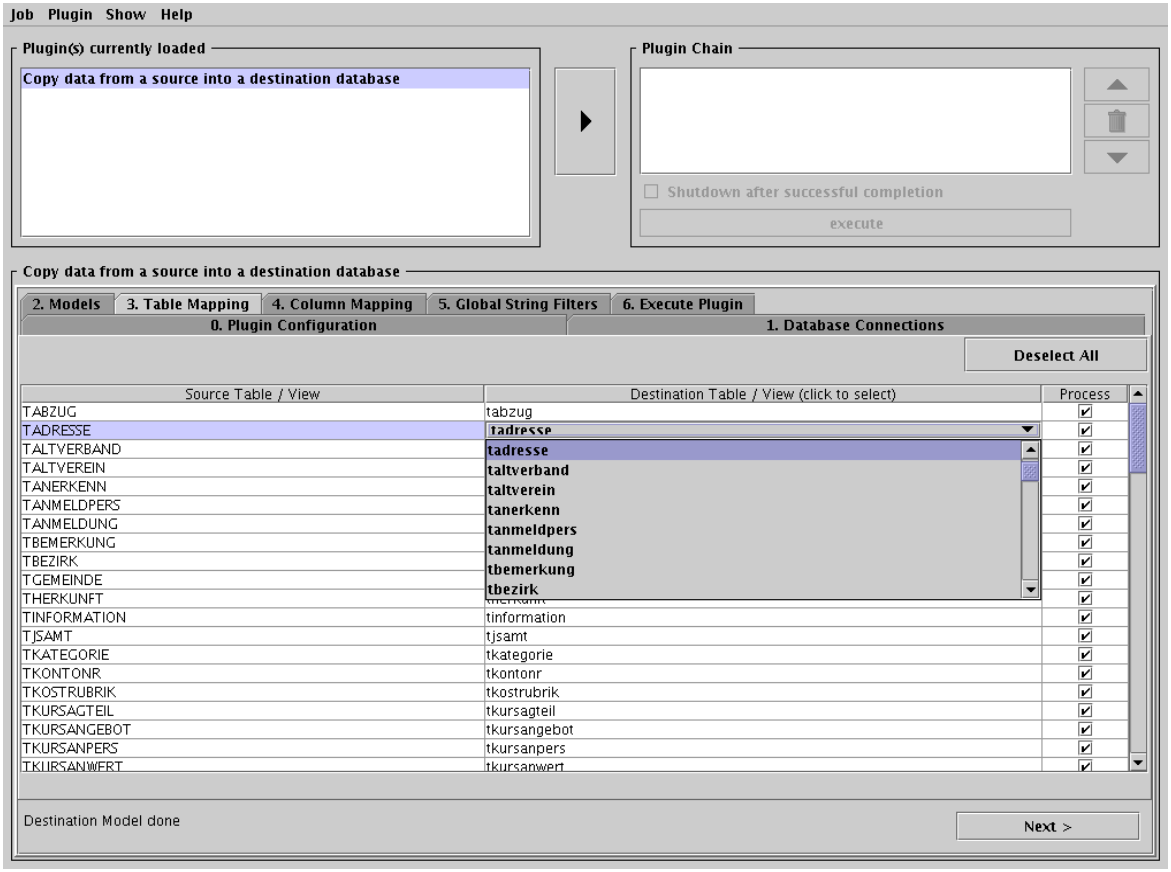
14.7 Table Mapping

In the leftmost column all tables and / or views are listed which were read when capturing the source database model. The second column shows available tables / views from the captured destination model. By default openDBcopy tries to automatically map a destination table / view to a given source table / view. Thereby it does not matter if the source table / view is written using small or capital letters – or mixed mode.

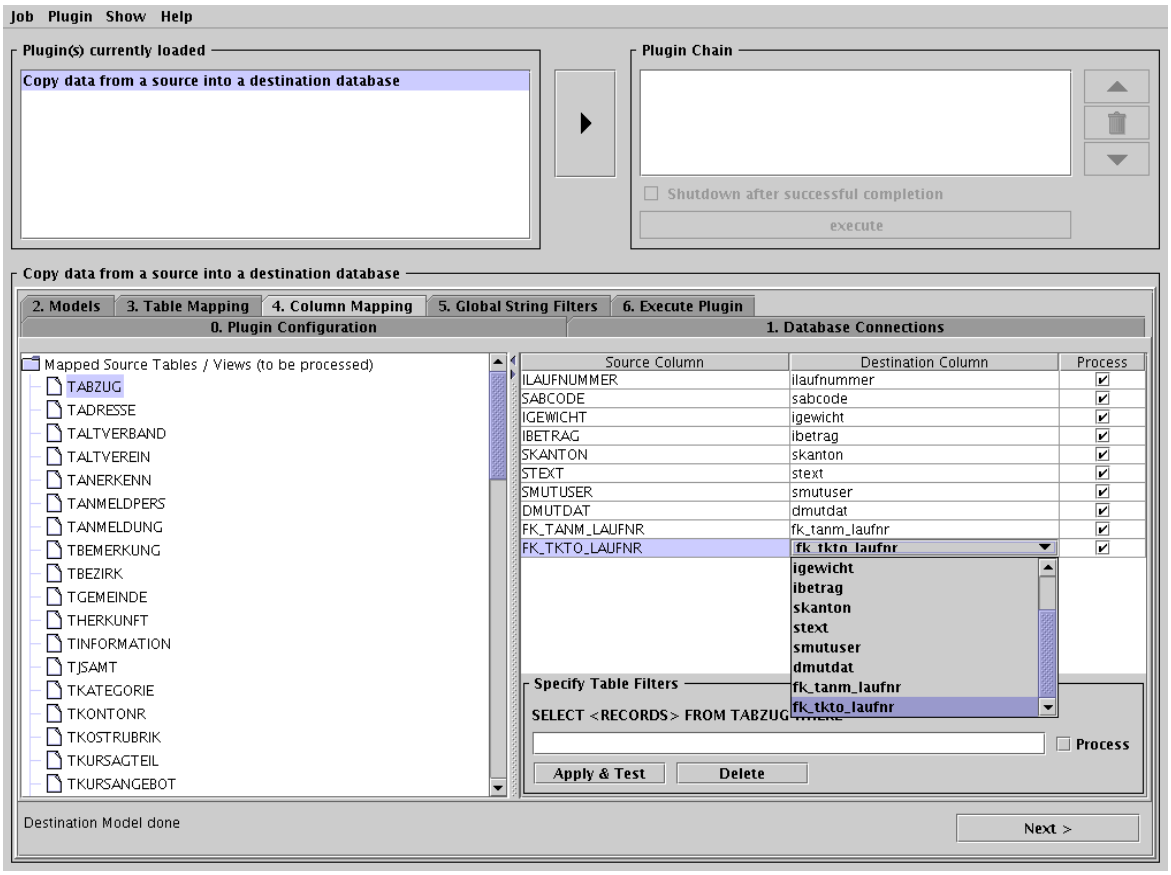
Fields which do not contain any entry in the second column, the mapping column, could not be automatically mapped. To map a certain source table / view to a destination table / view, click on the appropriate field and select a destination table / view from the drop down list provided.

Tables / Views, which were automatically mapped, can also be changed by clicking on the appropriate field and by choosing another destination table / view.

The last column specifies if the mapped source / destination table shall be processed. One can "Deselect All" tables / views or "Select All" once all rows have been deselected.



14.8 Column Mapping and Record Filtering



Click on a table / view in the tree navigation on the left hand side to see available source columns of selected table / view. Columns of the destination table / view, which could be mapped automatically (case insensitive), are listed in the second column next to the appropriate source column and are automatically checked to be processed.

Again one can change already mapped destination columns by clicking on the appropriate field. A drop down list appears.

If no destination column could be associated, please select one from the appropriate list when clicking on a field in the destination column, if required.

By default all records of a source table / view are being processed once one executes a database plugin. To filter records specify an SQL compliant WHERE statement in the appropriate table filter text field.

E.g. to filter records which have an internal id (field ILAUFNUMMER) greater than 200'000, enter

ILAUFNUMMER > 200000

and click on Apply and Test.

If the test is successful, the number of records which will be processed is shown next to the table filter buttons. If an error occurs there must be an error in the filter specification.

Again a filter can be set as “Process” or not. If applying and testing were successful, the check box is automatically selected.

The following SQL filter commands can be used:

Logical Operators:

```
>, <, =, AND, OR
```

Ordering Operators:

```
ORDER BY
```

(handy to reorder the records in the destination table)

e.g.

```
ILAUFNUMMER > 200000 AND ILAUFNUMMER < 300000 ORDER BY ILAUFNUMMER
```

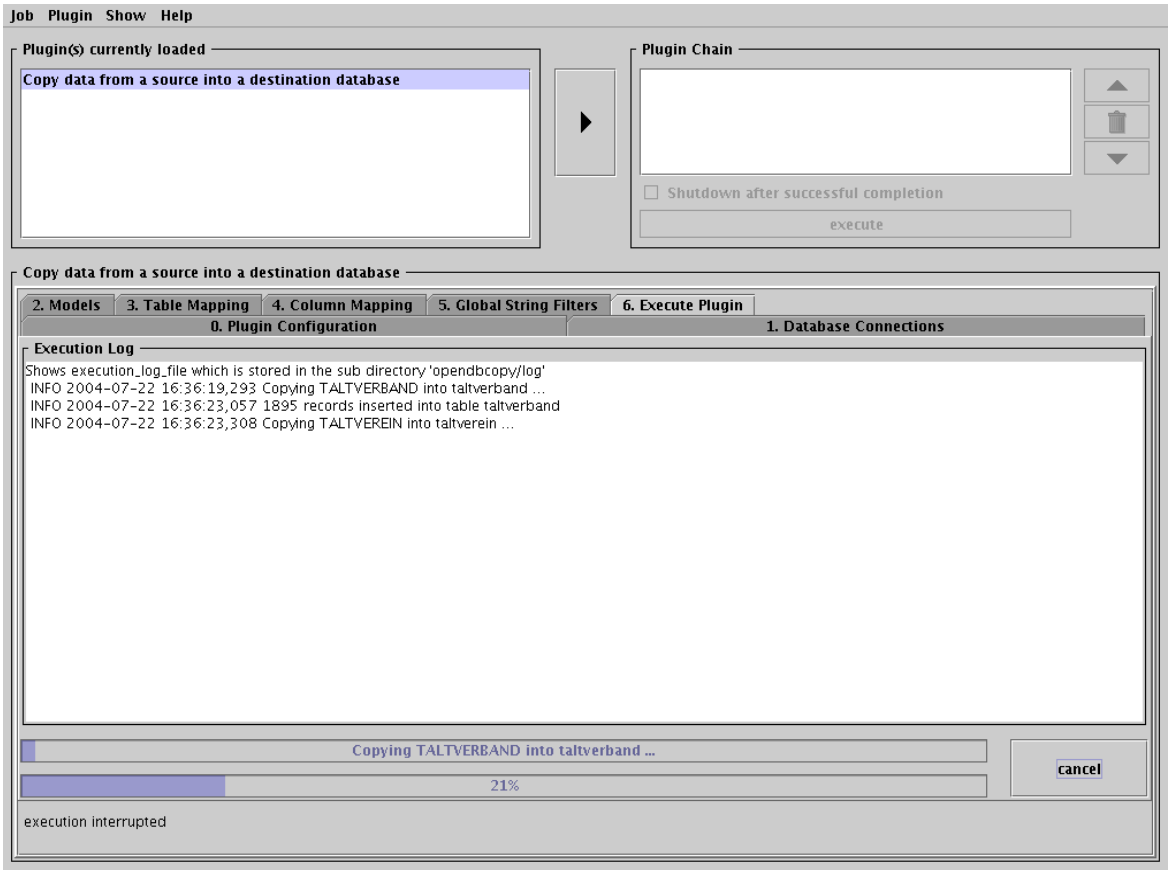
14.9 Global String Filters

Currently openDBcopy provides the following string filters

Filter Name	Description
Trim Strings	Remove spaces at the beginning and at the end of a string (any string compatible database column type such as CHAR, VARCHAR, TEXT, etc.)
Remove Multiple intermediate White Spaces	Having a string field in database column, e.g. a street number followed by street name and there is more than one white space between the two “texts” in the same field, all white spaces except one are removed.
Set NULL if String is empty (if column is nullable)	If a table provides string compatible fields which are nullable (containing nothing than SQL NULL) and the source field once trimmed has length = 0 (an empty string), this filter inserts SQL NULL into the destination field instead of an empty string. Use this filter to re optimize your string indexes again.

14.10 Execute Plugin

Once ready for copying click on Execute on the last tab. Progress and possible error files are displayed. The execution log file contains the full log of this plugin in ASCII format.



Operation progress is shown at the bottom. To cancel the copy process click Cancel. A commit to the database is done once all records of a given table / view are successfully copied to the appropriate destination table. Please note that not all JDBC drivers support commit, roll-back features. E.g. using MySQL each record inserted into the destination table is automatically committed.

If you test a copy process several times, use the delete plugin to clean destination tables again. These two plugins can easily be used together in a job. E.g. a batch job copies certain table data once a night to a remote database. But first it always deletes the appropriate destination tables.

15 Plugin: Delete selected Tables

15.1 Description

Use this plugin to delete all records of selected tables.

15.2 Prerequisite

- An existing database schema / catalogue

15.3 Database Connection

Specify a database connection.

15.4 Source Model

Select the schema / catalogue to capture. Before one can select the tables to delete, one has to let openDBcopy capture available tables.

15.5 Process Tables

Select the tables to delete data from.

15.6 Execute Plugin

Delete all records of selected tables by clicking Execute. The log displays how many records were deleted by table.

16 Plugin: Write Record Statistics to File

16.1 Description

Use this plugin to generate statistics into a comma separated value file. This CSV file can be opened with all editors, e.g. Open Office Calc, Microsoft Excel, ...

The statistics show the number of records per table, records over all tables and calculates the record number differences between tables in the source and destination database. Tables existing only in the source or destination database are tagged as UNMAPPED.

To create statistics of a single database, enter the same connection details for source and destination database.

Use this plugin as a basis for other statistical plugins. E.g. to create a different output format, one only has to modify the second part of the plugin, the data is already read and pre-calculated.

16.2 Parameters

The following parameters can or must be specified.

Name	Value	Required	Description
file	path / file name	yes	path and file name (relative or absolute) to store the generated comma separated value file
file_type	csv, ...	yes	File type ending

16.3 Prerequisite

- An existing source database schema / catalogue
- An existing destination database schema / catalogue or source again

16.4 Configuration

Specify a file location for the statistics. Keep the file ending csv (comma separated values).

16.5 Database Connection

Specify a database connection.

16.6 Database Models

Select the source and destination schema / catalogue to capture.

16.7 Execute Plugin

Once the basic statistics are read, this plugin does some calculations and finally generates a comma separated statistics file under the location specified.

Below an example statistics file.

Created by opendbcopy statistics on Thu Apr 08 13:00:11 CEST 2004

SOURCE	RECORDS	DESTINATION	RECORDS	+ / -
accessright	4	accessright	4	0
dictionary_block	8964	dictionary_block	8964	0
dictionary_part_of_speech	29	dictionary_part_of_speech	29	0
dictionary_permission	5	dictionary_permission	5	0
dictionary_recordtype	4	dictionary_recordtype	4	0
directory_permission	7260	directory_permission	7260	0
ics_keycode	1515	ics_keycode	1515	0
ics_keyname	1515	ics_keyname	1515	0
ics_organisation	7805	ics_organisation	7805	0
ics_system	0	ics_system	0	0
language	12	language	12	0
language_name	24	language_name	24	0
location	261	location	261	0
location_name	261	location_name	261	0
location_name_mapquest	236	location_name_mapquest	236	0
logo	11	logo	11	0
logo_status	3	logo_status	3	0
organisation_location	8093	organisation_location	8093	0
organisation_user	7349	organisation_user	7349	0
statistics	7	statistics	7	0
status	6	status	6	0
system	2	system	2	0
user	0	user	7405	-7405
user_system	7262	user_system	7262	0
TOTAL	50628	TOTAL	58033	-7405
UNMAPPED		dictionary_resources	0	
UNMAPPED		organisation	0	

17 Plugin: Dump Data into Flat Files (csv, txt, ...)

17.1 Description

Use this plugin to dump data of selected tables and columns into flat files. The flat file format can be chosen. The value separator is an important parameter. A future plugin maybe will provide the ability to create csv, txt, xml as data format for dumps and imports.

17.2 Parameters

The following parameters can or must be specified.

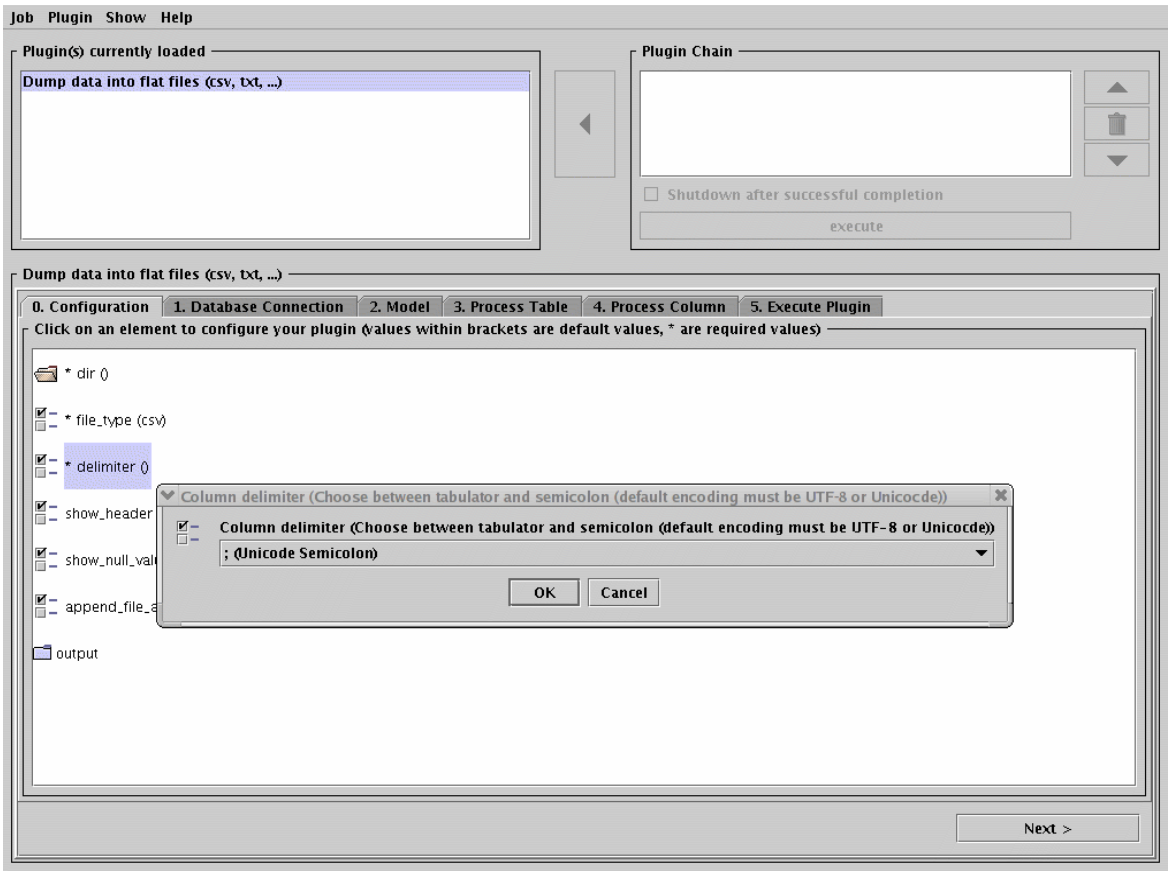
Name	Value	Required	Description
dir	path	yes	path (relative or absolute) to store dump files
file_type	csv, txt, ...	yes	Select the file type (csv, txt, ...)
delimiter	\t, ;, ...	yes	Currently one can only select between tabulator or semicolon ;. Other delimiters must be directly entered in the plugin.xml of this plugin.
show_header	true / false	no	Default is true. If true, column names are shown in the first row
show_null_values	true / false	no	Default is false. If false, null values are replaced by an empty value. If true, null is displayed.
append_file_after_records	a positive number	no	Default is 500. After a maximum of 500 records read, data is written to the appropriate file first before continuing reading. Use this parameter to control speed. Depending on the number of columns of a certain table and available memory select a number. Enter a high number (> 1000) if your tables are narrow and a lot of memory is available. Enter a low number (< 500) if your tables are wide (many columns) and memory is rare.

17.3 Prerequisite

- An existing source database schema / catalogue to dump data from

17.4 Configuration

Specify a path and column delimiter. Other options have default values or are optional.



17.5 Database Connection

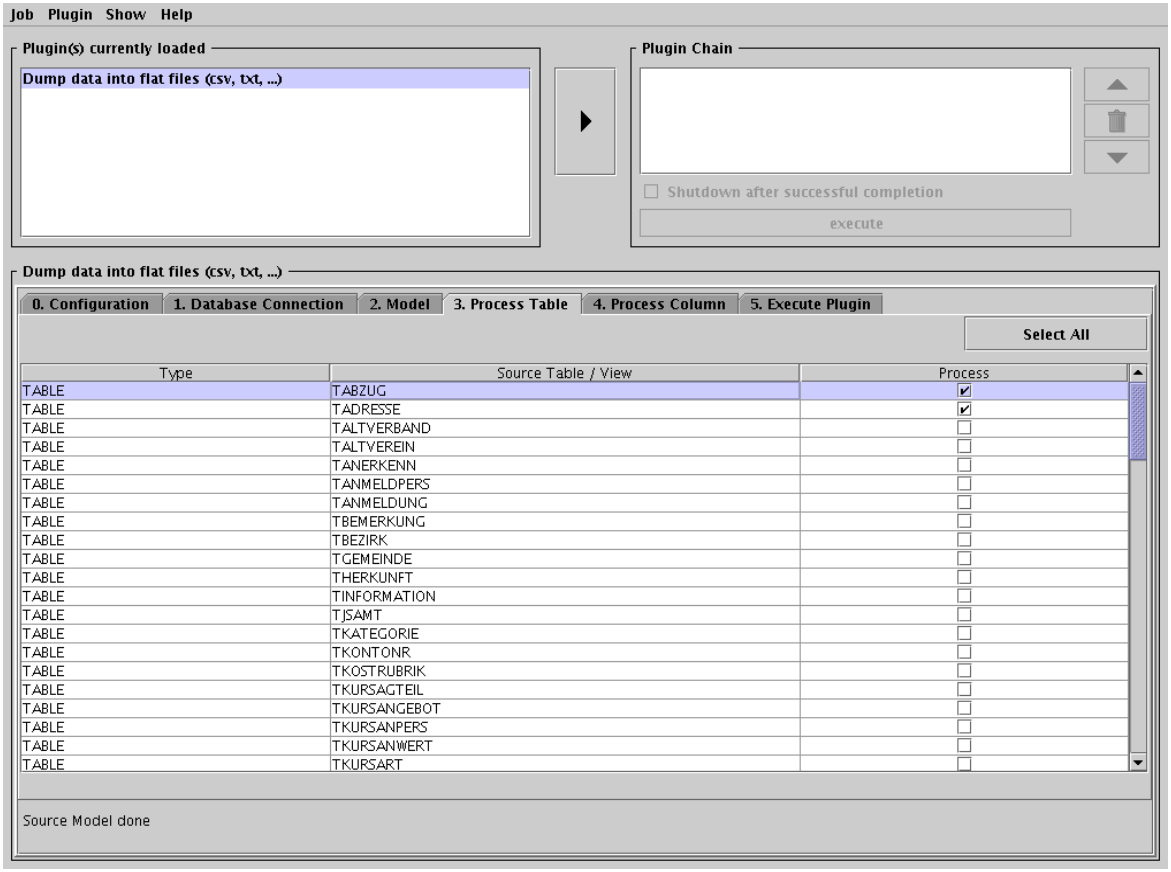
Specify a database connection.

17.6 Source Model

Select the source schema / catalogue to capture. No additional options are required.

17.7 Process Tables

Select the tables / views to dump.

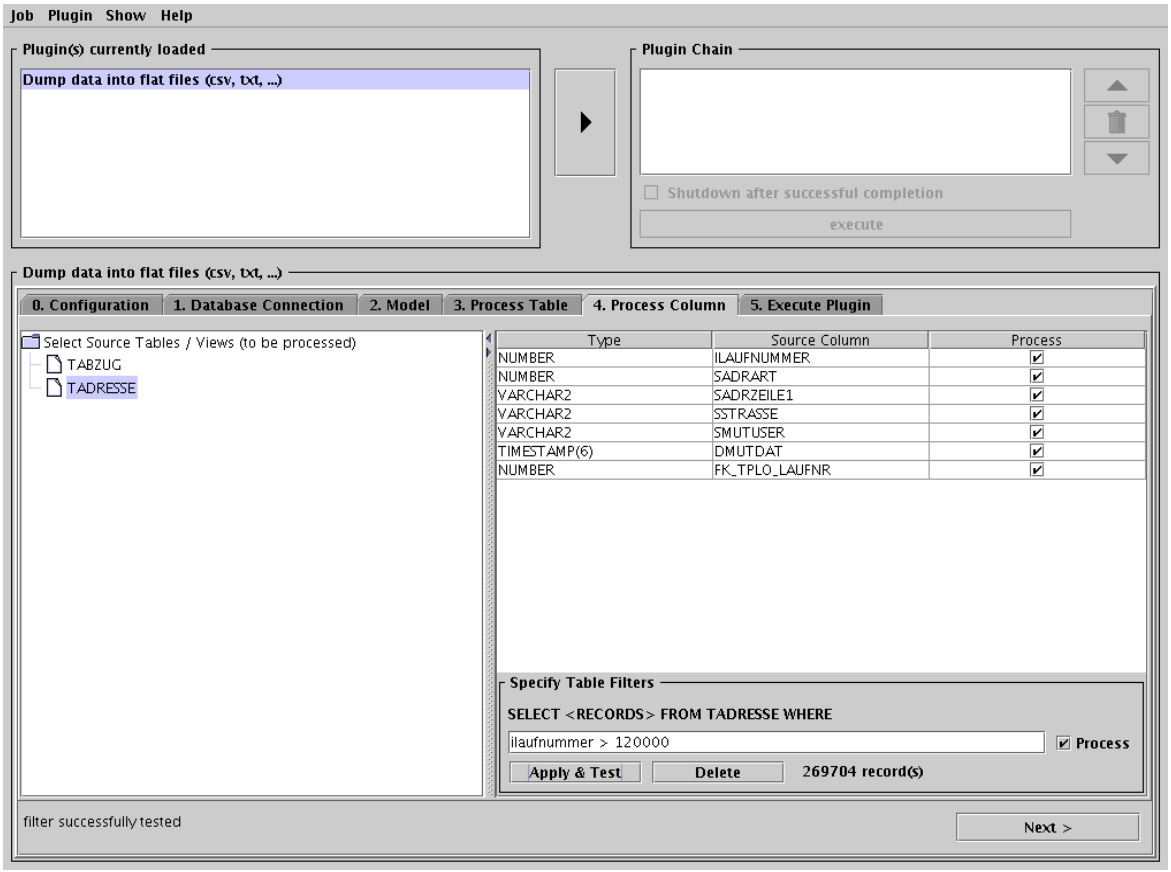


17.8 Process Columns

Select the columns to dump. In the following example a table filter is specified to narrow the number of records dumped. Specifying a filter

```
ILAUFNUMMER > 120000
```

results in 269704 records displayed when clicking on Apply & Test. This number shows how many records will be dumped of a table once executing the plugin.



17.9 Execute Plugin

To dump all selected tables / views into files, click on Execute on the last tab.

18 Plugin: Create Insert SQL Scripts

18.1 Description

Generate SQL Insert scripts for a destination database, given a source database, which may be a different RDBMS. Given the source and especially destination JDBC driver, this plugin is able to escape all data according to its JDBC specification. If referential integrity is supported and enabled, scripts are numbered so that an administrator executing the resulting INSERT SQL scripts can execute the scripts in the right order – according to referential integrity rules.

Another big advantage of this plugin is the ability to create valid SQL Insert scripts for a different destination database system, using the data of the old source database system. In the destination database a table or even columns may have different names or different column ordering within a table.

18.2 Parameters

The following parameters can or must be specified.

Name	Value	Required	Description
dir	path	yes	path (relative or absolute) to store SQL Insert Script(s)
show_qualified_table_name	true / false	no	If true, the table names within generated scripts are fully qualified, e.g. schema_name.table_name or catalogue_name.table_name. If false, only the table name itself is inserted into the script(s).
filelist	scripts	no	Output of this plugin

18.3 Prerequisite

- An existing source database schema / catalogue to read data from
- An existing destination database schema / catalogue

18.4 Configuration

Specify the directory where this plugin shall store generated SQL scripts. The second option, show_qualified_table_name should be enabled if required. Certain RDBMS do not support this option, others require it. For example the user executing the script is not the user of the destination schema.

18.5 Database Connections

Specify the source and destination database connection.

18.6 Database Models

Select the source and destination schema / catalogue to capture. Enable referential integrity if supported by the JDBC driver and RDBMS so that scripts generated can be numbered according to referential integrity rules.

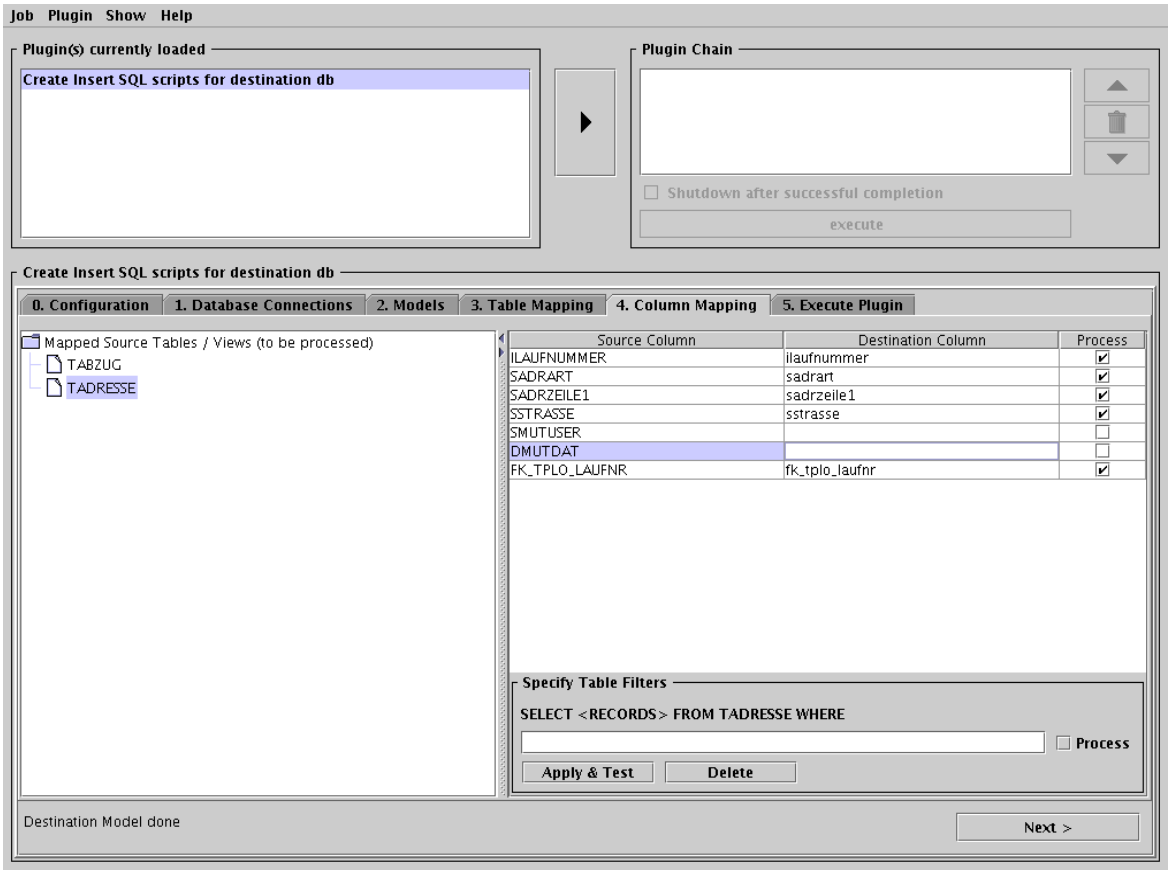
18.7 Table Mapping

openDBcopy tries to map source to destination tables. See Copy Plugin for further details.

18.8 Column Mapping

openDBcopy tries to map source to destination columns. In the example shown the destination model does not contain the two columns SMUTUSER and DMUTDAT. Clicking on one of those empty fields provides a list of available column names of the destination table. Either columns do not exist or are named different in the destination table.

Use the table filtering feature to reduce the number of record data generated.



18.9 Execute Plugin

Click Execute to generate the INSERT SQL scripts for the selected tables.

19 Plugin: Create ZIP Archive Files

19.1 Description

Use this plugin to compress (ZIP) single files, directories (including sub-directories) or a filelist (a list of individual files which may be stored in different paths.

19.2 Parameters

The following parameters can or must be specified.

Name	Value	Required	Description
file_dir_filelists_selection	file / dir / filelist	yes	Select the type of input. File lists are generated as output of other plugins.
file	path / file name	no	If file was selected as input type, specify path / file name of input file. Path may be relative or absolute.
dir	path	no	If dir (directory) was selected as input type, specify path of input directory. Path may be relative or absolute.
output file	path / file name	yes	Path / file name of compressed output file. Path may be relative or absolute.

19.3 Prerequisite

- Files to compress into a compressed ZIP file. Those files may exist before a job is started or created at runtime by previous plugins of a certain job.

19.4 Configuration

Specify input type file, directory or filelist.

If file or directory were selected, specify a file or directory.

Do not forget to specify the output file – the compressed ZIP archive.

19.5 Execute Plugin

Click Execute to compress the specified input. The ZIP archive can be opened by any tool supporting ZIP compression. The plugin uses Maximum Compression option to keep the compressed archive file size as little as possible.

20 Plugin: SSH Copy (scp, sftp), Secure Remote Execution

20.1 Description

Use this plugin for SSH operations. Currently only Secure copy from a local location to a remote host is implemented. The plugin uses the Open Source Library JCraft, see http://www.jcraft.com/jsch.

JCraft is a pure Java implementation of SSH2. Apache Ant 1.6 uses it for sshexec and scp tasks. Eclipse 3.0 uses it for SSH2 access to CVS repositories.

See the JCraft website for a complete list of supported features. Some of those will be implemented into this plugin by wrappers, using the Open Source JCraft library.

20.2 Parameters

The following parameters can or must be specified. For typical values see the screen shot on the next page.

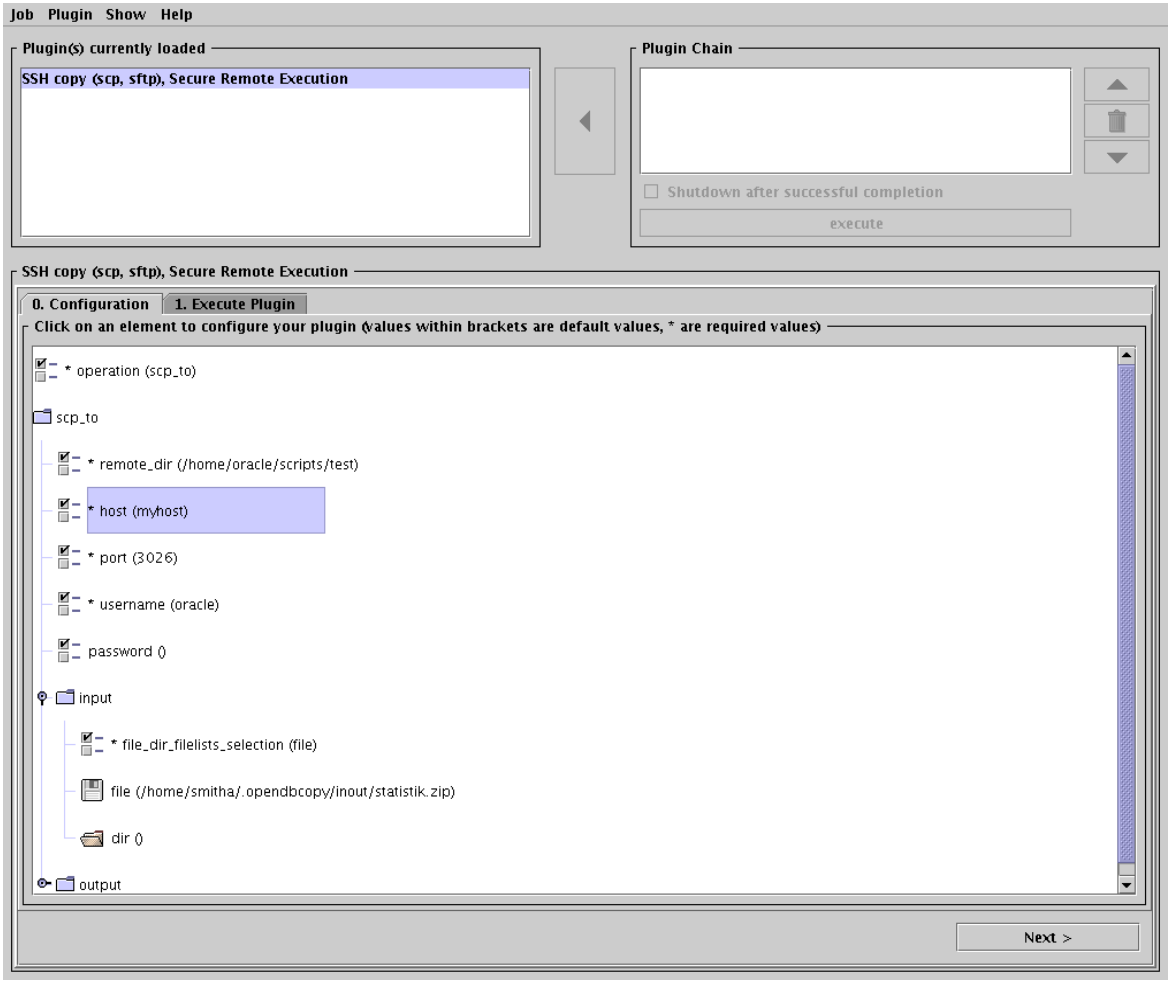
Name	Value	Required	Description
operation	scp_to	yes	Currently this plugin only supports **secure copy to a remote host**. Other features will be implemented in the future, if requested and required.
remote_dir	absolute destination path	yes	Absolute destination path. Syntax depends upon operating system.
host	IP Address or Host Name	yes	Host Name or IP Address
port	Port Number	yes	Port Number
username	User name	yes	The user name required to connect to the remote host
password	Password	no	Please note that the password is shown and saved in clear text within the plugin configuration
input file_dir_filelists_selection	file / dir / filelist	yes	Select the input type. Single file, a directory or filelist
input file	path / file name	no	If file was selected as input type, provide path / file name to the input file located on the local machine. Path may be relative or absolute.
input dir	path	no	If dir was selected as input type, provide path to the input directory located on the local machine. Path may be relative or absolute.

20.3 Prerequisite

- Existing local file(s) / directory to copy
- Access to remote host and write permission on remote directory

20.4 Configuration

Use the table above and tool tip texts for further details.



20.5 Execute Plugin

Click execute to securely copy the selected input file, directory or filelist to the remote host and directory.