Power*Architect User Guide

SQL Power [http://www.sqlpower.ca] Group Inc.

Power*Architect User Guide

SQL Power [http://www.sqlpower.ca] Group Inc.
Copyright © 2006 SQL Power Group Inc.

Table of Contents

List of Tables

Chapter 1. Introduction

Data Architects, DBAs, Analysts and Designers rely on state-of-the-art Data Modeling Tools to facilitate and simplify their data Modeling efforts, while maximizing the use of their resources. The Power*Architect software allows these busy technical professionals to perform this most intricate part of their job in a fraction of the time.

SQL Power Group's Power*Architect is an innovative data modeling tool designed primarily for Data Warehouse and Data Mart design. It allows the designer to open multiple concurrent source Database connections, drag and drop source schemas, tables and columns into the data modeling playpen, and forward-engineer the resulting target database and its associated ETL template.

The Power*Architect is a user-friendly DW data modeling tool created by data warehouse designers, and has many features geared specifically for the data warehouse architect, including:

- Access any JDBC- or ODBC-accessible source database;
- Design every aspect of the target database Data Model;
- Compare the database structure of any two databases, highlighting the structural differences and generating the required DDL to synchronize;
- Compare the Data Model data structures to an existing target database; Save a snapshot of all source systems' data structures in the project file, allowing data warehouse designers to evolve their target data model remotely;
- Forward engineer to Oracle, SQL Server, DB2, PostgreSQL and other databases;
- Forward engineer ETL Templates containing source-to-target data mappings;
- Invoke ETL Engine to load initial set of data into the target database;
- Enable easy centralized installation and updates to multiple end users (using Java WebStart™ technology).

Power*Architect can open multiple source databases concurrently, even those from competing database vendors. Another key feature of the Power*Architect that sets it apart from other data modeling tools is that it remembers the origin of each column, and is capable of automatically generating the source-to-target data mappings.

Whether you're building or maintaining your Data Warehouse data model, the Power*Architect will provide you a complete view of all required database structures and will expedite every aspect of your data warehouse design.

We firmly believe you can...

> Design your Data Warehouse in a fraction of the time with Power*Architect.

Power*Architect is a versatile tool for the busy data warehousing practitioner.

Basic Concepts

Project - a Power*Architect project consists of a view of multiple databases; you can load and save a Project to work on it at leisure.

Driver - Most programs need a distinct driver program to communicated with each different type of database. Power*Architect uses Java-based drivers, which normally come from the database vendor in the form of "JAR" [1] files. You need to inform Power*Architect about each driver before you can use it; do this from the User Preferences panel, under JDBC Drivers (just click Add and browse to the Jar file for your driver). If you do not have the JDBC driver for a given database, you can usually obtain one from the database vendor. If that fails, you can find a directory of databases drivers on Sun's web site [http://developers.sun.com/product/jdbc/drivers] .

Playpen - This is the main area of the Power*Architect window, in which you manipulate tables and relationships. You can play here to your heart's content, knowing it will not be saved until you ask the program to save.

Who this book is for

This book is a step-by-step guide on how to use the full capabilities of Power*Architect . It covers topics from how to install the Power*Architect through setting up database connections to engineering your data model.

We assume you are familiar with basic database terms. If you meet any terms that are unfamiliar, please refer to the Glossary at the end of this book.

This book also assumes you are familiar with basic computer operations.

We also assume you have SQL Power's Power*Architect software installed on your computer; if not, please refer to the Installation Guide below.

[1] Java Archive files; these are an extension to the file format used by PKZip/WinZip archives

Chapter 2. Installation Guide

Power*Architect Prerequisites

To run the Power*Architect you need a Java 1.5 or newer Java Runtime ("Java VM" or "JVM"). A current version of the Java VM for common platforms can be obtained from Sun Microsystems [http://java.sun.com/javase/downloads/index.html] . To ensure that your JVM is up-to-date, Apple Macintosh users should use Software Update (from the Apple Menu) while others should use the Java Updater (from, e.g., the Microsoft Windows Control Panel).

How to Obtain the Power*Architect Software

Power*Architect can be obtained from the download section [http://download.sqlpower.ca] of the SQLPower Website [http://www.sqlpower.ca/] . You should only need to download one file, choosing the platform-appropriate distribution (Windows-Installer for Microsoft Windows, "DMG" for Apple Macintosh, and ".tar.gz" for UNIX/Linux/other platforms). You should normally choose the download with the highest revision number available.

How to Install Power*Architect

Microsoft Windows

Double click on the Architect-setup-Windows-n.m.jar. This will launch the Microsoft Windows installer. Follow the on-screen instructions.

Macintosh OS X

Drag the architect-n.m.dmg file to the Applications folder

Other Platforms

Extract the Architect-generic-n.m.tar.gz package into the desired directory.

How to Run Power*Architect

Microsoft Windows

From the start menu, select All Programs. Then select the Power Loader Suite program group. Finally click on the Power Architect shortcut.

Macintosh OS X

From the Finder, select Applications, then select Power*Architect. To enable launching of the Architect directly from the Dock, either drag the image there or, while it is running, Apple-Click on the running icon and select Keep In Dock.

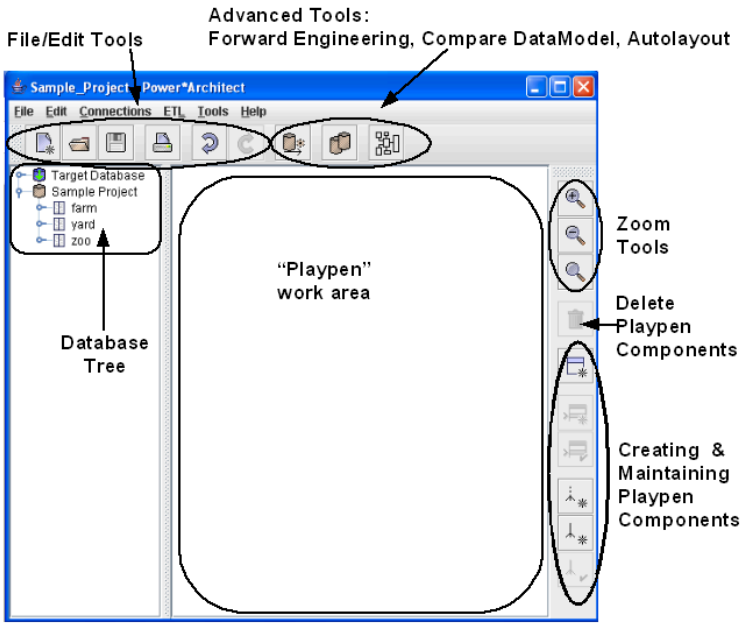
Other Platforms

In the directory into which you extracted the Architect, run the command **java -jar architect.jar** . If you have a visual browser, you may be able to launch the architect by clicking (or double-clicking) on the architect.jar file.

Chapter 3. How to Use the Power*Architect

Power*Architect User Interface

When you start the Power*Architect, you will see the Project window, shown below, which is the main view area and starting point for actions.



Database Tree - This is where you can add, maintain and explore imported connections. It uses a tree-node dropdown method. Therefore to explore inner components, you can expand components within this container as needed.

Playpen - This is the main area of the window, in which you manipulate tables and relationships. You can play here to your heart's content, knowing it will not be saved until you ask the program to save.

Playpen Components - These are the components that can go into the playpen. The playpen components are Tables and Relationships.

Zoom Tools allow you to control the magnification level of components display

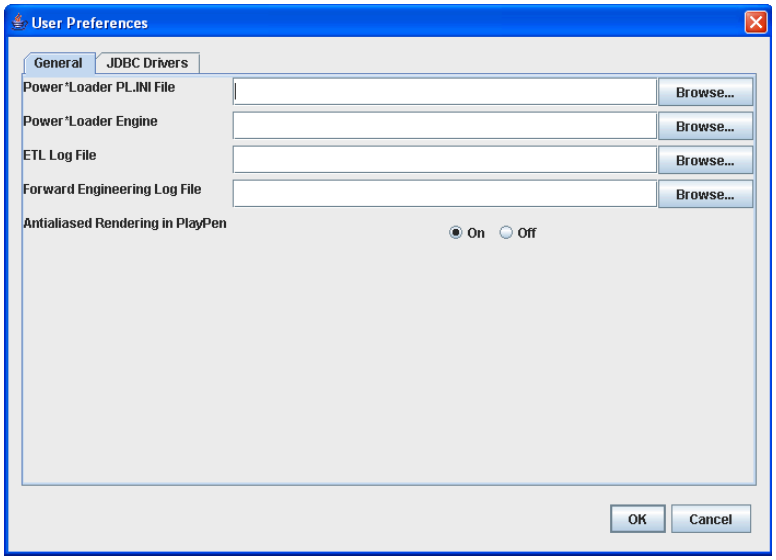
Delete will delete the selected component(s).

Create/Maintain Playpen Components is discussed in Using Components below.

Setting the User Preferences

When getting started, you need to set up some files and drivers to use the full functionality of the Power*Architect. If you have not already done so already, please go to "User Preferences" under the File

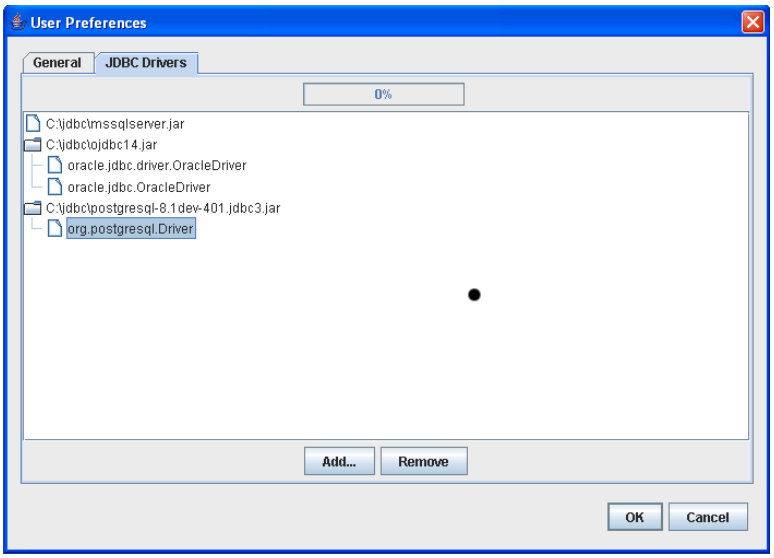
Menu to configure the Power*Architect. This will pop up a dialog (shown below) where you can specify your file locations.



The pl.ini file stores the database connections that you set up (see JDBC Drivers on page 8). If you leave this location blank, the Architect program will prompt you to use a default location when you start it up. If you have a pl.ini file from other SQL Power applications you should generally use the same file, as doing so will save you from having to re-enter all your database connection information. The Power*Loader Engine setting is optional; if you are not using the Power*Loader engine you can leave this blank; if you are, you should set it to the location where the Power*Loader Engine program is installed on your system. The next two settings are for log files that will be written when you use the ETL and Forward Engineering functions of the application. The final setting controls the operations of graphics in the PlayPen. Turning "antialiased rendering" on may give better display of the graphical database layouts shown in the PlayPen, but may use more CPU time in the process.

JDBC Drivers

Besides setting up these file locations, you must also tell the Power*Architect about the JDBC drivers you wish to use. JDBC Drivers are needed to access most databases, and are distributed in "JAR file" format. Click on the "JDBC Drivers" tab and click Add... to specify the location of a JDBC driver jar file; if it is valid, the system will list the names of any Driver classes found in it, as shown below (note that "ojdbc" stands for Oracle's JDBC driver, whereas "ODBC" is Microsoft's technology for database access).



Note: The tie-in between Drivers and Connections might seem a bit indirect if you are not used to using Java JDBC. In this section of the User Preferences you are simply telling the Power*Architect to load these drivers into memory when it starts up. When you set up an actual Connection, you implicitly tell the software which driver to use by the second field of the "db URL" as described on page 21. If you need more information on JDBC drivers, please refer to the Basic Concepts section of this manual on page 3.

A Brief Example

This section will show you how to set up [1] a simple database "from scratch", just to get you started using the tools, without modifying any live data. If you follow the example literally, you will create a trivial "customer and orders database".

1. Setup Driver. Select File->User Preferences and select the JDBC Drivers tab. If there is already an entry for the driver you wish to use,, click OK and go on to the next step. Otherwise, click the Add button, navigate to where you have the driver Jar file installed, and click OK.

2. Create a Connection. In the Database Tree section of the main window, right click and choose Add Source Connection->New Connection. For this example you can use a name like SampleDB, for both the Connection Name and the Database name (these names do not have to be the same, but we'll keep them the same for simplicity). If you select the JDBC Driver before you type the database name, then as you type the Database name, it will be added to the DB URL, so you don't have to type it an extra time. Fill in all the fields and click OK.

3. Now right click on the Project in the Database tree, and select Set Target Database, then choose the Database connection you just created (e.g., SampleHSQLDB). Note that Add Source Connection makes a database available to copy tables from, whereas Set Target Database specifies where SQL commands will get executed to create the structure you are building in your project.

4. You are now ready to design some tables. For this example, we will create the Customer and Orders table shown here.

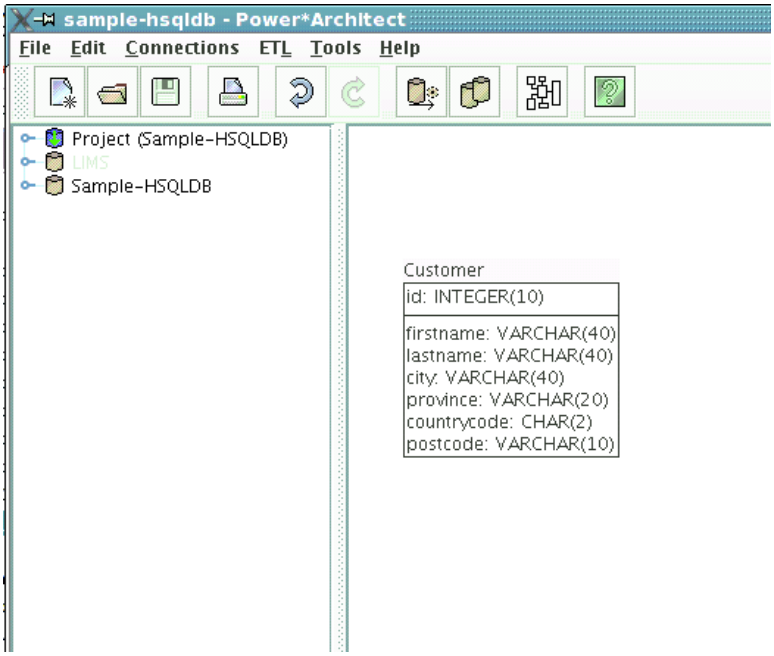
    a. Click on the New Table icon at the right side. The cursor will change to a crosshair. Move the cursor near the left of the Playpen area, and click. A "New Table" will appear.

[1] Assumes you have used some vendor-specific external tool to create a new database.

b. Double click on the title, and the Table Properties Dialog will appear. Rename this table to Customer, and the Primary Key to Customer_PK.

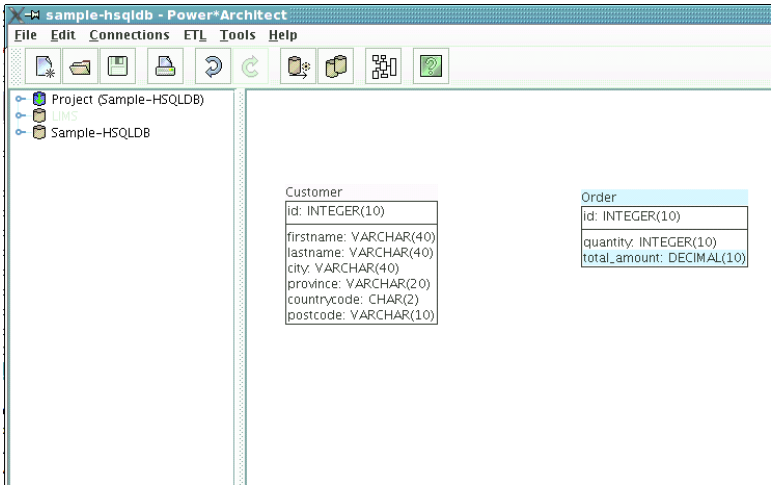
c. Click on the Insert Column icon, and a "New Column" will appear. Double-click on the column, or click on the Column Properties, and the Column Properties Dialog will appear. Rename the column to ID and make it part of the primary key.

d. Insert additional columns for Firstname, Lastname, Address, City, Province, Country Code [2] and Postal Code. The table should look something like the following:



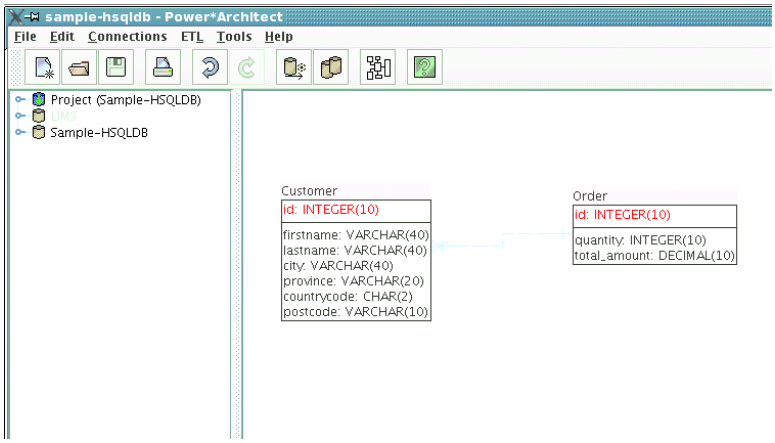
e. Create a second table, and name it Orders.

f. Create columns named Id (in the primary key), Quantity, and Total Amount. Your project should now look something like the following:

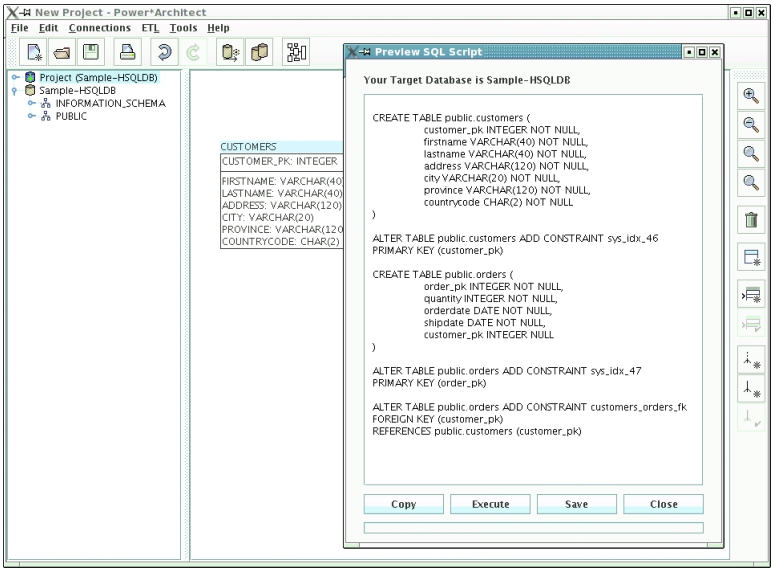


g. We need a relationship between these tables. An order should have a foreign key that refers to the customer. Click the "New Non-Identifying Relationship" icon. Select the Order table, then the

Customer table, and a link will be drawn as shown. Click on this link and the keys that take part in the relationship will be highlighted in red.



5. If you're happy with the database layout (you can always change it later), it's time to create the database. Click on the Forward Engineer button. You should see a window similar to the following:

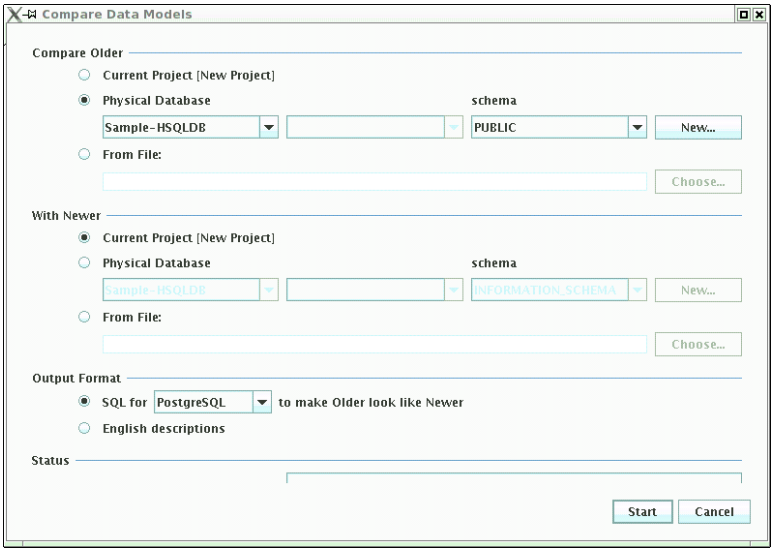


6. If this looks plausible, click Execute, and the tables and their relationship will be created. Congratulations! You have now created a simple database using the visual tools in Power*Architect. But let's not stop here. Suppose that after using this database, you realize that there should be a "shipping amount" field in the Order table (we never promised this would be completely realistic example).

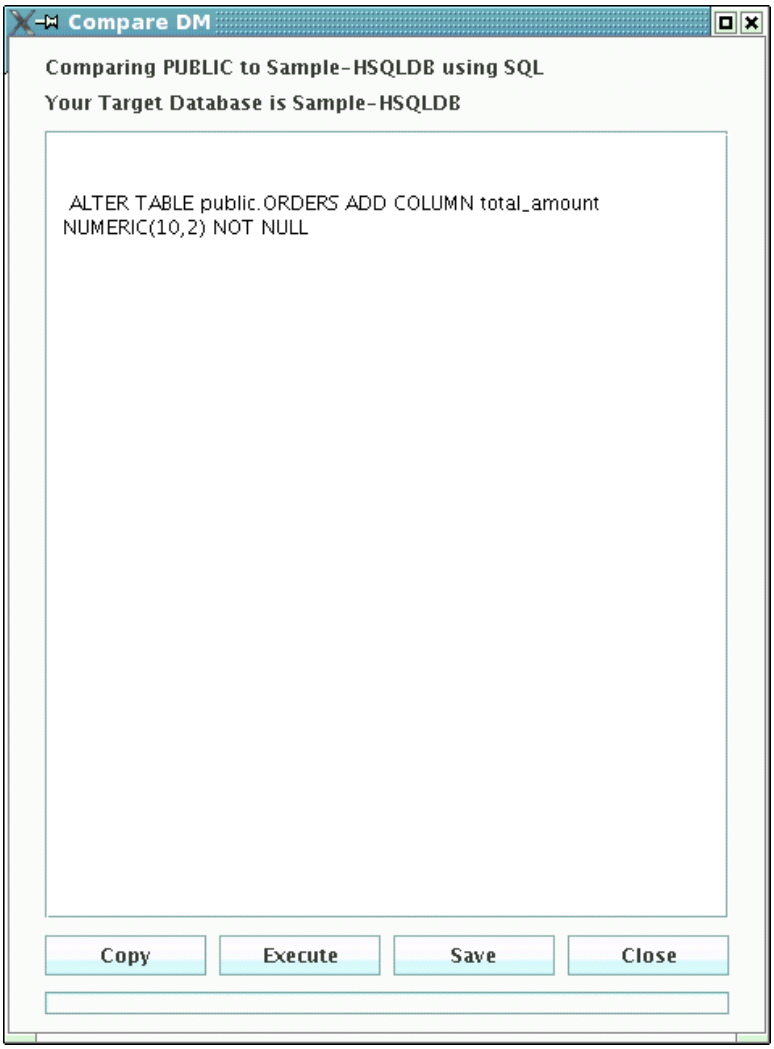
7. Select the Order table by clicking on its title.

8. Click the Insert Column field and, as before, rename the New Column, this time to Shipping_Amount. Change its type to Decimal(10,2).

9. Now we need to compare two different Data Models, the original database and the current project. Click the Compare DM icon. Set the "Older" to Physical Database SampleDB (you may need to change the Schema to Public). Set the "Newer" to "Current Project" (since it is now newer than the database you created in Step 6). Set the output format to SQL.



10. Click Start. You should see the SQL Preview window again, but this time with just an ADD for the column you just added:



11.Click Execute, and the new column will be added to your database table.

When you exit the program, it will ask to save your project. Since you might want to alter this in future, to experiment with some of the other tools without damaging any live data, you may wish to save the Project file.

The remainder of this document provides a more comprehensive explanation of the various functions that Power*Architect offers.

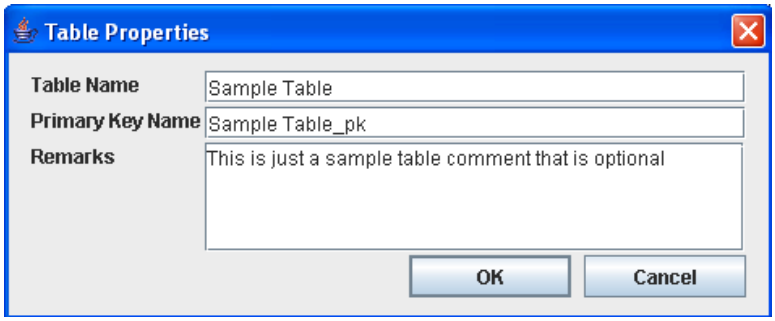
Using Diagram Components

Creating New Tables

There are several ways to create new tables in Power*Architect. The first way is to click on the New Table Icon on the sidebar menu. The cursor turns into a "+" cursor indicating the mode change. Simply click on the playpen on the spot you desire to place the new table at. The second method to create a table is to right click on the playpen and select the "New Table" option. A newly created table will then be placed at the point of the right click. A third way is to type the letter T with the mouse over the playpen.

Editing Tables

To edit a table, right click on the table title and select "Table Properties". This pops up the Table Properties dialog.



In this dialog, you can:

- Change the name of the table
- Rename the primary key section of the table
- Add comments/notes about the table

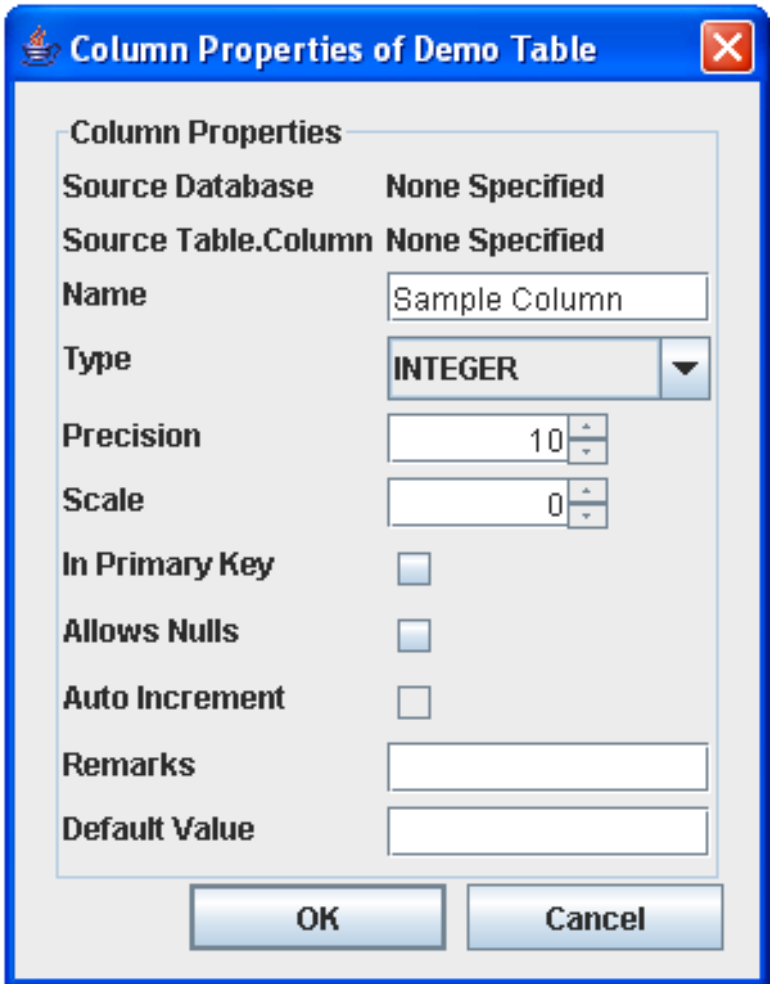
Creating New Columns

First select the table the new column will be placed in. Click on the "Insert Column" button and a column is created in the selected table. Another way to add a new column is to right click on a table and select the "New Column..." option. The new column will be added to the table below the selected column. It will be placed at the bottom of the table if no columns are selected. If a table has existing columns in the primary key and you wish to create new columns within the primary key, select a column that is already

in the desired primary key and then create a new column. The newly created column will be placed within the primary key as well.

Editing Columns

✔ Select the desired column, right click and select "Column Properties". The Edit Column Properties dialog pops up. Or you can select the column and click the "Edit Column" button on the Playpen toolbar.



In this dialog, you can:

- Rename the column
- Change the type of data the column holds
- Set the precision of the data
- Set the scale
- Indicate if the column is in the primary key or not
- Indicate if the column should handle null information or not
- Indicate if auto increment is allowed or not

- Add additional comments about the column
- Set a default value for the column

A special feature of the Power*Architect is that if a column orginated from a database, the Power*Architect is able to remember the database and table it orginated from.

Dragging Columns

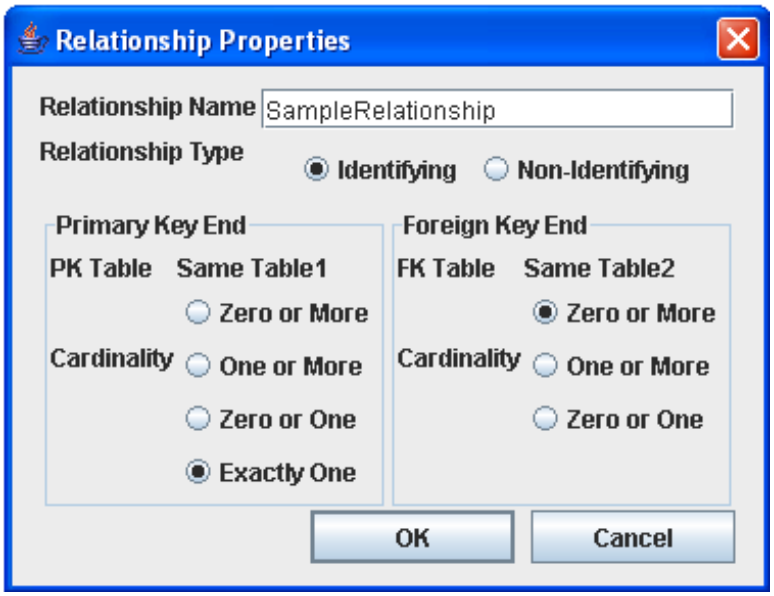
The Power*Architect allows you to drag columns within a table's key and from table to table freely. Simply select the desired columns and drag them into the desired destination. For now, only one column can be moved at a time.

Creating Non-Identifying and Identifying Relationships

 To create a relationship, select the desired type of relationship on the Playpen ToolBar. The non-identifying relationship icon is  (keyboard shortcut is shift+R). The identifying relationship icon is identified by  (shortcut key is R). The cursor changes to the "+" cursor to indicate the mode change. First click on the parent table and then click on the child table. Once this has been done, the relationship will be created, and will appear as a link between the two tables.

Editing a Relationship

 Select the desired relationship and right click on the relationship. Choose the "Relationship Properties" options. This can also be done by selecting the relationship and clicking on the relationship properties button. In both cases, the Relationship Properties dialog will appear.



In this dialog, you can:

- Rename the relationship

- Choose the relationship type
- Change the primary key end cardinality
- Change the foreign key end cardinality

Selecting Multiple Items in the Playpen

There are two ways to select multiple items in the playpen. One way is to hold down the shift key or the ctrl key as more items are being selected. The alternative method is to use the selection box.

Relocating Objects in the Playpen

The Power*Architect allows diagram objects to move around freely in the playpen. To do so, first select the items you want to move in the Playpen. Click and hold on one of the selected item and drag the items to a desired spot on the playpen.

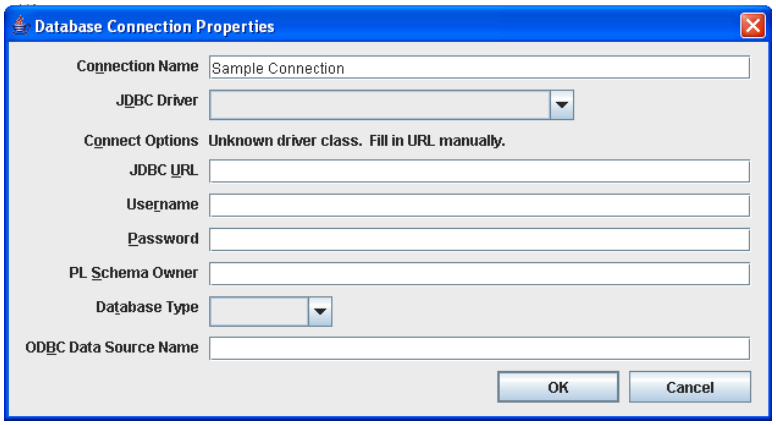
Deleting Diagram Components

Select the desired diagram components on the playpen and click on the delete items button on the Playpen Toolbar. It is also possible to delete the selected items by right clicking on one of the components and selecting "Delete Selected".

Database Usage in Power*Architect

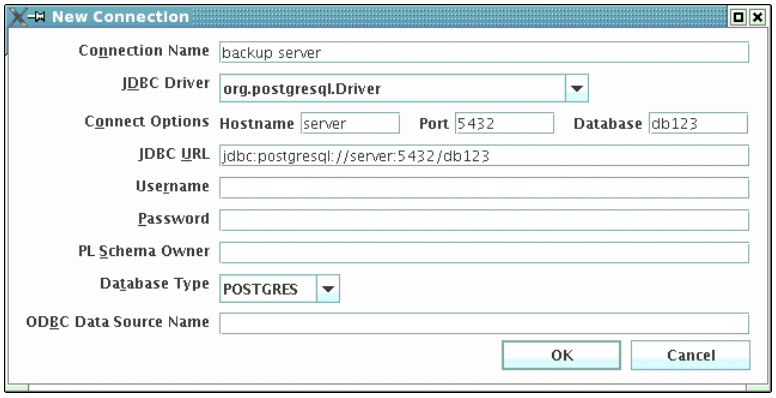
Adding a New Connection to the Power*Architect

To add a new connection, go to "Add Connections" under "Connections" and select "New Connection.." at the top. The other method is to right click a white space on the Database Tree and select "New Connection" under "Add Connections". Both ways open up this dialog:



You must know which JDBC driver to use before you can connect to a database. When you have selected the JDBC Driver, the Connect Options will change to allow you to enter the particular parameters that the given database driver needs. If you are using one of the fully-supported drivers, then as you enter these parameters, they will be added into the "JDBC URL" field in the order that the Java driver expects to see them (this string is sometimes called a "db URL" in Java terminology). In the example below, we've selected the PostgreSQL driver and entered the hostname and database name (the "port number" was already filled in; do not change this unless the database server software has been reconfigured to use a different value).

When you are finished, press the OK button. Any new connection will automatically be added in the user-preference.



Note: if you have too many connections (more than the height of the Architect window), the Architect does not have the ability to show all of them.

Adding an Existing Connection

Right click on empty space in the Database Tree and go to "Add Connection", there it shows all databases that were previously stored on the Architect

Editing Database Connection Properties

Select the database connection you wish to change and go to "Connections" and select the option "Connection Properties..", this leads you to the Connection Setting dialog. An alternate solution is to right click on the database and select "Connection Properties..." option.

Removing a Database Connection

Select the database connection you wish to change and go to "Connections" and select the option "Remove Connection..". Right clicking on the database connection and selecting "Remove Connection.." will perform the same action.

Setting the Current Project Database

 The project database can be set to an existing database connection or the user can create a new connection for it as well. Setting the project database can be achieved in one of several ways. The first method is to right click on any empty space on the Playpen and go to "Set Target Database". Another option is to right-click on the Target Database (the target database can always be identified by this icon:) and choose "Set Target Database" from there.

Navigating through the Database Tree

The Database Tree works like a tree-dropdown model. Clicking on any item will cause the component to expand display the items under that specific component. Each item will have an unique icon beside its name to identify the type of object it is. The table below shows what each icon means:

Table 3.1. Database Tree Icons

Icon	Representation
	Database
	Catalog
	Schema
	Owner
	Table
	Exported Key
	Imported Key

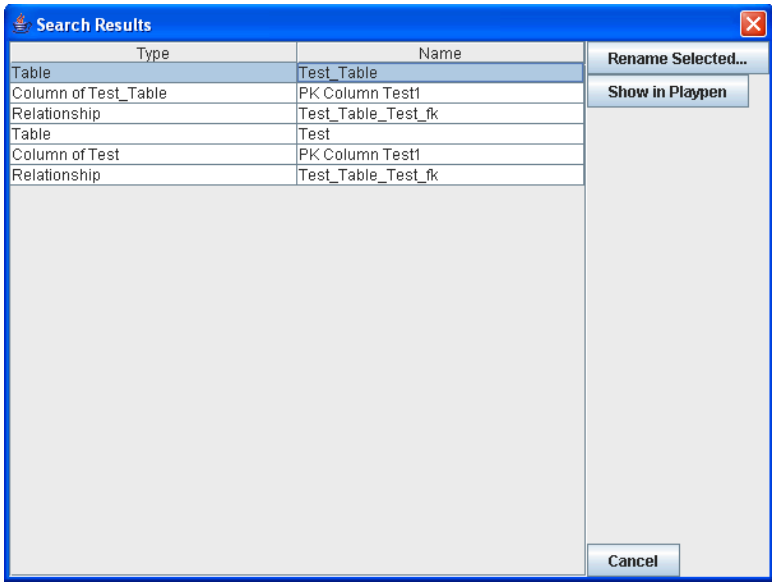
Moving Items from the Database Tree

To copy items from the Database Tree to the playpen, simply select the desired items in the tree and drag them onto the playpen. Depending on the size of the items dragged onto the playpen, it will take some time to load.

Find/Replace Function

What It Does

This functions searches the whole playpen for any relevant matches with the search constraints and displays the resulting matches. The user can request for those items to be focused on the screen.



How to Use Find / Replace Function:

Go to "Edit" under the menu bar and click on Find/Replace option. This shows the Find/Replace function. In this window, enter the search constraints and press "OK" when you are finished. This will popup a new window with your search result in a chart format.

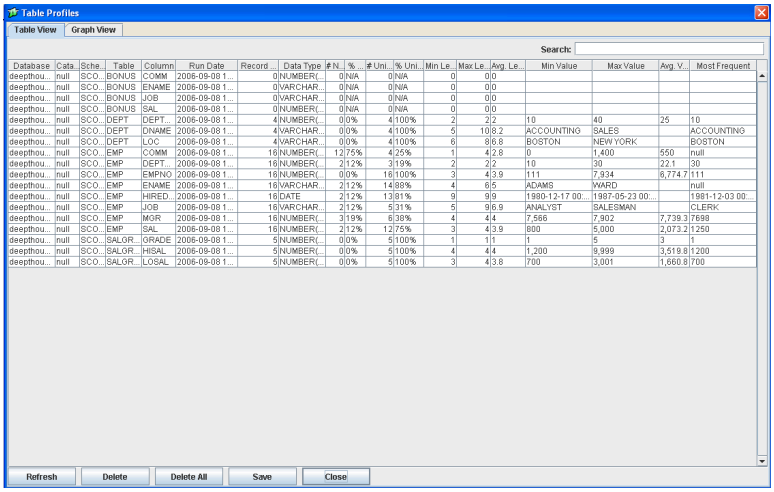
You can rename the component by selecting the column in the list and clicking on "Rename Component". You can also have Power*Architect focus on a certain component by selecting the component in the list and press the "Show in Playpen" button.

Profiling

 Profiling displays a summary of the data found in a database. The summary can be used for such tasks as; database optimization and data migration. Select the columns and tables that you wish to profile from the database tree on the left hand side of the screen. Then activate the profile feature by either going to the "Profile" menu and selecting "Profile...", or by right clicking on a selected item and select "Profile.." from the context menu, or by selecting the Profile Icon in the Advanced Tool icon bar. If there is still an existing Profile window, the new profiling results will be added on to the existing window, otherwise, the resulting profile will be displayed in a new window.

Table View

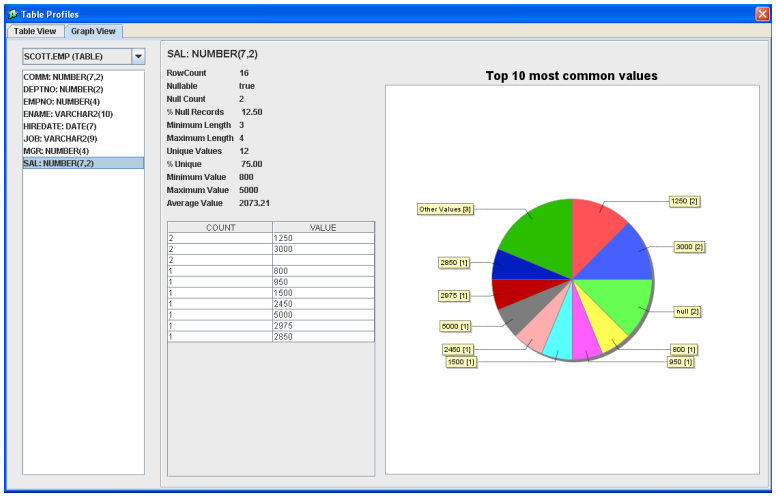
In the profiling window, the profiling information is sortable. Simply click on the column header and it will sort the data by ascending or decesending order. In addition, if you place your mouse pointer over a most frequent cell, it will display the value and frequency of the most recurring items in the column. You can narrow down the results by using the search bar on the top right corner of the profile window. To delete columns from the profile result, simply select the desired columns and press the "Delete" button. As for refreshing the data within the profile table, select any one of the columns within the table and hit "Refresh". This will update the contents of the whole table. If you wish the save the profile results, you can highlight the desired columns to save, and click the "Save" button. If no column is selected, the Architect will save all the displayed results. You have the option to save it in CVS, PDF or HTML format.



Graph View

Besides the tabular view, there is also the graphical view. Simply click on "Graph View" tab on the top to switch. On the left side of this window, you can select which column you want to profile. In the middle,

it will show statistics about that particular column. It will also display the most frequent 'n' values and its frequency within the table. The pie chart on the right displays the portionality of the most frequent 'n' values in the column (the value of 'n' can be set in the "Project Settings" under the "File" menu).



Forward Engineering and Compare Data Model

These two functions are similar; they both involve using the PlayPen (usually) and generating SQL. The Forward Engineer function always creates a SQL script to generate a database identical with the complete current project (current PlayPen contents). The Compare DM function can output either an English-language description or a SQL script describing the differences between two databases, or stored projects, or one of these and the PlayPen.

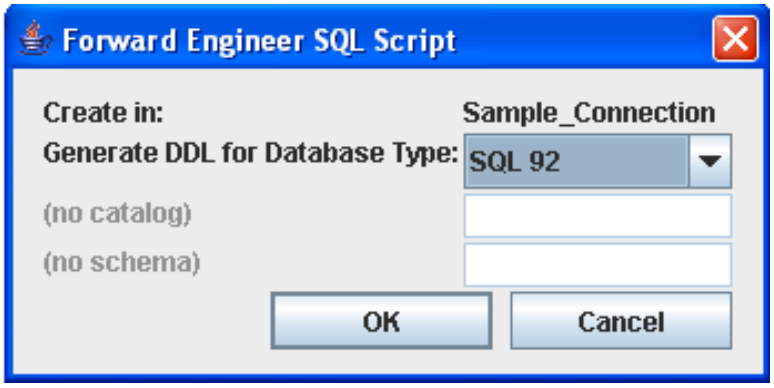
Forward Engineering

What It Does

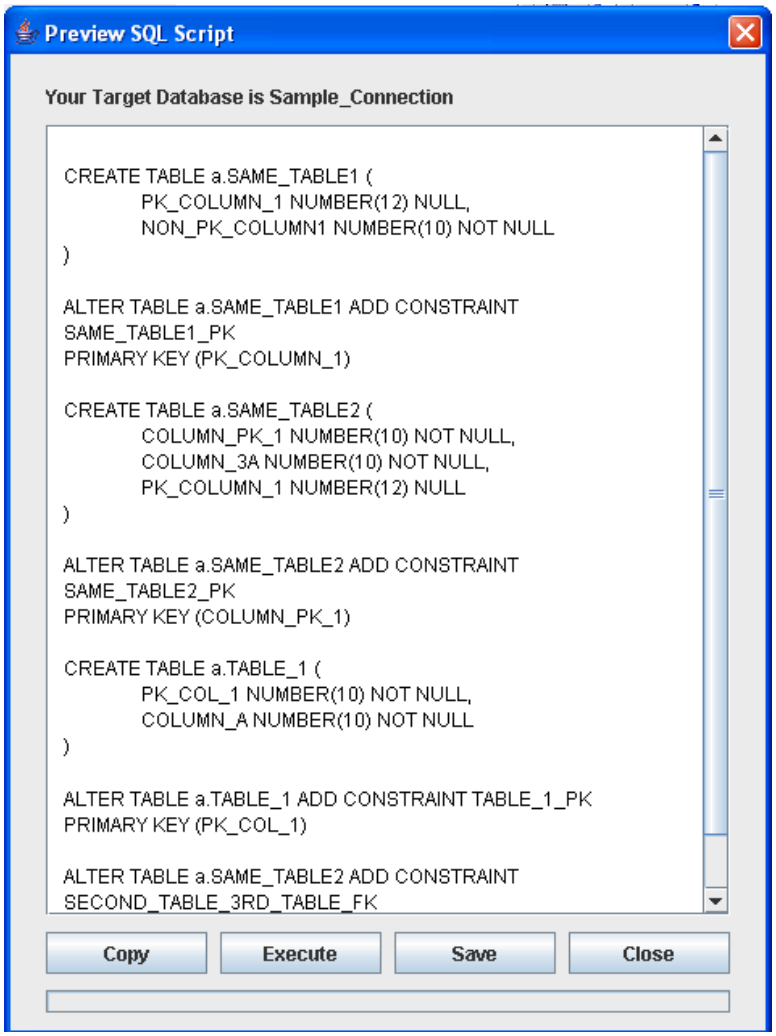
Forward Engineering creates a SQL Script that can be run to update or put the components of the current Playpen into a database

How To Use Forward Engineering

First set the target connection to the database you want the changes to be made in. Then go to "Tools" and click "Forward Engineering". Another way is to press the "Forward Engineering" button at the top. This pops up a dialog that looks similar to the one below:



Fill in the fields as necessary and hit "OK" when you are done. Depending on the situation, a dialog warning you of possible side-effects of creating the script may pop up. Finally a script that would create data structure currently in the Playpen will be displayed. It is the same dialog used in CompareDataModel-Part 2- In SQL Script.



Compare Data Model Function

What It Does

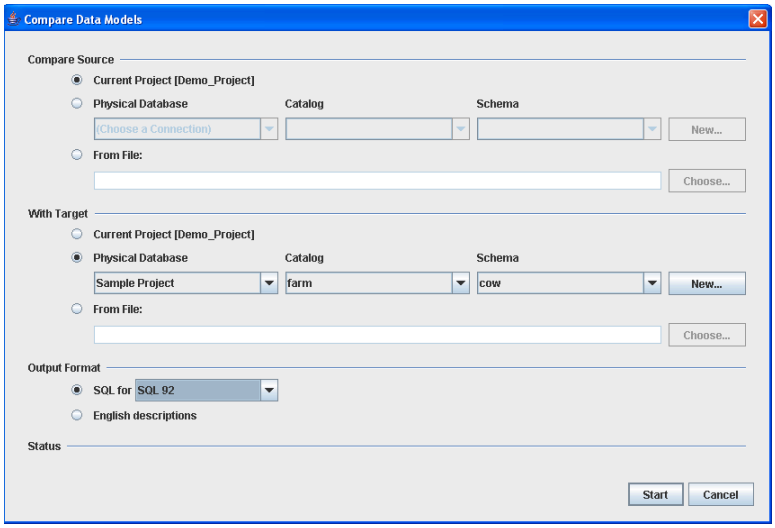
The Compare DataModel Function takes two databases or Power*Architect projects (or one of each) or the current PlayPen, and compares and contrasts the structure of the two database/projects.

How to use Compare Data Model Function:

Part 1


To start the Compare Data Model Function, either go to "Tools" and click "Compare Data Model" or just click the "Compare Data Model" icon on the project toolbar. This will bring up the main Compare Data Model window. Here select the source and target models you want to compare and contrast. You

can choose either the current playpen, an existing database or select a saved project file function. If one of or both the source and the target has invalid models, the start button will not enable. The similarities and differences can be displayed either in SQL Script language or in English.



Part 2

English Descriptions

If this option is selected, a side-by-side document will be displayed stating the similarlities and differences of the source and target in plain English. The left text gives English descriptions to make the source database look like the target database. In addition to the text, they are also colour coded. You can copy the text to the clipboard by pressing the copy button, or save the results to a text file. The table below explains what each colour means.

Table 3.2. Compare Database Model Colour Codes

Colour	Explanation of the Colour Code
Black	This component exists in both databases
Green	This component only exists in this database but not the other
Red	This component does not exist on this database but exists on the other
Blue	This component is a column and is on different keys in the two databases

In SQL Script

If this option is chosen, this will produce a SQL Script in the SQL dialect chosen in Step 1 to make the source database look like the target database. You can either copy the results to the clipboard, or save the results in a text file. If the source has a valid connection database, the Execute button will enable and you can directly execute the changes. If the source does not have a valid connection, the execute button is disabled.

Autolayout

What It Does

It displays the selected tables (or all tables) in an organized manner.

How to Use Autolayout:


Select the tables on the playpen that you want to organize and hit the autolayout button at the top. If one or zero tables are selected, the program will autolayout every table in the playpen. Note that the layout algorithm may produce a few surprises when run with a small number of tables; it works best for a large or medium-sized collection of tables.

SQLRunner

What It Does

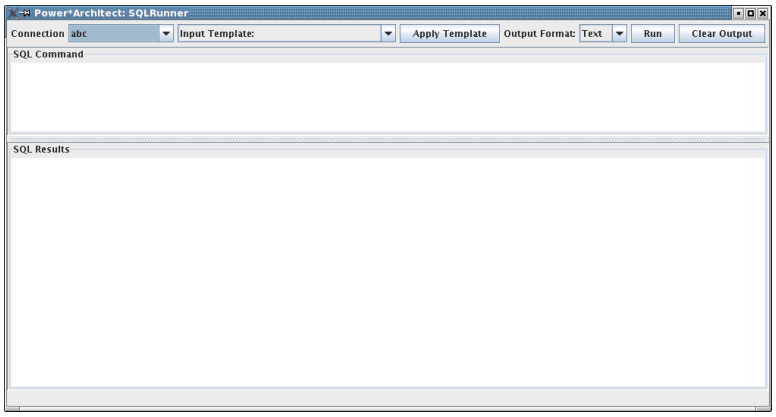
SQLRunner is a "fall-back" tool that lets you work at the raw SQL command level. This is an advanced topic and should only be used by (or made available to) those familiar with the intricacies of SQL commands and the details of your database; like a sharp knife, this tool is very useful in the hands of a skilled chef, but a slip of the fingers here can be quite messy...

SQLRunner was written by Ian Darwin, and is distributed under a liberal free-software, open-source license which permits its inclusion in programs such as Power*Architect.

How to Use SQLRunner

SQLRunner is started from the menu entry under the Tools menu, and begins with the GUI window shown below. The first thing you should do is select which database connection you wish to use. The list of Connections is the same as the main program uses, as set up in the JDBC Connections window.

The basic steps to using SQLRunner are to type a command in the top (SQL Command) window and click the Run button; the results are displayed in the bottom (SQL Results) window. To save you some typing, there is a "Statement Template" mechanism that will insert a template for SELECT, INSERT or UPDATE SQL statements (just select the template you want and click "Apply Template" and the template will replace the current Input Statement.



The command can actually be one of two kinds: either one of a half-dozen escape commands listed below, or, anything that is valid input to your database's command interface (e.g., programs such as psql or SQL*Plus™).

Table 3.3. SQLRunner Escape Characters

Escape Sequence	Action
\dt	Describe list of all tables
\dtT	Describe column names of table named T
\dmX	Set the mode, where X is the first letter of the mode (t for text, s for SQL, h for HTML or x for XML; not needed in the embedded version because the GUI has a control for this)
\oF	Send output to the given file instead of the screen (though you can usually just view the output and copy-and-paste to save parts of it into a file; does not work in GUI versions).
\q	Exit the program (not supported in embedded versions).

SQL Statements are entered one at a time, can be more than one line long, and need not end with a semicolon. These statements are not interpreted by SQLRunner itself, so anything that the given database and driver accepts can be used. For example, with Oracle™, you can use PL*SQL™ statements. With most drivers you should be able to use stored procedures. Each SQL statement is executed in its own transaction context, that is, changes are committed immediately (so be careful!).

Output (Results) Window

Command Output in the chosen format (see below) appears in the SQL Output window. A scrollbar will appear if the information cannot all be seen at once.

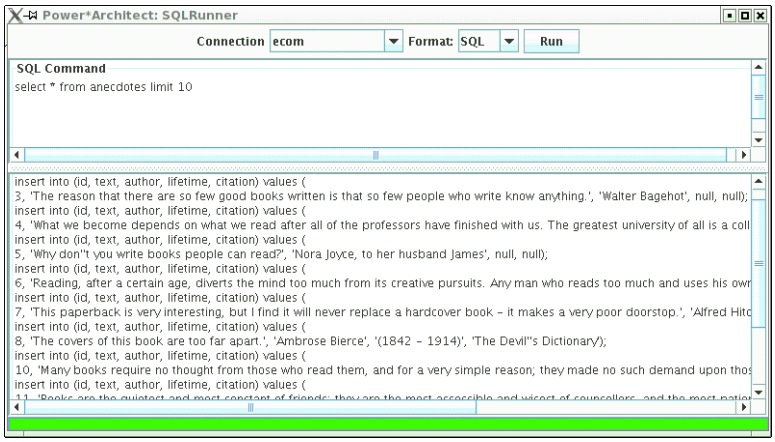
A visual indication of the success or failure of the command is displayed below the output: green for success, red for failure. As well, failures will be accompanied by a pop-up window containing details on the failure.

The Clear Output button clears the contents of the output window.

Output Formats

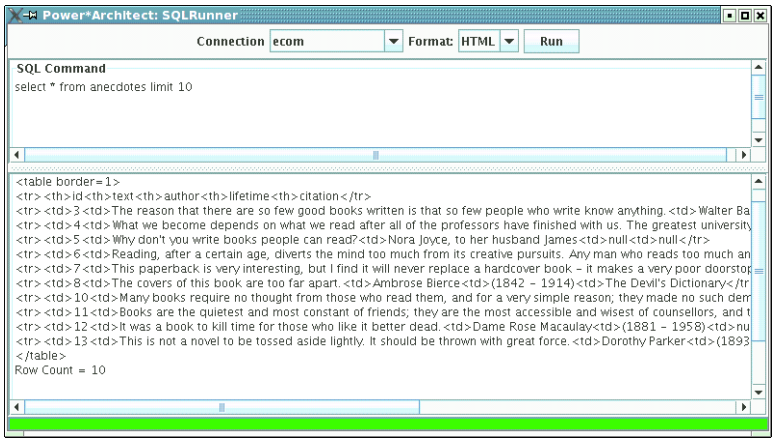
There are several output modes for the display of SQL "select" results: text, SQL, HTML and XML (output from the escape commands are always displayed as plain text). Text mode is the default, and is primarily a raw display format. SQL output is most useful with the output of a SELECT statement; it will generate SQL that will attempt to re-create the data in another database. HTML mode generates an HTML table to display the results of a Select. XML format is similar but may be used for exporting data into other applications.

For example, with SQL mode selected, a "select * from anecdotes" (a table in a sample bookstore web site's database, used to display a casual quotation about books) looked like this:

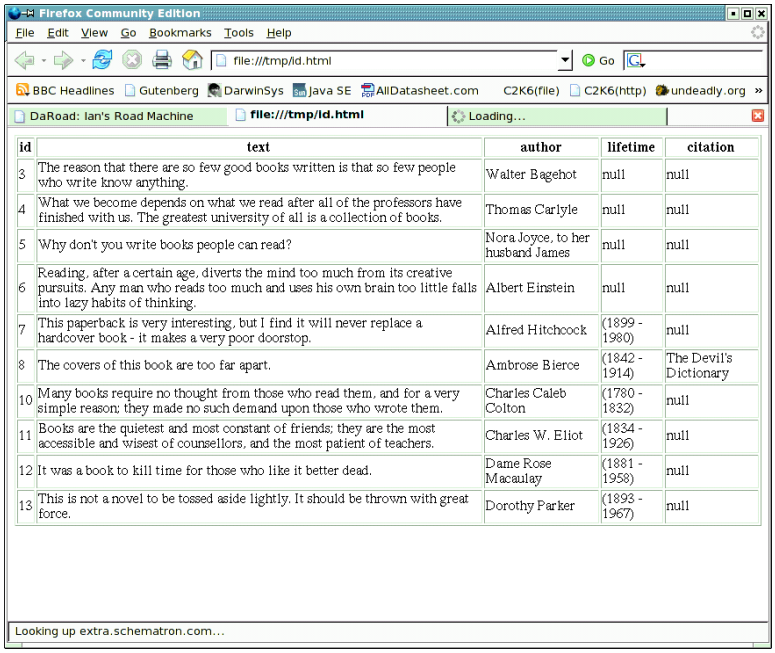


This could, as you can see, be used to create a SQL script to re-create the contents of the database. In fact, some developers use SQLRunner primarily for this purpose: to create stable test databases from "live" data that was created by their application.

You can view this same data in HTML just by changing the Format selection to HTML and clicking the Run button again:



When copied and pasted into an HTML file and viewed in a browser, the output looked like this:



id	text	author	lifetime	citation
3	The reason that there are so few good books written is that so few people who write know anything.	Walter Bagehot	null	null
4	What we become depends on what we read after all of the professors have finished with us. The greatest university of all is a collection of books.	Thomas Carlyle	null	null
5	Why don't you write books people can read?	Nora Joyce, to her husband James	null	null
6	Reading, after a certain age, diverts the mind too much from its creative pursuits. Any man who reads too much and uses his own brain too little falls into lazy habits of thinking.	Albert Einstein	null	null
7	This paperback is very interesting, but I find it will never replace a hardcover book - it makes a very poor doorstop.	Alfred Hitchcock	(1899 - 1980)	null
8	The covers of this book are too far apart.	Ambrose Bierce	(1842 - 1914)	The Devil's Dictionary
10	Many books require no thought from those who read them, and for a very simple reason; they made no such demand upon those who wrote them.	Charles Caleb Colton	(1780 - 1832)	null
11	Books are the quietest and most constant of friends; they are the most accessible and wisest of counsellors, and the most patient of teachers.	Charles W. Eliot	(1834 - 1926)	null
12	It was a book to kill time for those who like it better dead.	Dame Rose Macaulay	(1881 - 1958)	null
13	This is not a novel to be tossed aside lightly. It should be thrown with great force.	Dorothy Parker	(1893 - 1967)	null

With a bit of formatting, or even a CSS style sheet, this HTML page could be made quite usable.

SQLRunner is not perfect, but it is adequate for many purposes involving direct use of SQL.

Quick Start Wizard

What It Does

The QuickStart Wizard is a simple five-step process to set up a database using the Power*Loader.

How to Use The Quick Start Wizard

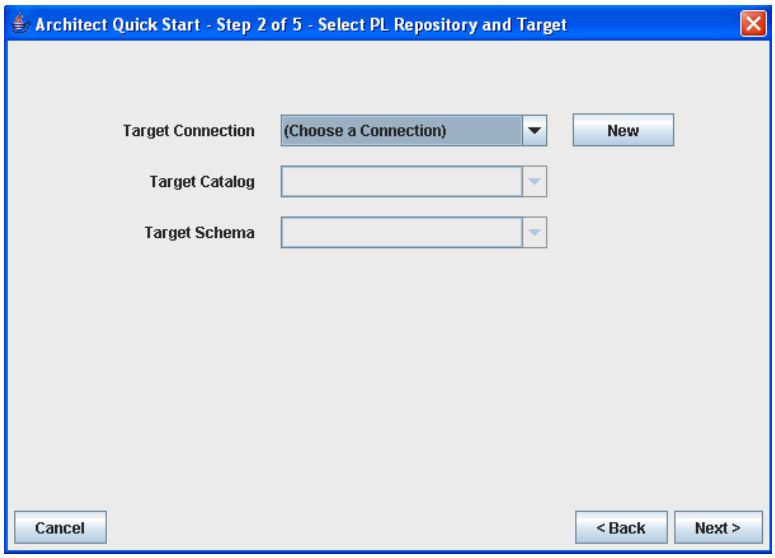
To run the QuickStart Wizard, go to ETL and click QuickStart Wizard button under Power*Loader.

Step 1

Select the source playpen components you want and click next.

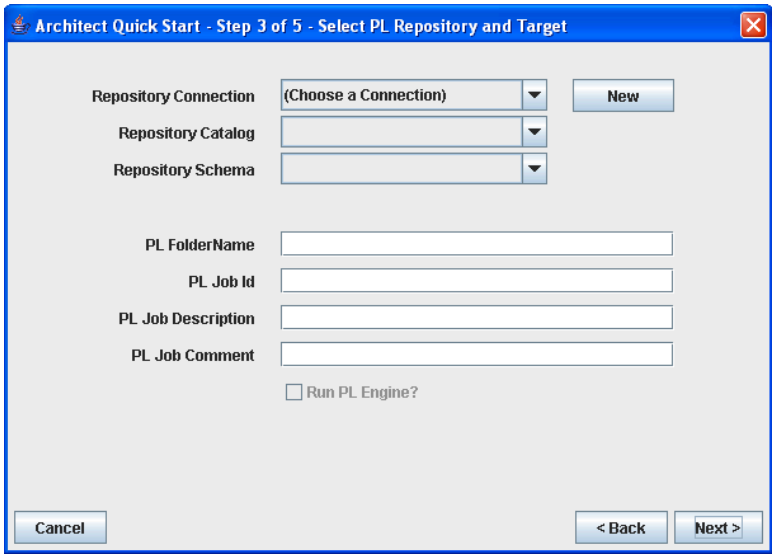
Step 2

Select or create the target database.



Step 3

Select or create the repository database and fill in any additional information. The JobID must be filled in order to continue.



Step 4

A SQL Script will be created; you can review the script and if you wish to execute it, press "Execute"

Step 5

A summary report of the process will be shown.

Chapter 4. Database Product Notes

Database	Notes
Oracle	Is fully supported.
SQL Server	Is fully supported.
PostgreSQL	Is fully supported.
IBM DB2	Is fully supported.
HSQLDB	Works; used in samples.
Derby	Does not work; the current version (10.1.2) has what we consider some unwarranted chumminess with the JDBC Driver Manager that breaks because we use our own Java "ClassLoader"
MySQL	Not tested yet.

Chapter 5. Troubleshooting

We have worked hard to ensure that Power*Architect works correctly. However there are probably always going to be some combinations of different database products and database configurarations, user actions, computer setups, and so on, that just don't work. We apologize in advance for any inconvenience this may cause...

If you are having trouble with Power*Architect, we may ask that, in order to help us to diagnose the problem, you take some or all of the following actions:

- Prepare a description of what you were doing
- Prepare a copy of any errors you encountered
- Post your problem to the Power*Architect help forum [http://www.sqlpower.ca/forum/forums/show/2.page]

Chapter 6. Glossary

This section lists some database-related terms and their meanings.

Column	The set of all instances of a given field from all records in a table [http://foldoc.org/foldoc/foldoc.cgi?table] .
Database	One or more large structured sets of persistent data, usually associated with software to update and query [http://foldoc.org/foldoc/foldoc.cgi?query] the data. A simple database might be a single file containing many records [http://foldoc.org/foldoc/foldoc.cgi?records] , each of which contains the same set of fields [http://foldoc.org/foldoc/foldoc.cgi?fields] where each field is a certain fixed width.
Data Modelling	The product of the database design process which aims to identify and organize the required data logically and physically.
Data Warehousing	A database, often remote, containing recent snapshots of corporate data. Planners and researchers can use this database freely without worrying about slowing down day-to-day operations of the production database.
ETL	Extraction, Transforming and Loading – the process of maintaining and transforming data into and out of a relational database.
Foreign key	A column [http://foldoc.org/foldoc/foldoc.cgi?column] in a database table [http://foldoc.org/foldoc/foldoc.cgi?table] containing values that are also found in some primary key [http://foldoc.org/foldoc/foldoc.cgi?primary+key] column (of a different table). By extension, any reference to entities of a different type. Some RDBMSs [http://foldoc.org/foldoc/foldoc.cgi?RDBMSs] allow a column to be explicitly labelled as a foreign key and only allow values to be inserted if they already exist in the relevant primary key column.
Identifying Relationship	Where the key of the parent table is a subset of the key of the child table.
JDBC	Java DataBase Connectivity, an unofficial acronym for the "java.sql" package of functionality used to access relational databases from programs written in the Java programming language.
Key	A value used to identify a record [http://foldoc.org/foldoc/foldoc.cgi?record] in a database, derived by

	applying some fixed function to the record. The key is often simply one of the fields [http://foldoc.org/foldoc/foldoc.cgi?fields] (a column [http://foldoc.org/foldoc/foldoc.cgi?column] if the database is considered as a table with records being rows, see " key field [http://foldoc.org/foldoc/foldoc.cgi?key+field] "). Alternatively the key may be obtained by applying some function, e.g. a hash function [http://foldoc.org/foldoc/foldoc.cgi?hash+function] , to one or more of the fields. The set of keys for all records forms an index [http://foldoc.org/foldoc/foldoc.cgi?index] . Multiple indexes may be built for one database depending on how it is to be searched.
Primary key	The candidate key [http://foldoc.org/foldoc/foldoc.cgi?candidate+key] selected as being most important for identifying a body of information (an entity, object or record [http://foldoc.org/foldoc/foldoc.cgi?record]).
Record (row)	One or more structured sets of persistent data, usually associated with software to update and query [http://foldoc.org/foldoc/foldoc.cgi?query] the data. A simple database might be a single file containing many records [http://foldoc.org/foldoc/foldoc.cgi?records] , each of which contains the same set of fields [http://foldoc.org/foldoc/foldoc.cgi?fields] where each field is a certain fixed width.
SQL	Originally SEQUEL [http://en.wikipedia.org/wiki/SQL#History] and still pronounced that way by many practitioners, SQL is the Standard Query Language; a unified language for creating queries that is accepted (with some variations) by all modern relational databases.
Table	A collection of records [http://foldoc.org/foldoc/foldoc.cgi?records] in a relational database [http://foldoc.org/foldoc/foldoc.cgi?relational+database] .

Some of these terms are from FolDoc, "The Free On-line Dictionary of Computing", http://www.foldoc.org/, Editor Denis Howe.

Chapter 7. Acknowledgements

The Apache Software Foundation

The Power*Architect development team is grateful to the Apache Software Foundation and their contributors; their high-quality reusable Java libraries have been invaluable in the development of the Architect. The text of the Apache License follows, because we are redistributing several Apache libraries upon which the Architect depends.

The following license applies to these library jar files, which are distributed as part of the Architect download:

- commons-beanutils.jar
- commons-digester.jar
- commons-logging.jar
- commons-beanutils-bean-collections.jar
- commons-beanutils-core.jar
- jakarta-regexp-1.2.jar
- commons-collections-3.1.jar
- commons-dbcp-1.2.1.jar
- commons-pool-1.3.jar

Apache License Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or

management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or

Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file,

excluding those notices that do not pertain to any part
of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form
or documentation, if provided along with the Derivative
Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally
appear. The contents of the NOTICE file are for
informational purposes only and do not modify the
License. You may add Your own attribution notices within
Derivative Works that You distribute, alongside or as an
addendum to the NOTICE text from the Work, provided that
such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your
modifications and may provide additional or different
license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such
Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise
complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly
state otherwise, any Contribution intentionally
submitted for inclusion in the Work by You to the
Licensor shall be under the terms and conditions of this
License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall
supersede or modify the terms of any separate license
agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks. This License does not grant permission to
use the trade names, trademarks, service marks, or
product names of the Licensor, except as required for
reasonable and customary use in describing the origin of
the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable
law or agreed to in writing, Licensor provides the Work
(and each Contributor provides its Contributions) on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied, including, without
limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for
determining the appropriateness of using or
redistributing the Work and assume any risks associated
with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no
legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable

law (such as deliberate and grossly negligent acts) or
agreed to in writing, shall any Contributor be liable to
You for damages, including any direct, indirect,
special, incidental, or consequential damages of any
character arising as a result of this License or out of
the use or inability to use the Work (including but not
limited to damages for loss of goodwill, work stoppage,
computer failure or malfunction, or any and all other
commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While
redistributing the Work or Derivative Works thereof, You
may choose to offer, and charge a fee for, acceptance of
support, warranty, indemnity, or other liability
obligations and/or rights consistent with this License.
However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not
on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor
harmless for any liability incurred by, or claims
asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the
following boilerplate notice, with the fields enclosed
by brackets "[]" replaced with your own identifying
information. (Don't include the brackets!) The text
should be enclosed in the appropriate comment syntax for
the file format. We also recommend that a file or class
name and description of purpose be included on the same
"printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the
"License"); you may not use this file except in
compliance with the License. You may obtain a copy of
the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in
writing, software distributed under the License is
distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See
the License for the specific language governing
permissions and limitations under the License.

JGoodies Karsten Lentzsch

The Power*Architect team is also grateful to JGoodies for their excellent forms layout manager for Swing. JGoodies forms is released under the BSD license, reproduced below.

The following license applies to these library jar files, which are distributed as part of the Architect download:

- forms-1.0.6.jar

The BSD License for the JGoodies Forms
==========

Copyright (c) 2002-2006 JGoodies Karsten Lentzsch. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

o Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

o Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JUnit

The Power*Architect team would also like to extend our sincere thanks to the JUnit.org team. JUnit forms an invaluable part of our development process, but it is not redistributed as part of the Architect download so its license is not reproduced here.

If you develop software, you should become test infected too! Learn about JUnit at http://www.junit.org/ [http://www.junit.org/] .

The Eclipse Foundation

The Power*Archited was primarily developed and tested using the Eclipse [http://www.sqlpower.ca/] Java Development Tools, one of the more productive Java environments around.

Sun Microsystems

Last but not least, many thanks to Sun Microsystems [http://java.sun.com/] and their various Java development teams for creating, extending, bugfixing, documenting, and supporting the Java platform over the past *N* years!