SQuirreL, a Universal SQL Client

by Gerd Wagner and Glenn Griffin

Do you use a Relational Database System (RDBMS)? If so, you have probably run across one or more of the following situations:

- Typing a long SQL statement to change one value in the DB.
- Re-typing the same SQL statement over and over, possibly with slight variations.
- Working with multiple databases on separate machines.
- Using databases from different vendors, such as Oracle, MySQL or PostgreSQL.
- Teachers, students, or other folks who need to work with databases but are not SQL experts.

For anyone who needs to work with an RDBMS, SQuirreL can make life easier.

What is SQuirreL?

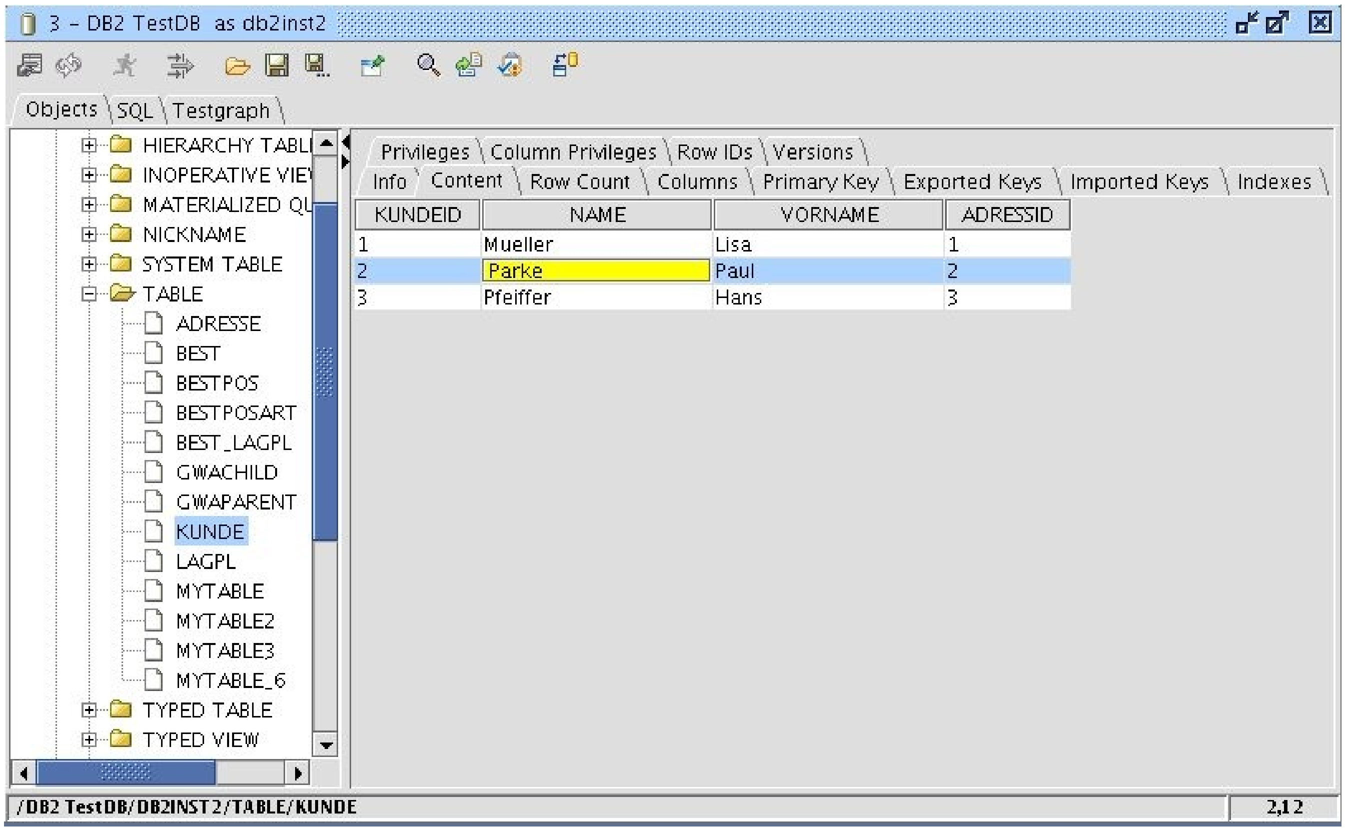
The SQuirreL SQL client provides a simple graphical interface to relational databases. Because it is built using Java, it can access any JDBC-compliant database running on any machine, allowing remote access to multiple databases. A SQuirreL user can:

- easily view and edit data in any JDBC-compliant database,
- view the database's meta-data,
- work with multiple databases on both local and remote machines,
- use a single, consistent interface to work with different database engines, and
- expand the tool's capabilities and include DB-specific functionality using plugins.

The user can click on tables to view them and edit data, or use full SQL operations. Data can be viewed in read-only mode for safety, or in an editable mode where it may be modified by simply typing the new data into the table. All of the meta-data for the database (eg: data types, table column names, etc.) are accessible through SQuirreL. In cases where multiple types of database engines are being used (eg: Oracle, MySQL, PostgreSQL, etc.), the user does not need to learn multiple DB-management tools since SQuirreL-SQL provides a common mechanism for accessing them all. In those cases where a Database engine has non-standard quirks, SQuirreL's plugin architecture allows users to include DB-specific components to handle those operations. The plugin architecture also allows developers to create add-on functions that users may choose to include or not as they wish.

As examples of operation, figure 1 shows SQuirreL providing simple access to a single table, and figure 2 shows the full-function SQL window.

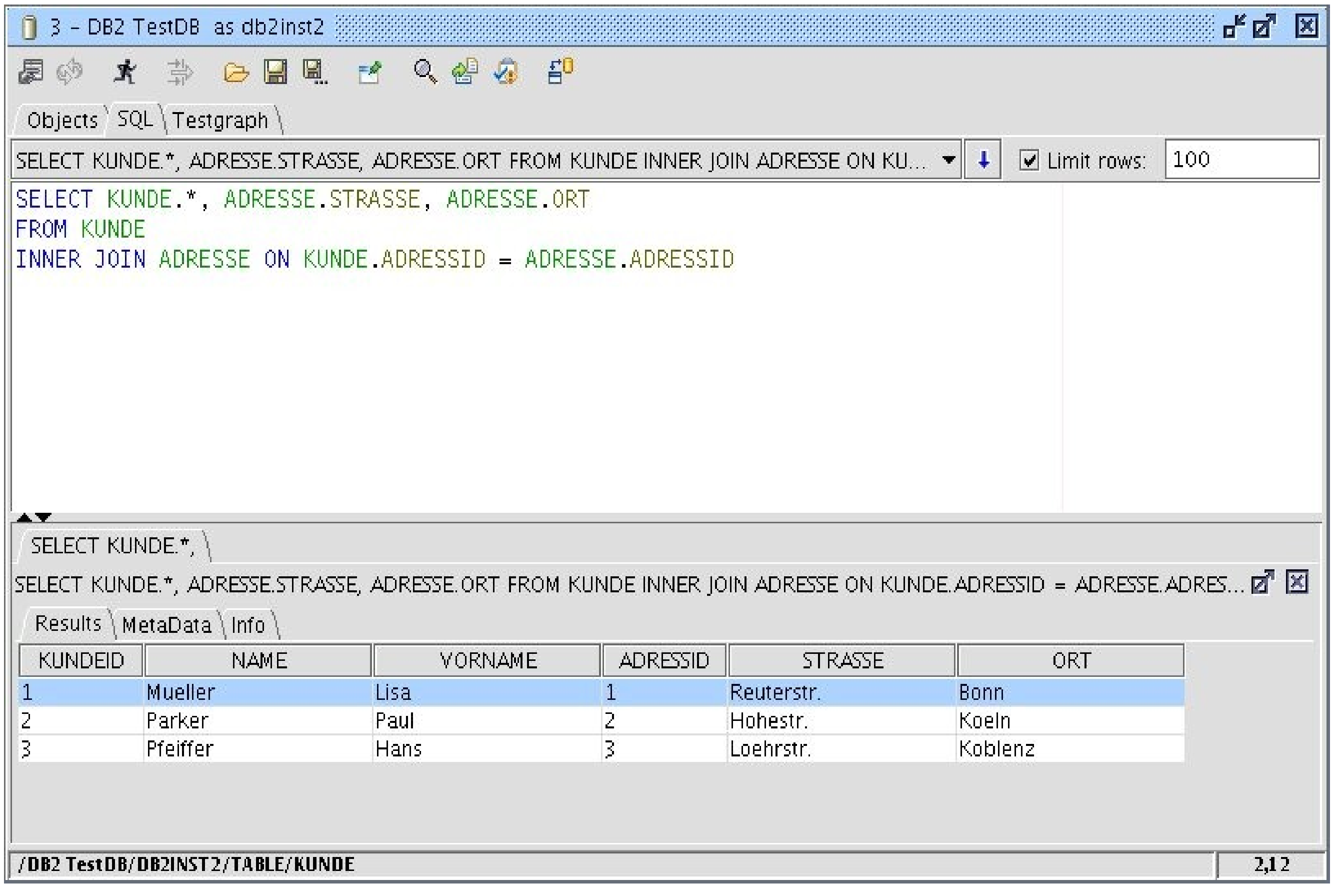
Figure 1: Editing data of a single table.



KUNDEID	NAME	VORNAME	ADRESSID
1	Mueller	Lisa	1
2	Parke	Paul	2
3	Pfeiffer	Hans	3

/DB2 TestDB/DB2INST2/TABLE/KUNDE 2,12

Figure 2: Executing SQL and DDL statements.



```
SELECT KUNDE.*, ADRESSE.STRASSE, ADRESSE.ORT
FROM KUNDE
INNER JOIN ADRESSE ON KUNDE.ADRESSID = ADRESSE.ADRESSID
```

KUNDEID	NAME	VORNAME	ADRESSID	STRASSE	ORT
1	Mueller	Lisa	1	Reuterstr.	Bonn
2	Parker	Paul	2	Hohestr.	Koeln
3	Pfeiffer	Hans	3	Loehrstr.	Koblenz

/DB2 TestDB/DB2INST2/TABLE/KUNDE 2,12

Background
Through the JDBC standard and today's broad availability of JDBC drivers the Java platform is able to access almost all relational databases. JDBC provides a formerly unknown level of uniformity and simplicity. Thus the JDBC API together with the Java platform offers software developers uniform and easy to use access to almost all relational databases. The Open Source client SQuirreL aims to make these advantages also available for database users.

In the next section we will show how to set up and use SQuirreL to get easy, uniform access to databases. After that we will talk about the groups of people that can benefit from using SQuirreL and the features that will help them. We will end with a code examples and the steps needed to create a new plugin.

Download and installation
SQuirreL may be downloaded and used for free (under the LGPL license) from www.squirrelsql.org. On that site you will find the following:
- the SQuirreL installer (squirrel-sql-<version>-install.jar),
- the SQuirreL MacOS X (squirrel-sql-<version>-MacOSX-install.jar

Plugins were once distributed separately as zipped archives. However, this lead to version problems for users and developers, so all plugins available from www.squirrelsql.org are included in the installer now. The installer allows the user to choose just the base software, a “standard” set of plugins that we believe will be useful to most people, and optional plugins that enhance SQuirreL in a variety of ways.

Before installing SQuirrel, you will need to have the Java Runtime Environment JRE version 1.5.x or higher available in your environment.

The installation jar file uses the IzPack installer and is directly executable. After the usual screens for accepting the license and selecting the directory to install into, the installation program asks whether you want the “basic” or the “standard” installation. The basic installation contains all of the functions you will need to view and edit the data and metadata in your databases. The “standard” installation also includes a set of plugins that we have found useful and that are not DB-vendor-specific. These plugins are:

- Code Completion – The same code-completion function as found in IDEs.
- Syntax – Syntax highlightning and abbreviations.
- Edit Extras – Auxiliary functions to work with SQL code for example formatting
- Graph – Creates a chart of the tables and foreign-key relationships between them
- SQL Script – Generates SQL and DDL scripts
- SQL Bookmarks – Manages SQL code templates
- Look and Feel – Allows changes to the look and feel

If you choose the basic installation, you can always add one or more of these plugins later, by re-launching the installer.

In addition to the installation directory, SQuirreL uses two directories that you may need to know about:

1. Within the installation directory is a sub-directory named “plugins”.   This is where all of the code for the plugins is located.

2. The other directory is created when SQuirreL runs for the first time. It contains alias and driver definitions as well as various history and customization files. This directory is located in C:\Documents and Settings\<username>\.squirrel-sql on Windows, $HOME/.squirrel-sql on Linux and /Users/<username> on Mac OS X.

We strive to release several times a year. In addition to the regular releases, we create a “snapshot” release each week which consists of an IzPack installer and source archive which represents code modifications made during that week. The purpose of these snapshots are to provide a way for end-users to provide immediate feedback to bug-fixes and enhancements made outside of the normal release cycle.

You are now ready to run SQuirreL.

Connecting to Databases

Getting your client connected to your database can be tricky. In SQuirreL this is a two-step process:

- Get the JDBC .jar containing the appropriate driver and tell SQuirreL where that

driver is, and
- Define a link to a specific database on a specific machine using that driver.

These are referred to as defining the "Driver" and creating an "Alias", where the "Alias" can be thought of as a specific instance of the more general "Driver" configuration.

Every time SQuirreL is started it opens the Driver and the Alias windows on the desktop,as shown in figure 3. In the Drivers window you will see a blue check mark next to each of the drivers that is currently available in SQuirreL's environment. If the driver for the database that you need to use has a red 'X' next to it, you will need to configure that driver before continuing. To do that, you will need to get the driver from your database vendor and load it onto your local machine, then tell SQuirreL (through the Drivers window) where to find it. You can also add new Drivers for database engines that SQuirreL does not already know about.

The next step is to create an Alias (figure 4), which describes a connection to a specific database on a specific machine. While creating the Alias, you will need to enter the URL for the database. This is usually the hardest part of using JDBC since each DB vendor uses a different format, and some options (such as port numbers) may be unique to your installation. SQuirreL tries to help by including a "hint" in the URL field that identifies the syntax expected by the driver. You may need more information from your DB Administrator (e.g machine name, port number, user name, password etc.) in order to properly create the URL and establish the connection.

Now that you have created an Alias for your database, just double-click its name in the Alias window and SQuirreL will open a connection to it.

Figure 3 SQuirreL desktop

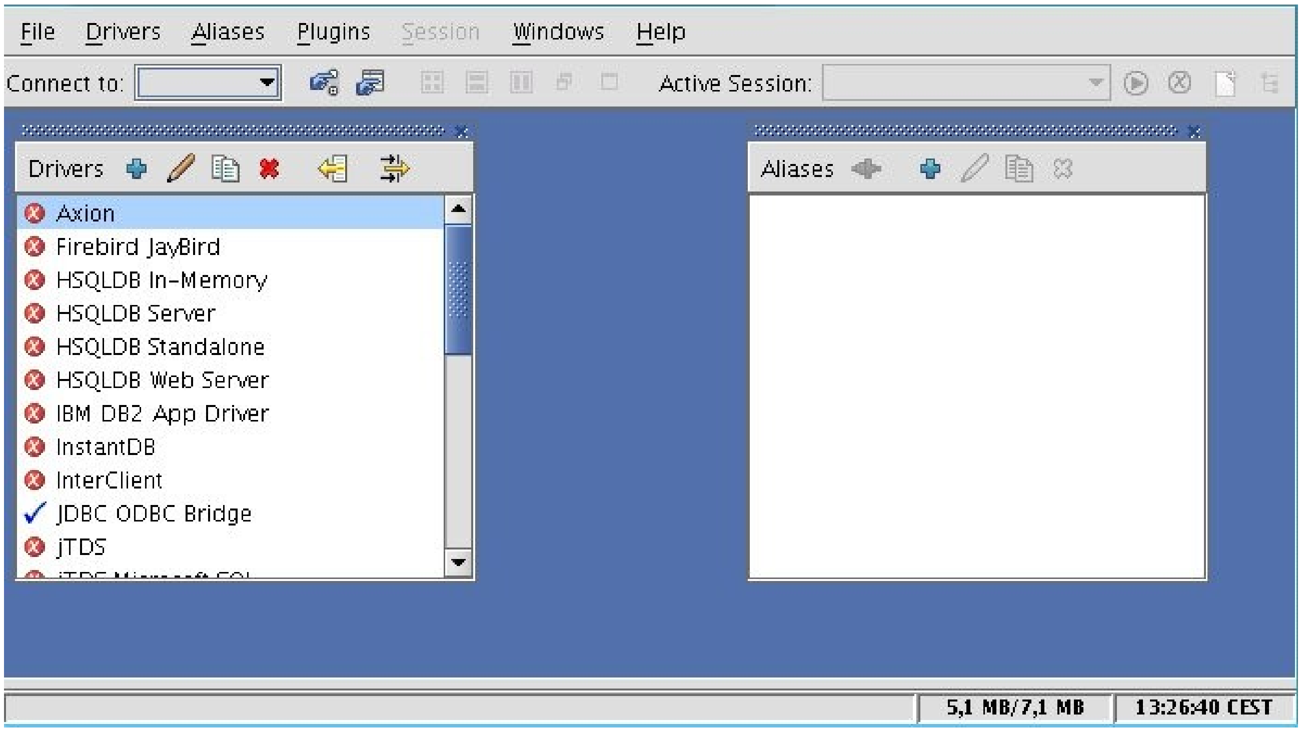
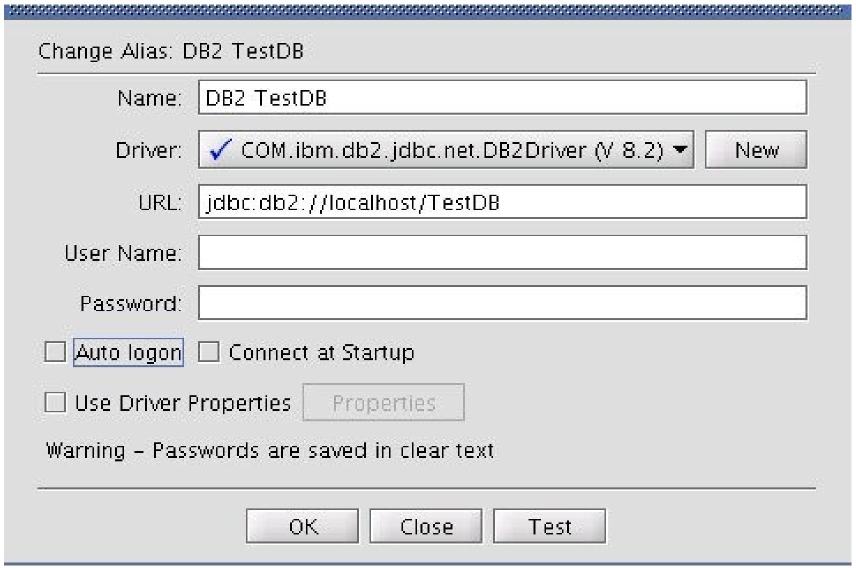


Figure 4 Alias window


Change Alias: DB2 TestDB

Name: DB2 TestDB

Driver: ✓ COM.ibm.db2.jdbc.net.DB2Driver (V 8.2)

URL: jdbc:db2://localhost/TestDB

User Name:

Password:

☐ Auto logon ☐ Connect at Startup

☐ Use Driver Properties

Warning - Passwords are saved in clear text

Working with the database – a Session

When you open a connection to a database you will get a Session window. A Session corresponds to a connection to a single database. You may have multiple sessions with several databases open at the same time, and each one will have its own window.

SQuirreL follows the philosophy that simple things should be simple to do, and complicated things should be as easy as possible. To do this, the Session window has two ways of working with the database, each of which corresponds to a tabbed panel in the window.

The Objects tab provides a simple tabular view of the database. All of the database meta data (types of data and names for those types, current size of DB, etc.) is displayed in tabular form by clicking on the database in the tree view in the left pane, then selecting a sub-tab in the right pane. Clicking on a table name in the tree view (figure 4) gives access to the contents of the table as well as the table's meta-data, such as column descriptions, row counts, etc. The table can be displayed in a text form, a read-only table, or an editable table. When the output is an editable table, changing the value in the table on the screen will change the data in the database. (Simple!) Data can also be imported from and exported to files, and all of the standard data types including BLOBs and CLOBs are supported. New rows of data may be inserted and rows can be easily deleted in the table. DB updates can be made instantaneously, or they can be done within the context of a user-controlled transaction.

The SQL tab (figure 2) supports general SQL operations. While the Objects tab is simple to use, it cannot handle complex operations. Examples of these would include multiple tables in a single operation, such as a join, structural changes like "alter column" or "add table", or vendor-specific operations such as viewing stored procedures. The SQL tab allows you to enter any SQL text, and that text is passed to the DB engine for processing. The results are returned as tables, which can be presented as text, read-only, or, for SELECTs on a single DB table, as an editable table. The results are returned in a tabbed pane at the bottom of the SQL tab panel and include the meta-data associated with that response. The SQL tab also has a history combo box, which lets you select previous statements to be repeated, or edited before re-entering.

Especially Plugins add many functions to the SQL Editor of the SQL Tab. To make it easy for users to overview and call the functions the user can open the so called tools popup by the Ctrl+T shortcut, see figure 5. The tools popup shows all editor related functions with a selection name, a short description and if present the functions own shortcut. The popup's contents may be filtered by typing the beginning of a selection name. This way all editor functions are keyboard accessible and the only shortcut the user needs to know is Ctrl+T.

Like most good general purpose programs, SQuirreL lets the user customize their environment. When there are two ways to do something, SQuirreL implements both and provides a parameter to let the user control which one is used. These parameters come in three flavors:

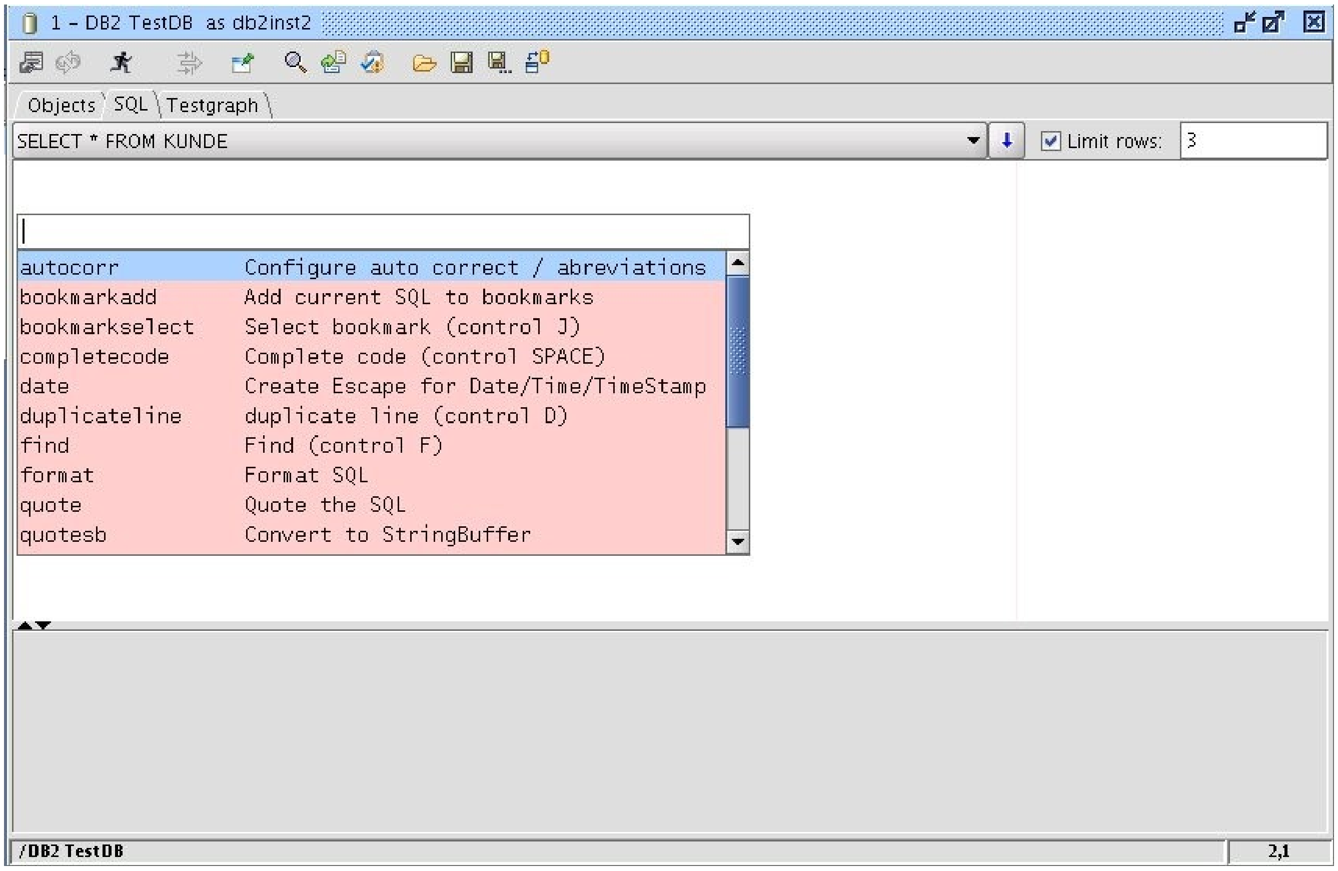
- Global Preferences are settings that are generally set once and apply to all sessions. These options include which toolbar/status bars to show, whether or not to show tool tips, JDBC timeout and debug settings, proxy configuration, and controls on how to display certain data types such as BLOB/CLOBs and date/time fields.
- Session Properties relate more to individual sessions. You can set the defaults used on all new sessions, and then customize them for a particular session. The session properties include where to put the tabs for the panels, which kind of output form to use, limits on what kind and how much data to retrieve and display, and controls used

on the SQL tab such as the statement separator character.

- Plugin-specific properties allow the user to customize how the plugin works to enhance some aspect of SQuirreL. Since some plugins introduce behavior which is appropriate for only one particular database (Oracle, DB2, SQL-Server, Sybase, Derby, H2, HSQL, etc.) these properties can apply to all sessions for that database. For example, the statement separator is usually the same for all Oracle sessions (“;”) that are used to load Oracle scripts. The same holds true for all Sybase sessions (“GO”). Since SQuirreL can connect to multiple database types simultaneously, plugins give the ability to keep these statement separators specific to the particular database for each session.

There are several dozen parameters that you can adjust. The default settings should be adequate to get you started, but you will probably want to look through the Global Preferences and Session Properties windows and adjust them to your taste.

Figure 5 Tools Popup



SELECT * FROM KUNDE

autocorr	Configure auto correct / abreviations
bookmarkadd	Add current SQL to bookmarks
bookmarkselect	Select bookmark (control J)
completecode	Complete code (control SPACE)
date	Create Escape for Date/Time/TimeStamp
duplicateline	duplicate line (control D)
find	Find (control F)
format	Format SQL
quote	Quote the SQL
quotesb	Convert to StringBuffer

Plugins

The tables 1 to 3 presents all plugins available on www.squirrelsql.org together with a short description. This overview is followed by a more detailed description of the five most commonly used ones.

Table 1 standard plugins (part of SQuirreL's standard installation)

Name	***Description***
Look and Feel	Allows the selection between several look-and-feels.
SQL Bookmarks	Definition and reuse of SQL Templates.
SQL Scripts	Saving and loading scripts from files as well as serveral functions to generate scripts.
Graph Plugin	Visualization of tables and their relations.
Edit Extras	Several auxiliary editor functions.
Code Completion	Completion of SQL and DDL code.
Syntax	Syntax highlighting, abreviations and integration of the Netbeans editor.

Table 2 released plugins (not in the standard installation)

Name	***Description***
MySQL Plugin	Specific functions for the MySQL database.
Oracle Plugin	Specific functions for the Oracle database.
SQL Validator	Checking SQLs against SQL standards. The plugin uses the Web Service of Mimer SQL.

Table 3 Beta plugins

Name	***Description***
Firebird Plugin	Specific functions for the Firebird database.
Microsoft MSSQL Plugin	Specific functions for the Microsoft SQL Server-database .
Session Scripts	Allows to define scripts that are executed during Session starts.

Code completion plugin

Code completion is one of the most widely used features in modern IDEs. The Code completion plugin (see figure 7) offers completion for almost all constructs in SQL and

DDL:
- Key words, including SQL standard keywords as well as those key words delivered by the JDBC driver.
- Tables
- Columns
- Views
- Stored procedures. completion generates the complete JDBC call syntax including templates for parameters.
- Catalogs
- Schemas

Beyond this, the plugin offers so called completion functions that can be used to generate SQL Joins. To explain completion functions we will describe an example using the tables BEST, BEST_LAGPL and LAGPL as shown in figure 6. To generate the Join from BEST to LAGPL you'd write the following expression:

```
#i,BEST,BEST_LAGPL,LAGPL,
```

If the cursor is positioned at the end of this expression and Code completion is called by the Ctrl + Space shortcut the plugin generates the following code for you:

```
INNER JOIN BEST_LAGPL ON BEST.BESTID = BEST_LAGPL.BESTID
INNER JOIN LAGPL ON LAGPL.LAGPLID = BEST_LAGPL.LAGPLID
```

Graph plugin

With the Graph plugin you can visualize groups of tables together with their foreign key relations. All the plugin initially does is to add the menu item 'Add to graph' to the tables context menu in the Object tree. When the user chooses this item the Session main window receives a new tab. This tab shows the graph of the selected tables, see figure 6.

The Session main window can have an arbitrary number of Graph tabs. The user can name tabs and save them. When the Session is opened next time the Graph tabs will be opened as well. This way the user can make the most important tables and relations available with a single mouse click.

All foreign key columns of a table are marked by an '(FK)' at the end of their names (figure 6). If you double click on a foreign key column, the table the foreign key points to is added to the Graph. By using the table’s context menu, all parents, all children or both can be added to the Graph, which gives an easy way to browse through the table structures.

SQuirreL's Plugin API enables plugins to communicate with each other. Through this the Graph plugin makes use of the SQL Scripts plugin. This way all visual elements in a Graph can be translated to DDL by mouse click.

The Graphs created by the plugin can be easily scaled and arranged as desired for printing on either a single page or spread across multiple pages.

Syntax plugin
The Syntax plugin shows the power of SQuirreL's plugin API because it replaces a central part of SQuirreL, the SQL editor, with a new component, the Netbeans editor (http://editor.netbeans.org). The Syntax plugin is responsible for SQL code coloring. In the Session properties the user can choose the colors for SQL keywords, table names, column names, etc. The plugin also allows the user to define auto corrections and abbreviations. Auto corrections and abbreviations work the same way as in common office products. They can be configured or completely switched off.

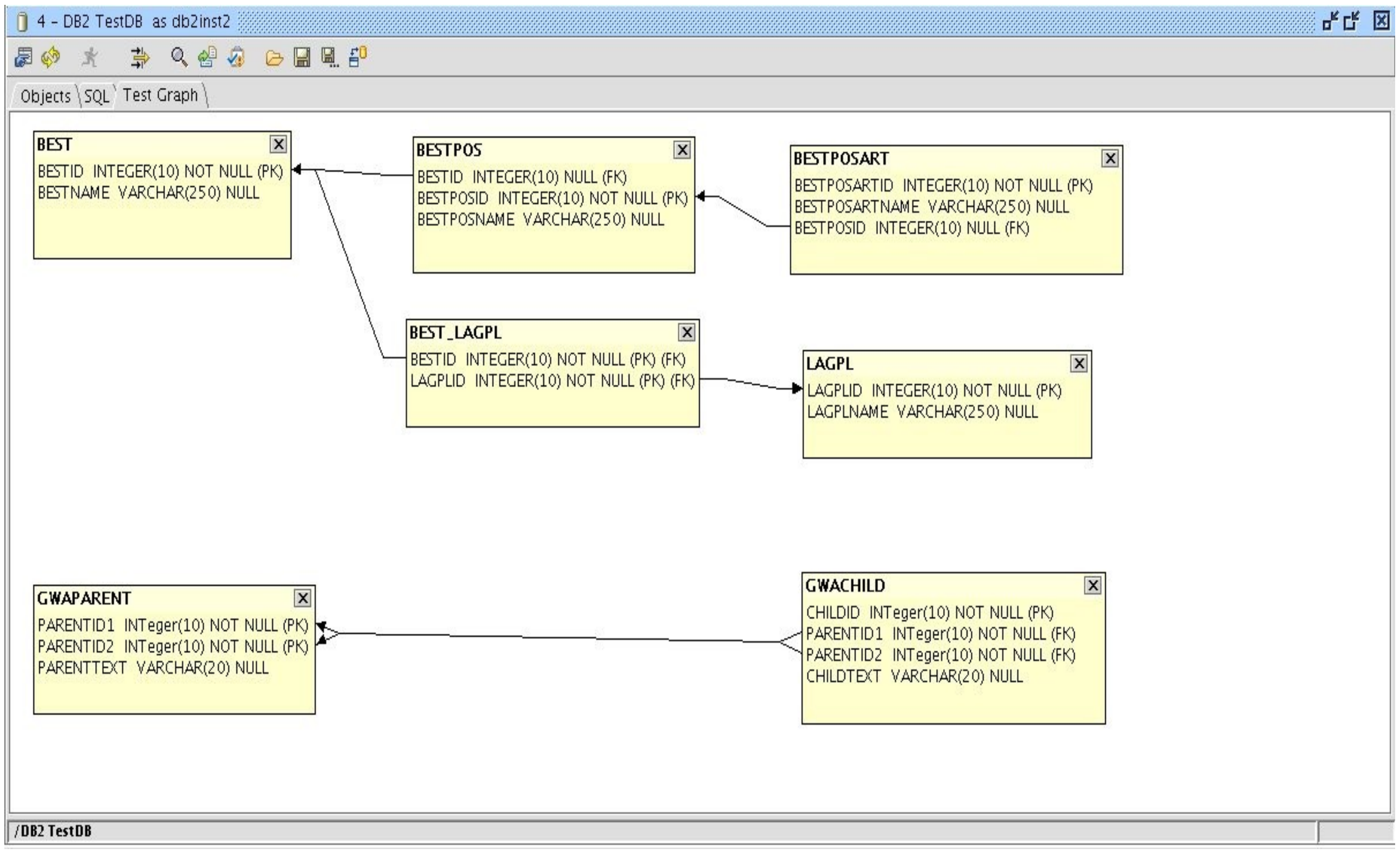
Script plugin
The Script plugin can create SQL scripts based on the tables displayed in the Objects tab or in a Graph. Insert scripts can be generated based on the entire contents of a table, or just the data selected using a WHERE clause. Scripts can also be created to store the results of an SQL query in a temporary table. In that case the user can choose to immediately run the script or to save it for later use.

SQLBookmarks plugin

With the SQL Bookmarks plugin the user can define SQL and DDL code templates. To quickly insert a template into the SQL editor you use the Ctrl + J shortcut. This opens a popup similar to the tools popup. The popup allows you to choose one of all existing templates.

The plugin includes predefined templates for the most common SQL and DDL problems. This makes learning SQL and DDL a lot easier.

Figure 6 Graph plugin



BEST
BESTID INTEGER(10) NOT NULL (PK)
BESTNAME VARCHAR(250) NULL

BESTPOS
BESTID INTEGER(10) NULL (FK)
BESTPOSID INTEGER(10) NOT NULL (PK)
BESTPOSNAME VARCHAR(250) NULL

BESTPOSART
BESTPOSARTID INTEGER(10) NOT NULL (PK)
BESTPOSARTNAME VARCHAR(250) NULL
BESTPOSID INTEGER(10) NULL (FK)

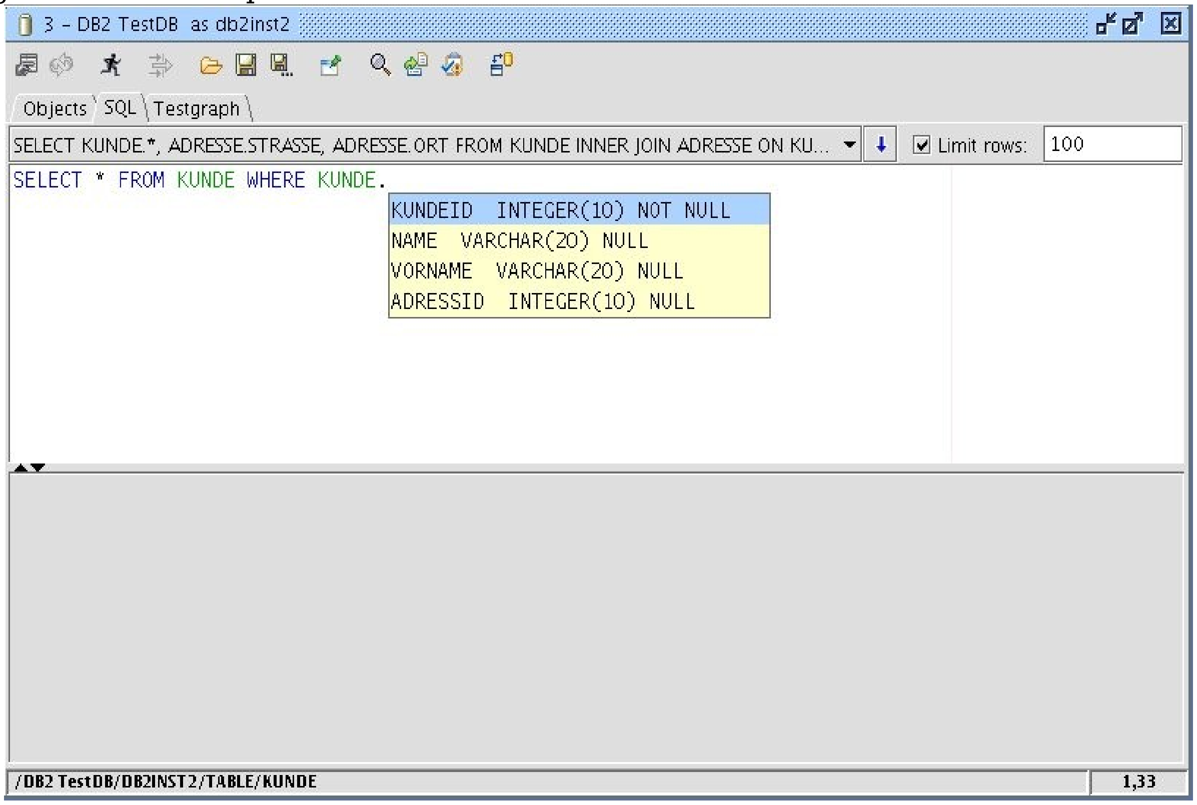
BEST_LAGPL
BESTID INTEGER(10) NOT NULL (PK) (FK)
LAGPLID INTEGER(10) NOT NULL (PK) (FK)

LAGPL
LAGPLID INTEGER(10) NOT NULL (PK)
LAGPLNAME VARCHAR(250) NULL

GWAPARENT
PARENTID1 INTeger(10) NOT NULL (PK)
PARENTID2 INTeger(10) NOT NULL (PK)
PARENTTEXT VARCHAR(20) NULL

GWACHILD
CHILDID INTeger(10) NOT NULL (PK)
PARENTID1 INTeger(10) NOT NULL (FK)
PARENTID2 INTeger(10) NOT NULL (FK)
CHILDTEXT VARCHAR(20) NULL

Figure 7 Code Completion



SELECT KUNDE.*, ADRESSE.STRASSE, ADRESSE.ORT FROM KUNDE INNER JOIN ADRESSE ON KU...

```
SELECT * FROM KUNDE WHERE KUNDE.
```

```
KUNDEID  INTEGER(10) NOT NULL
NAME  VARCHAR(20) NULL
VORNAME  VARCHAR(20) NULL
ADRESSID  INTEGER(10) NULL
```

How SQuirreL can help you

We said that SQuirreL will help make life easier for people who use databases, but different users have different needs depending on the role that they are playing. In this section we will look at how SQuirrel can support those roles. We will also describe some of SQuirreL's functions in more detail.

- Application administrators
  The role of Application Administrator requires fixing problems with an application, including those related to the data in the database. A typical scenario may be that an application user informs the administrator of a data problem that can not be solved through the application itself. This might be an incorrect value in a field, a missing entry in a relationship table, an extra row in a table, or any other abnormal problem. Using SQuirreL's Object tab view, the administrator can quickly locate the source of the problem by viewing the contents of the database tables as tables on the screen. Incorrect values can be corrected simply by typing in the new value. Rows in the tables can be inserted or deleted by a few mouse clicks. For very large tables, SQuirreL provides a popup window that lets you limit the rows retrieved by adding conditions to the WHERE clause used to read the data.

  When the problem involves complex data relationships, the SQL tab view can be used to enter SQL queries. The query can include any standard and vendor-specific SQL that your database engine will handle. If you need to repeat a query or make a request similar to a previous one you can grab one or more lines from the SQL history, edit them as needed and then execute the new query. If the result comes from a single table in the database, the data can be edited on the screen and the table updated immediately. To make simple corrections you can switch to the Object tab, or you can stay in the SQL tab to make more complex updates using the full SQL capabilities.

  Complex tasks that are repeated (nightly or monthly cleanup, periodic re-loading tables of product descriptions, etc.) can be automated using the SQL Scripts plugin.

  All of SQuirreL's tables allow you to selectively copy data as either a space-separated list or as HTML format which can be pasted into documents or web pages.

- Software developers, maintainers, and Testers
  When writing software or testing an application, you need to be able to exercise specific functions and to test error paths. The most important ways that SQuirreL supports software developers these efforts are:
  1. Setting up data before running a test, checking the data after running a test, and fixing data that was corrupted during the test.
  2. Quickly looking at and fixing data in a System Test or Production environment
  3. Giving simple, consistent remote access to multiple machines, eg: such as unit

test, system test and production.

4. Providing a nice environment for figuring out what should be in an SQL statement, making it easy to create/edit existing SQL, tweak data to support the SQL test if needed, run the SQL and examine the results.

Many of these tasks are very similar to those needed by an application administrator, requiring exploring and adjusting data in the database, so the descriptions in the Application Administrator section are relevant here as well. The main difference is that developers and testers must frequently wipe out old data and restore a known set of test data. The SQL Script plugin mentioned in the previous section can aid that task.

The Code Completion plugin provides completion of SQL and DDL code and generates SQL join syntax.

The Syntax plugin does syntax highlighting and allows you to create user defined abbreviations and auto corrections. Together with the user defined Code templates introduced by the SQL-Bookmarks plugin, SQuirreL is a powerful code generator.

Quite often developers need to test SQL strings from program code in an SQL tool, or an SQL statement was written and tested in an SQL tool and now needs to be put into the program code. In these situations string delimiters need to be added to or removed from the statement. With the help of the Edit Extras plugin this can be done with one mouse click.

Another common problem for developers is to analyze SQL strings from traces or debugging output. Quite often these strings come without any formatting, e.g. a long string with no line feeds. The Edit Extras plugin provides easy formatting of SQL statements. Insert statements are formatted in such a way that column names are placed above their values. The plugin will also format Timestamp/Date/Time values in the standard JDBC syntax with a single mouse click.

Many applications make use of Views or Stored Procedures. Views and Stored Procedures may be executed in SQuirreL, but there is no SQL standard for storing them. Since the method for reading and editing them is product dependent, you will need a product-specific plugin to provide that ability. Several of these exist, but you may need to create one for your database engine. We will show how easy that is in the 'Programming plugins' section.

As a side note, developers of JDBC drivers should find that SQuirreL is a convenient test bed. Not only are drivers easy to add and replace, but SQuirreL exercises all aspects of a JDBC driver and provides the results in easy-to-read tables, so developers HTML format which can be pasted into documents or web pages.

- Database administrators

For Database administrators SQuirrel provides easy and consistent access to all of the local and remote machines they are responsible for. They can use SQuirreL to examine parameters in databases or check table sizes.

Compared to other user groups database administrators more often have to deal with product specific quirks of databases. Because they are non-standard, those quirks cannot be supported by SQuirreL's main code. On the other hand, SQuirreL's plugin API is sophisticated enough that it can even support critical database functions. Since the plugin is Java code, it has the ability to start a separate process, so even when a function is not available through a vendor's API, the plugin may still be able to perform it by calling the vendor's tools. The advantage of doing it this way would be to relieve the administrator of knowing the details of those tools.

There are already several product-specific plugins. Nonetheless we would like to urge database administrators who are interested in Java programming to have a look at the 'Programming plugins' section and to write product specific plugins for SQuirreL. If you have to administer more than one database product you will appreciate a client that unifies standard features as well as handling product-specific variations.

- People Learning or Teaching SQL
  For someone who is just starting to learn SQL, the SQL Bookmarks plugin provides code templates for many common SQL and DDL statements. The plugin also allows you to define your own templates.

  The Syntax plugin does an on the fly SQL syntax check and flags errors using colored text. If the mouse pointer is placed on an error, an error message tool tip is shown. SQuirreL's Graph plugin (figure 6) generates a graphical view of the tables and the foreign keys in the database and thus gives students an easy overview of the relational structures. The scripting functions of graphs allow the user to re-translate the visual representation back into abstract DDL code and thus clarify the connection of both.

  For students it is important to understand that all relational databases follow common principles that are vendor independent. Because SQuirreL allows unified access to almost all relational databases it particularly points out these principles.

- Understanding an existing database
  When joining an existing project it takes time to understand how the database is organized. SQuirreL's Graph plugin can help by allowing you to see the tables and their relationships arranged visually in one or more graphs. The graphs can be saved under user-defined names and are reopened when the session is opened next time. Also graphs may be scaled arbitrarily and be printed over several paper pages.

  Writing SQL statements can be difficult when you are not familiar with the table and column names. The Code Completion plugin can auto-complete SQL and DDL code, much in the way that IDEs help with code completion. The completion pop up window for columns (figure 7) shows the column names as well as their most important properties. In case those properties are not enough you may place the cursor on a table or view name and the use the right mouse menu to jump to its definition in the Object tree.

Programming plugins

We actively encourage people to write plugins for SQuirreL. Since plugins are independent of the core SQuirreL code, we pretty much let people do what they want. On the other hand, if you think you would like to build one, it is a good idea to get in touch with us first. For one thing, someone else may be working on that same feature already. Another reason is that you may need an enhancement to the plugin API. If you are building a feature that you need, then it will probably be useful to other people as well. We would like to work with you to make your plugin available under the same LGPL license as SQuirreL, but if that is not possible you can still count on our support. You can reach us through the squirrel-sql-develop mail list under Mailing Lists on the www.squirrelsql.org home page.

We will explain plugin programming by creating a simple example that displays the definition of a View or Stored Procedure. It does this by copying the definition of the View or Stored Procedure to the SQL editor, though any changes made there do not cause changes in the database. To change Views or Stored Procedures the user would have to run the usual DDL commands himself. SQuirreL needs to use a plugin to access those definitions because each database vendor stores and retrieves those definitions in a proprietary manner.

While the code in the example is specific to IBM's DB2 database, it can used as a template for creating plugins that work with other vendors' products.

The complete example code is available in the sql12 module of SQuirreL's CVS repository. At www.squirrelsql.org you can find all information needed to check out the module. The example code will be in the plugins/example/ subdirectory of your checkout directory.

To make SQuirrel work with a plugin the complied class files of the plugin need to be packed to a Jar file. The Jar file must be copied to the plugins directory of your SQuirreL installation. During startup SQuirreL loads the plugin Jar files and searches each Jar for an implementation of the IPlugin interface. This implementation is the central class of a plugin. In our example this class is named ExamplePlugin. ExamplePlugin is derived from DefaultSessionPlugin which then implements IPlugin. The central methods of ExamplePlugin are initialize() and sessionStarted(), see listing 1. To check if SQuirreL correctly loaded a plugin you may choose the menu item „Plugins“ --> „Summary“.

The initialize() method is called once during SQuirreL's startup. In our implementation of initialize() the resources e.g. the file example.properties is loaded. This file contains the labels of the menu items we are going to add to the Object tree's context menus. The labels are '(DB2) Script View' and '(DB2) Script Procedure'.

The sessionStarted() method will be called when a Session is opened. In our example sessionStarted() implementation we first check if the Session is for a DB2 database. If not the plugin will remain inactive. If it is an DB2 database we add the menu items to the context menu of the Object tree's view and stored procedure nodes. If for example the user selects the '(DB2) Script View' item the actionPerformed() method of class ScriptDB2ViewAction is called, see listing 2.

In the actionPerformed() method of ScriptDB2ViewAction we first gather the objects selected in the tree. Further down we need the 'simple name' attribute of these objects. The simple names are nothing but the names of the views. Then follows the product specific code: We access the IBM DB2 specific system table 'SYSIBM.SYSVIEWS'. The rest of the code is needed to output the view definition to the SQL editor.

The code to script stored procedures is very similar to the view scripting code and will not be presented here.

Listing 1 ExamplePlugin:

```
public synchronized void initialize() throws PluginException
{
   // Initializing the resources in example.properties.
   // example.properties contains the labels for the menu items.
   _resources = new PluginResources
      ("net.sourceforge.squirrel_sql.plugins.example.example", this);
}


/**
 * Called when a session is opened.
 * The menu items are added are added here.
 *
 * @param   session  The started Session.
 */
public PluginSessionCallback sessionStarted(ISession session)
{
   try
   {
      Connection con = session.getSQLConnection().getConnection();
      String driverName =
         con.getMetaData().getDriverName().toUpperCase();

      if(false == driverName.startsWith("IBM DB2 JDBC"))
      {
         // The plugin only knows how to script views and stored
         // procedures for the IBM DB2 database. So if this is
         // not an DB2 Session we tell SQuirreL not
         // to use the plugin.
         return null;
      }

      // Add entries to the view and stored procedure
      // nodes in the Object tree.
      IObjectTreeAPI otApi =
         session.getSessionInternalFrame().getObjectTreeAPI();

      ScriptDB2ViewAction viewAct =
         new ScriptDB2ViewAction(getApplication(),
                                 _resources,
                                 session);

      otApi.addToPopup(DatabaseObjectType.VIEW, viewAct);


      ScriptDB2ProcedureAction procAct =
         new ScriptDB2ProcedureAction(getApplication(),
                                      _resources,
                                      session);

      otApi.addToPopup(DatabaseObjectType.PROCEDURE, procAct);

      // ...
```

Listing 2 ScriptDB2ViewAction:

```
public class ScriptDB2ViewAction extends SQuirreLAction
{
   private ISession _session;


   public ScriptDB2ViewAction(IApplication app,
                              Resources rsrc,
                              ISession session)
   {
      super(app, rsrc);
      _session = session;
   }

   public void actionPerformed(ActionEvent evt)
   {
      try
      {
         Statement stat =
            _session.getSQLConnection().createStatement();

         SessionInternalFrame sessMainFrm =
            _session.getSessionInternalFrame();

         IDatabaseObjectInfo[] dbObjs =
            sessMainFrm.getObjectTreeAPI().
               getSelectedDatabaseObjects();
         StringBuffer script = new StringBuffer();
         for (int i = 0; i < dbObjs.length; i++)
         {
            ITableInfo ti = (ITableInfo) dbObjs[i];

            ////////////////////////////////////////////////////////
            // IBM DB 2 specific code to read view definitions.
            String sql =
               "SELECT TEXT " +
               "FROM SYSIBM.SYSVIEWS " +
               "WHERE NAME = '" + ti.getSimpleName() + "'";

            ResultSet res = stat.executeQuery(sql);
            res.next();

            script.append(res.getString("TEXT"));
            script.append(getStatementSeparator());
            res.close();
            //
            ////////////////////////////////////////////////////////
         }

         stat.close();

         sessMainFrm.getSQLPanelAPI().
            appendSQLScript(script.toString());

         sessMainFrm.getSessionPanel().
            selectMainTab(ISession.IMainPanelTabIndexes.SQL_TAB);
      }
      catch (Exception e)
      {
         throw new RuntimeException(e);
      }
   }

   /**
    * Returns the user defined statement separator with
    * suitable line feeds.
    */
   private String getStatementSeparator()
   {
      String statementSeparator =
        _session.getQueryTokenizer().getSQLStatementSeparator();

      if (1 < statementSeparator.length())
         statementSeparator = "\n" + statementSeparator + "\n";
      else
```

```
         statementSeparator += "\n";

      return statementSeparator;
   }
}
```

Conclusion

SQuirreL provides a single, uniform interface that works with all relational databases. The user interface provides features similar to modern IDEs and makes it easy to access and modify databases, understand database structure, and work with and learn SQL. SQuirreL emphasizes the things that all databases have in common while being able to handle product-specific quirks through the plugin architecture. New features can also be quickly added by creating new plugins.

SQuirreL is helping people to be more productive in many different environments every day. The users of SQuirreL are our best source for ideas for improvement. Anyone with a suggestion can contact us at squirrel-sql-users@lists.sourceforge.net.

Finally, a word to software developers. SQuirreL developers are motivated both by the fun of programming and the desire to add functions that they need. It is interesting to explore the world of relational databases using Java, and it is a real kick to see a feature that you have built being used by people around the world. Working with the SQuirreL code is a great way to learn how to use Java to create extremely flexible applications. We invite you to join us. You can reach us at
squirrel-sql-develop@lists.sourceforge.net.