SQuirreL, ein universeller SQL Client

von Gerd Wagner und Glenn Griffin

Verwenden Sie eine relationale Datenbank? Falls ja sind Sie wahrscheinlich mit der ein oder anderen folgenden Situationen vertraut:

- Um einen einzigen Wert in der Datenbank zu ändern, müssen Sie ein umständliches Update Statement schreiben.
- Sie müssen dasselbe SQL mit nur kleinen Änderungen wieder und wieder schreiben.
- Sie arbeiten mit verschiedenen Datenbanken auf verschiedenen Servern.
- Sie benutzen Datenbanksysteme von verschiedenen Herstellern.
- Als Lernender sind Sie gerade dabei, sich in relationale Datenbanken einzuarbeiten.
- Sie müssen sich mit den Datenstrukturen einer vorhandenen Datenbank vertraut machen.

Allen, die sich hier angesprochen fühlen, kann SQuirreL die Arbeit erleichtern.

Was ist SQuirreL?

SQuirreL ist ein einfacher grafischer SQL Client für relationale Datenbanken. Weil SQuirreL in Java geschrieben ist, läuft es auf jeder Java-fähigen Plattform und kann auf jede Datenbank zugreifen, für die ein JDBC-Treiber existiert. Mit SQuirreL kann man

- Daten ansehen und in der Ansicht editieren,
- Datenbank-Metadaten ansehen,
- mit mehreren Datenbanken auf beliebigen Servern arbeiten,
- von einer einheitlichen Oberfläche aus mit verschiedenen Datenbanksystemen arbeiten,
- mit Hilfe produktspezifischer Plugins, spezielle Funktionen des jeweiligen Datenbanksystems einbinden.

Der Benutzer kann die Daten einzelner Datenbanktabellen direkt ansehen und bearbeiten oder er kann SQL Statements ausführen. Die Daten können der Sicherheit halber im Read-Only-Modus angezeigt werden oder im Editiermodus. Im Editiermodus können die

Daten einfach durch editieren der Zellen geändert werden (siehe Abbildung 1). Alle Metadaten einer Datenbank, zum Beispiel Datentypen, Tabellendefinitionen einschließlich Indices und Constraints, sind mit SQuirreL sichtbar. Wo Datenbanken von verschiedenen Herstellern benutzt werden, muss der Benutzer nicht die spezifischen Tools eines jeden Herstellers erlernen, da SQuirreL einen einheitlichen Zugriff auf alle Systeme ermöglicht. Falls ein Datenbanksystem produktspezifische Funktionen zur Verfügung stellt, kann der Benutzer entsprechende produktspezifische Plugins einbinden, um von SQuirreL aus auf diese Funktionen zuzugreifen. Allgemein bietet die Plugin Architektur Programmierern die Möglichkeit, Funktionen zu SQuirreL hinzuzufügen, die von anderen Benutzern nach belieben eingebunden werden können.

Die Abbildungen 1 und 2 zeigen SQuirreL in Aktion. Abbildung 1 zeigt den den einfachen Zugriff auf eine einzelne Tabelle (die gelbe Zelle wird gerade editiert). Abbildung 2 zeigt den SQL Editor mit Ausgabe.

Abbildung 1: Ansehen und ändern von Daten einer einzelnen Tabelle.

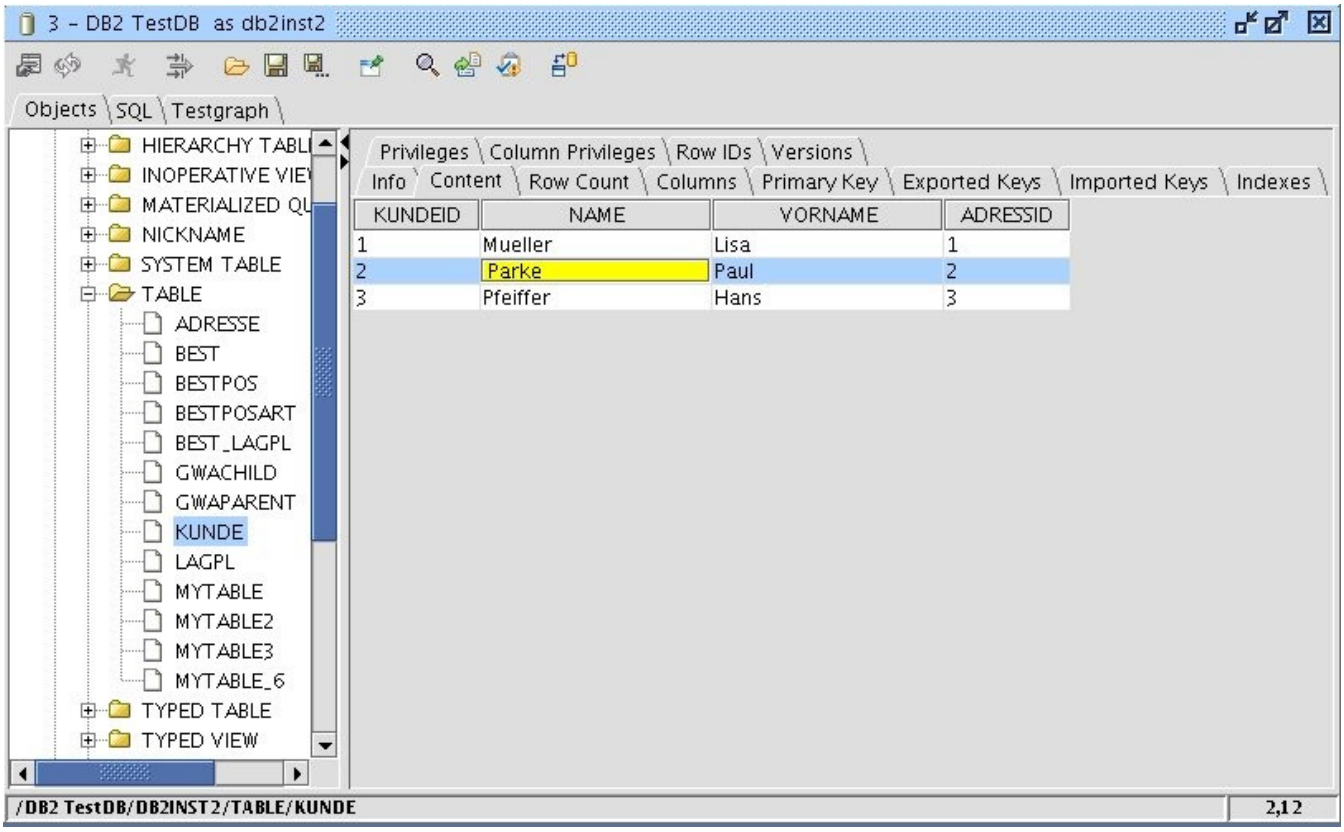
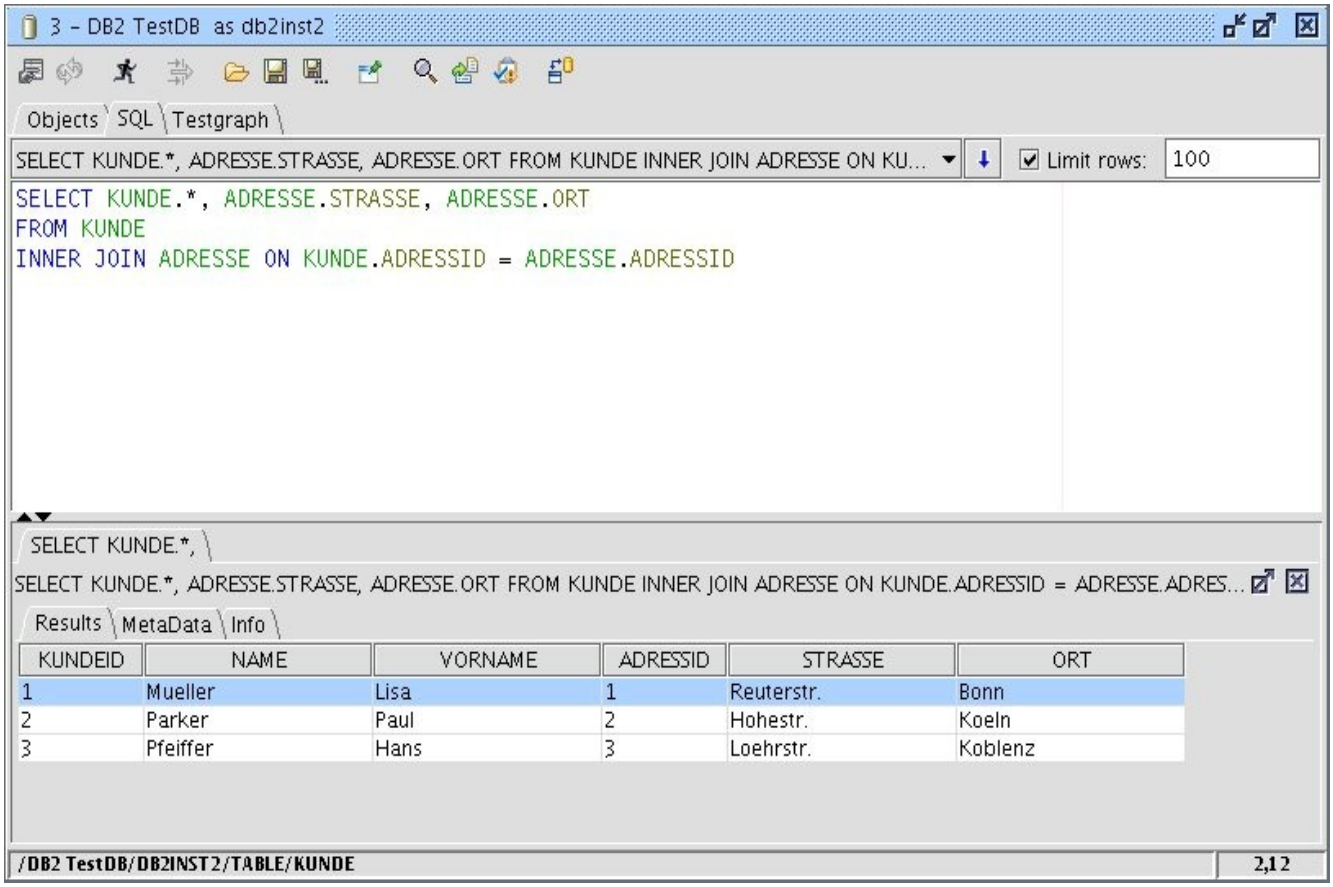


Abbildung 2: Ausführen von SQL und DDL Anweisungen.



Hintergrund

Durch den JDBC Standard und durch die großflächige Verfügbarkeit von JDBC Treibern kann die Java Plattform auf nahezu alle relationalen Datenbanksysteme zugreifen. JDBC bietet Softwareentwicklern ein bis heute unerreichtes Maß an Einheitlichkeit und Einfachheit beim Zugriff auf Datenbanken. Der Open Source Client SQuirreL setzt sich zum Ziel, diese Vorteile auch dem Datenbankbenutzer zu gute kommen zu lassen.

Im nächsten Abschnitt beschreiben wir die Installation von SQuirreL und wie der Zugriff auf eine Datenbank konfiguriert wird, anschließend besprechen wir eine Reihe nützlicher Plugins. Danach stellen wir die Zielgruppen vor, an die sich SQuirreL wendet, und beschreiben wie SQuirreL sie bei ihrer Arbeit unterstützt. Zuletzt zeigen wir die Schritte, die zur Entwicklung eines Plugins nötig sind, und besprechen den Programm Code eines Beispiel-Plugins.

Download und Installation

SQuirreL 2.0 final kann kostenlos unter der LGPL Lizenz von www.squirrelsql.org heruntergeladen werden. Auf dieser Seite finden Sie das SQuirreL Installationspaket und eine Reihe von Plugins. Das Installationspaket enthält alles was für den Anfang benötigt wird. Zu dem Installationspaket gehören Standard-Plugins, die für die meisten Benutzer nützlich sind. Es ist nach der Installation leicht möglich, weitere Plugins hinzuzufügen, so dass es wahrscheinlich sinnvoll ist, zunächst die ersten Schritte mit dem Standardpaket zu tun.

Für die Installation und zur Ausführung von SQuirreL wird ein Java Runtime Environment (JRE) der Version 1.4.x oder höher benötigt.

Die Installations-Jar-Datei ist direkt ausführbar. Im Rahmen der Installation kann der Benutzer entscheiden, ob er die Basis oder die Standard Version installieren möchte. Der Unterschied zwischen beiden besteht ausschließlich in den bereits erwähnten Standard-Plugins. Bei den Standard-Plugins handelt es sich um:

- Code Vervollständigung – Vervollständigung von SQL und DDL Code wie bei gängigen IDEs
- Syntax – SQL Syntax Hervorhebung sowie Abkürzung und Autokorrektur
- Edit Extras – Hilfsfunktion zum Arbeiten mit SQL, z.B. SQL Code Formatierung
- Graph – Grafische Darstellung von Tabellen und ihren Foreign Key Beziehungen
- SQL Script – Generierung von SQL und DDL Scripten
- SQL Bookmarks – Verwaltung von SQL Code Vorlagen
- Look and feel – Unterstützung einer Vielzahl von Look and Feels

Neben dem Installationsverzeichnis können für den Benutzer zwei weitere Verzeichnisse von Interesse sein: Direkt unterhalb des Installationsverzeichnisses befindet sich das Plugins Verzeichnis. Wenn Sie nachträglich ein Plugin installieren wollen, müssen Sie es nur in dieses Verzeichnis entpacken. Bei nächsten Neustart von SQuirreL wird das Plugin automatisch geladen. Das zweite Verzeichnis wird erstellt, wenn SQuirreL zum ersten mal startet. Es enthält Informationen zu benutzerbezogenen Einstellungen. Dieses Verzeichnis liegt im Benutzerordner/Home directory und trägt den Namen „.squirrel-sql“.

Nach diesen Schritten sind Sie in der Lage, SQuirreL zu starten. Bevor wir die Verwendung des Programms beschreiben noch der Hinweis, dass auf www.squirrelsql.org wöchentlich ein Installationspakte zur Verfügung steht, dass einen Snapshot des aktuellen Entwicklungsstandes enthält. Die Snapshots sollen erfahrenen Benutzern die Möglichkeit bieten, sich mit den SQuirreL Entwicklern kurzfristig auszutauschen.

Die Datenbankverbindung

Die erste Verbindung zu einer Datenbank herzustellen, ist nicht immer einfach. Mit SQuirreL gehören dazu zwei Schritte:

1. Beschaffen und Einbinden des JDBC Treibers
2. Definition einer Verbindung zu der gewünschten Datenbank

In SQuirreL bezeichnen wir diese beiden Schritte als Konfiguration des Treibers und die Definition eines „Alias“. Einen Alias kann man sich als eine Instanz einer Treiberkonfiguration vorstellen.

Immer wenn SQuirreL gestartet wird, werden das Treiber (Drivers) und das Alias Fenster auf dem SquirreL-Desktop geöffnet, siehe Abbildung 3. Im Treiber Fenster sind Treiber, die bereits fertig konfiguriert sind, mit einem blauen Haken versehen. Treiber, die noch nicht konfiguriert sind, und für die SQuirreL nur eine Vorlage für die Konfiguration bereit hält, sind mit einem roten X markiert. Um einen Treiber zu konfigurieren, müssen Sie den Treiber zunächst von Ihrem Datenbankhersteller beziehen und auf Ihren lokalen Rechner

kopieren. Anschließend müssen Sie SQuirreL über das Treiber Fenster mitteilen, wo die Treiberdateien liegen. Dabei können Sie entweder eine Konfigurationsvorlage verwenden oder eine komplett neue Treiberkonfiguration erstellen.

Als nächstes ist ein Alias zu definieren, der die Daten für eine Verbindung zu einer bestimmten Datenbank auf einem bestimmten Server enthält (Abbildung 4). Zur Definition des Alias gehört die Angabe der URL für die Datenbank. Dies ist gewöhnlich der schwierigste Aspekt der Verwendung von JDBC, weil jeder Treiberhersteller ein eigenes Format für die URL festlegt. Um dem Benutzer die Definition eines Alias zu erleichtern, gehört bereits zu jeder Treiberkonfigurtion eine Beispiel URL, die während der Alias Definition automatisch eingeblendet wird, wenn der Treiber für den Alias ausgewählt wird.

Nachdem Sie den Alias definiert haben, können Sie die Verbindung zur Datenbank durch Doppleklick auf den Aliasnamen herstellen.

Abbildung 3: Der SQuirreL Desktop

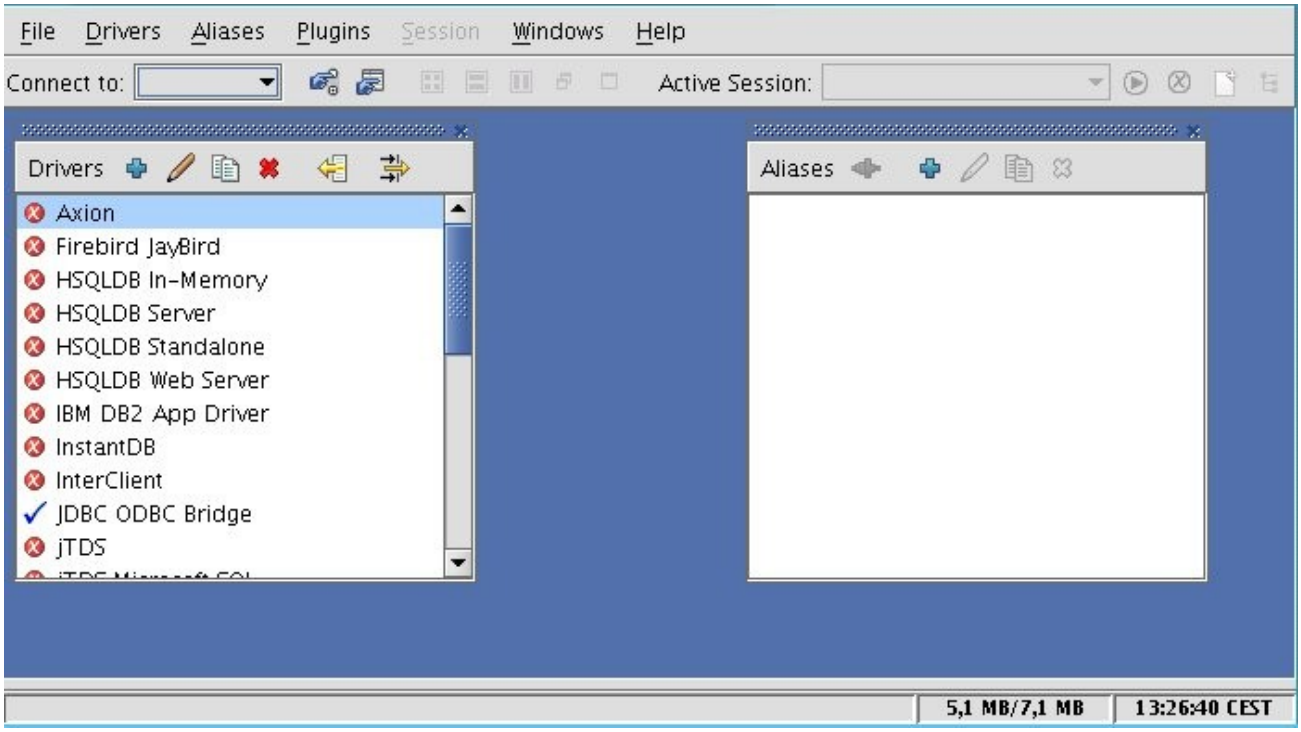
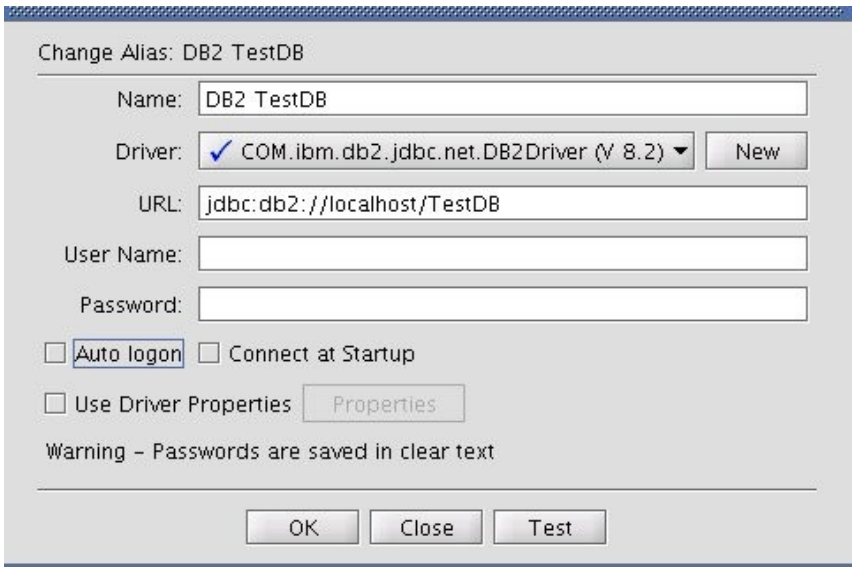


Abbildung 4: Definition eines Alias.



Arbeiten mit einer Datenbank – die Session

Wenn eine Datenbankverbindung mit SQuirreL hergestellt wird, öffnet sich ein Session-Fenster. Jede Session gehört zu genau einer Datenbankverbindung. Mehrere Session können gleichzeitig geöffnet sein.

SQuirreL verfolgt den Grundsatz, dass einfache Dinge einfach getan werden sollen und komplexe Dinge so einfach möglich. Dementsprechend bietet die Session zwei Möglichkeiten, mit einer Datenbank zu arbeiten. Beide werden durch jeweils ein Tab auf dem Session Fenster repräsentiert.

Das Objects-Tab (Abbildung 1) liefert eine einfache tabellarische Sicht auf die Datenbank. Sämtliche Datenbank Metadaten (Datentypen, Übersicht über unterstütze Funktionen,...) sind in tabellarischer Form über den Baum auf der linken Seite des Tabs zugänglich. Wird in dem Baum eine Tabelle angeklickt, dann stehen die Daten der Tabelle selbst sowie sämtliche Tabellenmetadaten (Spaltendefinitionen, Indices, Constraints, ...) zur Verfügung. Die Daten der Tabelle können in Textform, in einer Read-Only-Tabelle oder als editierbare Tabelle dargestellt werden. Wenn die Tabelle editierbar ist, dann führt das Ändern der Zelleninhalte sofort zur Änderung der Daten in der Datenbank. Es ist möglich, Zeilen neu einzufügen oder zu löschen. Darüber hinaus können Daten in eine Datei exportiert oder aus einer Datei importiert werden. Dabei werden alle Standard Datentypen einschließlich BLOBs und CLOBs unterstützt. Alle beschriebenen Änderungen können, wenn gewünscht, in bentutzergesteuerte Transaktionen gebettet werden.

Das SQL-Tab (Abbildung 2) ermöglicht das Ausführen von SQL-Befehlen. Das Objects-Tab ist zwar einfach zu benutzen, erlaubt aber keine komplexen Operationen wie zum Beispiel Abfragen auf mehrere Tabellen, Strukturänderungen von Datenbankobjekten oder das Ausführen von Stored Procedures. Das SQL-Tab erlaubt es, komplexe Anweisungen direkt an die Datenbankengine zu schicken. Die Ergebnisse der Anweisungen können, sofern sie von tabellarischer Struktur sind, wahlweise in textueller Form, als Read-Only-Tabelle oder, wenn eine Abfrage sich auf nur eine einzige Tabelle bezieht, in editierbarer Form dargestellt werden. Die Ergebnisse werden in der unteren Hälfte des SQL-Tabs

ausgegeben und enthalten Metadateninformationen zum jeweilige Ergebnis. Schließlich enthält das SQL-Tab eine SQL-History Liste, aus der früher verwendete SQL-Anweisungen wieder abgerufen werden können.

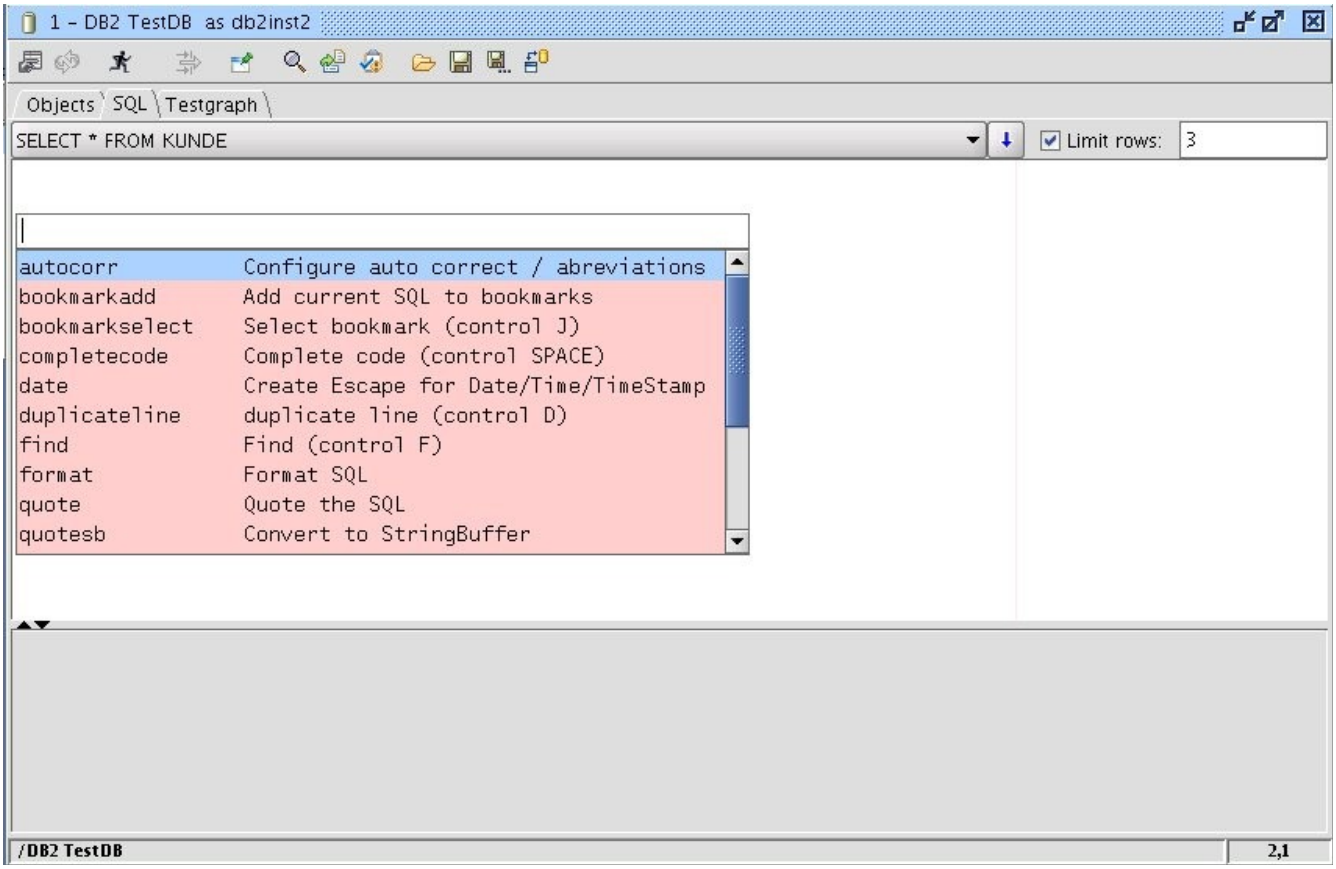
Insbesondere Plugins fügen dem SQL-Editor viele Funktionen hinzu. Um dem Benutzer die Übersicht über die Funktionen und ihren Aufruf zu erleichtern, steht das Tools-Popup zur Verfügung, das mit der Tastenkombination Strg+T zu öffnen ist (Abbildung 5). Das Tools-Popup stellt jede Editorfunktion mit einem Selektionkürzels, einer kurzen Beschreibung und, sofern vorhanden, der eigenen Tastenkombination dar. Der Inhalt des Popups kann durch Eingabe des Beginns eines Selektionkürzels gefiltert werden. Mit Hilfe des Tools-Popup sind sämtlich Funktionen des Editors über die Tastatur zugänglich und man benötigt nur eine Tastenkombination, nämlich Strg+T.

Wie die meisten Programme, die einen größeren Benutzerkreis ansprechen, erlaubt SQuirreL viele benutzerbezogene Einstellungen. Es gibt zwei Arten von benutzerbezogene Einstellungen:

1. Globale Einstellungen, die für alle Sessions gelten. Zum Beispiel, welche Toolbars angezeigt werden sollen, ob Tooltips angezeigt werden sollen, JDBC Timeout und Debug Einstellungen oder wie bestimmte Datentypen etwa Datum/Zeit, BLOB/CLOB dargestellt werden sollen.
2. Session Einstellungen beziehen sich auf eine individuelle Session. Es können aber auch Default-Einstellungen für Session erfasst werden. Zu den Sessioneinstellungen gehören: das Ausgabeformat von Daten (Text, Read-Only, Editierbar), Begrenzung der Größe von Ausgabetabellen, ob benutzergesteuerte Transaktionen verwendet werden sollen, die Positionierung von Tab-Reitern.

Insgesamt gibt es duzende von Einstellungen. Für die ersten Schritte sollten jedoch die Defaults angemessen sein. Der Benutzer kann später Schritt für Schritt SQuirreL nach seinem Geschmack konfigurieren.

Abbildung 5: Tools Popup



Plugins

In den Tabellen 1 bis 3 geben wir einen kurzen Überblick über die Plugins, die bei www.squirrelsql.org verfügbar sind. Auf den Überblick folgt eine detailiertere Vorstellung der 5 wichtigsten Standard Plugins.

Tabelle 1: Standard Plugins (Teil der SQuirreL Standard Installation)

Name	***Beschreibung***
Look and Feel	Unterstützung einer Vielzahl von Look and Feels.
SQL Bookmarks	Verwaltung von SQL Code Vorlagen.
SQL Scripts	Generierung von SQL und DDL Scripten.
Graph Plugin	Grafische Darstellung von Tabellen und ihren Foreign Key Beziehungen.
Edit Extras	Hilfsfunktion zum Arbeiten mit SQL, z.B. SQL Code Formatierung.
Code Completion	Vervollständigung von SQL und DDL Code wie bei gängigen IDEs.
Syntax	SQL Syntax Hervorhebung sowie Abkürzung und Autokorrektur.

Tabelle 2: Freigebene Plugins (nicht Teil der Standard Installation)

Name	*Beschreibung*
MySQL Plugin	Spezielle Funktionen für die MySQL Datenbank.
Oracle Plugin	Spezielle Funktionen für die Oracle Datenbank.
SQL Validator	Prüfung von Anweisungen gegen den SQL Standard. Das Plugin benutzt den Webservice von Mimer SQL.

Tabelle 3: Beta Plugins

Name	*Beschreibung*
Firebird Plugin	Spezielle Funktionen für die Firebird Datenbank.
Microsoft MSSQL Plugin	Spezielle Funktionen für die Microsoft SQL Server Datenbank.
Session Scripts	Erlaubt es Scripte zu definieren, die beim Start einer Session ausgeführt werden.

Das Code Completion Plugin
Code Vervollständigung (Abbildung 6) ist eine der meist genutzten Funktionen in modernen IDEs. Das Code Completion Plugin erlaubt die Vervollständigung von nahezu alle SQL-und DDL-Konstrukten:
- Schlüsselworte des SQL Standards sowie Schlüsselwörter, die vom JDBC Treiber des jeweiligen Datenbanksystems geliefert werden.
- Tabellen
- Spalten
- Views
- Stored Procedures, die Vervollständigung liefert die komplette Standard JDBC-Aufruf Syntax einschließlich Vorlagen für die Parameter
- Kataloge
- Schemas

Darüber hinaus bietet das Plugin sogenannte Completion Funtions, die zur Generierung von SQL-Joins dienen. Zur Erläuterung der Completion Functions betrachten wir ein Beispiel, das auf den Tabellen BEST, BEST_LAGPL und LAGPL aus Abbildung 7 basiert. Um den Join von BEST nach LAGPL zu generieren, würde man den folgenden Ausdruck im SQL-Editor eingeben:

#i,BEST,BEST_LAGPL,LAGPL,

Steht der Cursor am Ende dieses Ausdrucks und die Code Vervollständigung wird mit Strg+Leertaste aufgerufen, dann wird der folgende Code generiert:

INNER JOIN BEST_LAGPL ON BEST.BESTID = BEST_LAGPL.BESTID

INNER JOIN LAGPL ON LAGPL.LAGPLID = BEST_LAGPL.LAGPLID

Das Graph Plugin
Das Graph Plugin ermöglicht die Visualisierung von Tabellen und ihren Fremdschlüssel-Beziehungen. Initial fügt das Plugin lediglich die Funktion „Add to graph“ dem rechten Mausmenu der Tabellenknoten im Baum des Objects-Tab hinzu. Wenn der Benutzer diese Funktion anwählt, wird dem Session-Fenster ein weiters Tab hinzugefügt. Dieses Tab zeigt den Graphen der ausgewählten Tabellen an, siehe Abbildung 7.

Der Benutzer kann dem Session-Fenster eine beliebige Anzahl von Graph Tabs hinzufügen. Für die Tabs können Namen vergeben werden und sie können gespeichert werden. Beim nächste Öffnen derselben Verbindung werden die gespeicherten Graph Tabs automatisch mit geöffnet. Auf diese Weise stehen dem Benutzer die für Ihn wichtigsten Tabellen und Beziehungen immer in visueller Form zur Verfügung.

Fremdschlüsselspalten werden im Graphen mit einem (FK) am Ende dargestellt (Abbildung 7). Beim Doppelklick auf eine solche Spalte, wird dem Graph die Tabelle, aus der der Fremdschlüssel stammt, hinzugefügt. Über das rechte Mausmenu können alle Kind- und/oder alle Elterntabellen einer Tabelle dem Graph hinzugefügt werden. Auf diese Weise ermöglicht das Graph Plugin das Browsen durch Tabellenstrukturen.

Die SQuirreL Plugin API erlaubt es Plugins, miteinander zu kommunizieren. Auf diese Weise kann das Graph Plugin die Funktionen des SQL Scripts Plugins benutzen, um aus der visuellen Darstellung die zugehörigen abstrakteren DDL Befehle für Tabellen und Fremdschlüssel per Mausklick zu generieren.

Es ist möglich einen Graphen über eine oder mehrere Papierseiten beliebigen Formats zu Drucken. Dazu wird der Graph über das rechte Mausmenu in den Druckmodus geschaltet. Er kann dann beliebig skaliert werden. Zusätzlich können die Seitenränder von frei definierbaren Papierseiten eingeblendet werden und ebenfalls skaliert werden. So kann der Graph optimal auf die Seiten verteilt werden.

Das Syntax Plugin
Beim Syntax Plugin zeigt sich die Mächtigkeit der SQuirreL Plugin API. Das Plugin ersetzt einen zentralen Bestandteil von SQuirreL, nämlich den SQL Editor, durch eine neue Komponente, nämlich den Netbeans Editor (http://editor.netbeans.org). Das Plugin ist verantwortlich für die farbliche Hervorhebung der SQL Syntax Elemente, siehe Abbildung 2. Darüber hinaus bietet es die Möglichkeit, Abkürzungen und Autokorrekturen ähnlich wie in Office Produkten zu definieren.

Das SQL Script Plugin
Das Script Plugin kann DDL Scripte für Tabellen und Ihre Eigenschaften generieren. Diese Funktionen sind im Baum des Objects-Tab und in Graphen über die rechte Maustaste verfügbar. Für Tabelleninhalte können Insert Scripte entweder für alle Daten einer Tabelle oder für das Ergebnis einer SQL-Abfrage generiert werden. Für eine beliebige Abfragen kann ein Script generiert werden, das das Ergebnis der Abfrage in

einer temporären Tabelle speichert.

Das SQL Bookmarks Plugin
Das SQL Bookmarks Plugin erlaubt es, SQL und DDL Code Vorlagen zu definieren und zu verwalten. Um eine Vorlage in den SQL Editor zu übernehmen, benutzt man die Tastenkombination Strg+J. Dadurch wird ein Popup ähnlich dem Tools Popup geöffnet, aus dem die gewünschte Vorlage ausgewählt werden kann.

Das Plugin bringt eine Reihe von Default-Vorlagen für die häufigsten SQL und DDL Probleme mit. Auf diese Weise kann das Plugin eine Hilfe beim Erlernen von SQL und DDL sein.

Abbildung 6: Code Completion

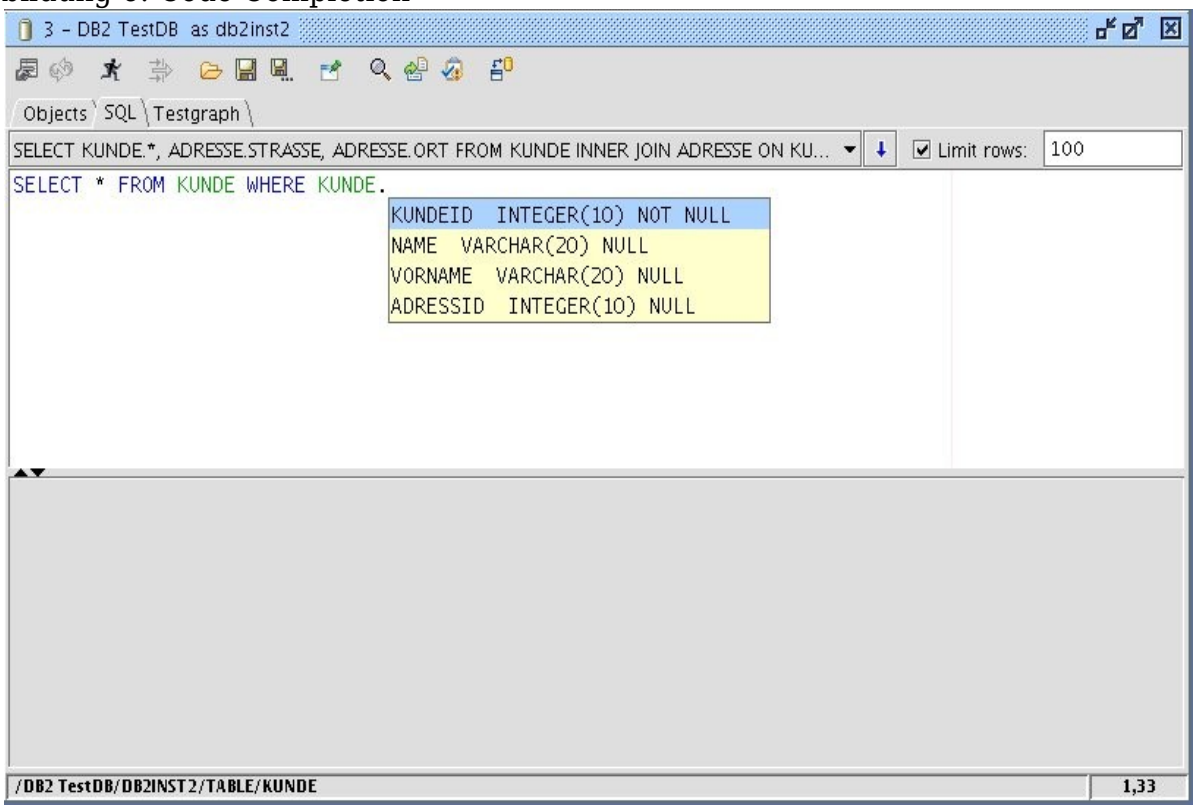
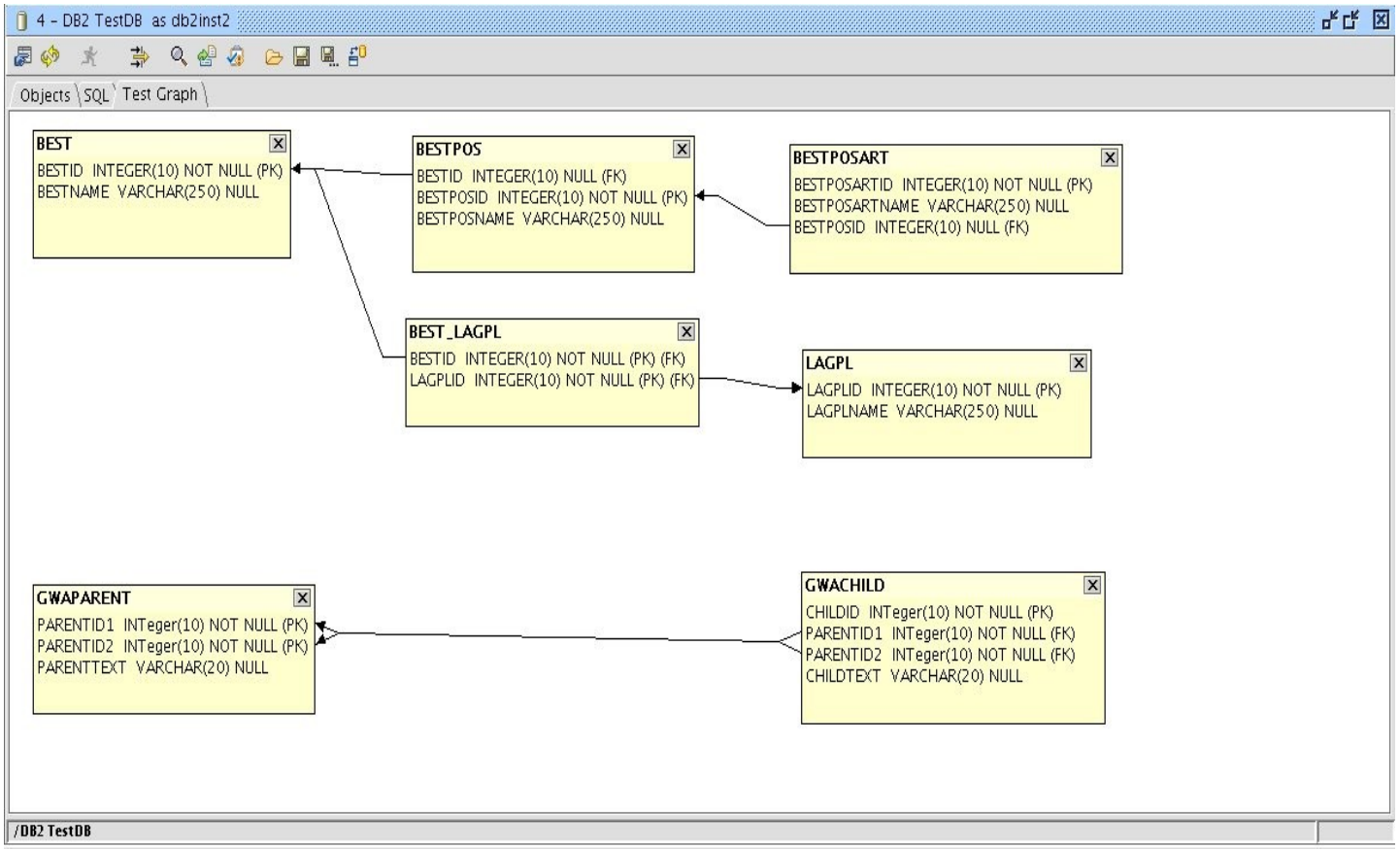


Abbildung 7: Graph plugin



Wie SQuirreL Sie bei der Arbeit unterstützt

SQuirreL kann Menschen, die relationalen Datenbanken benutzen, die Arbeit erleichtern. Verschiedene Benutzer haben abhängig von ihrem Betätigungsfeld verschiedene Ansprüche an SQuirreL. Im folgenden werden wir einige Betätigungsfelder diskutieren und erläutern wie SQuirreL diese unterstützt. Dabei werden auch weitere Funktionen von SQuirreL vorgestellt.

Anwendungsbetreuer
Die Rolle eines Betreuers einer Anwendung, die auf einer relationalen Datenbank basiert, besteht unter anderem darin, Datenproblem zu beheben, die über die Anwendung selbst nicht behoben werden können. Dabei könnte es sich um einen inkorrekten Wert in einem Feld, um einen fehlenden Eintrag in einer Beziehungstabelle oder ein ähnliches Problem handeln. Mit Hilfe der Tabellenanzeige im Objects-Tab kann der Betreuer das Problem lokalisieren und beheben indem er die Daten durch einfaches editieren der Tabelle ändert oder mit ein paar Mausklicks Zeilen hinzufügt oder löscht. Für große Tabellen bietet SQuirreL die Möglichkeit, in einem Dialog eine WHERE-Bedingung zu erstellen, um die Daten zu filtern.

Falls ein Problem eine komplexere Datenanalyse erfordert, müssen SQL Abfragen erstellt werden. Diese Abfragen können alle Standard-oder produktspezifische Elemente enthalten, die das jeweilige Datenbanksystem unterstützt. Für den Fall, das ähnliche

Abfragen mehrfach ausgeführt werden müssen, kann der Benutzer entweder die Abfrage aus der SQL-History heraussuchen oder mit Hilfe des SQL-Bookmark-Plugin die Abfrage unter einem Namen abspeichern und mit Strg+J wieder abrufen. Wenn das Resultat einer Abfrage auf einer einzigen Tabelle beruht, kann das Resultat der Abfrage direkt editiert werden.

Alle Tabellen in SQuirreL bieten die Möglichkeit, Daten selektiv als Tab-separierte Liste oder im HTML-Fomat zu kopieren, so dass sie direkt in Dokumente oder Web-Pages eingefügt werden können.

Softwareentwickler und Tester

Softwareentwickler und Tester sind möglicherweise die Benutzergruppen, denen SQuirreL die weitreichendste Unterstützung bietet. Die Tätigkeiten bei denen SQuirreL Entwickler und Tester hauptsächlich unterstützt sind:

1. Bereitstellen/Wiederherstellen der Datengrundlage für einen Test und Analyse der Daten nach einem Testlauf.
2. Auffinden und korrigieren von Daten in einer Test- oder Produktionsumgebung.
3. Erstellen von SQL-Statements sowie das Übertragen von SQL-Statements aus SQuirreL in den Applikations-Quellcode und zurück.

Mit Hilfe des SQL-Script Plugins können die Daten, die für einen Test benötigt werden, als Insert-Scripte abgelegt werden. Nach einem Testlauf können die veränderten Daten gelöscht und mit Hilfe der Scripte einfach wieder neu angelegt werden.

Die Code Vervollständigung zusammen mit der Join Generierung sowie die konfigurierbaren Abkürzungen vereinfachen das Schreiben von SQL beträchtlich.

Mit den Code Vorlagen des SQL-Bookmark-Plugins, den konfigurierbaren Abkürzungen und den Funktionen des SQL-Script Plugins verfügt SQuirreL über eine leistungsfähige Maschinerie zur Generierung von SQL und DDL-Code.

Wenn SQL-Statements zwischen Applikations-Code und SQuirreL ausgetauscht werden, müssen String Delimiter entweder eingefügt oder entfernt werden. Das Edit Extras Plugin bietet die Funktionen, um dies per Mausklick zu erledigen.

Nicht selten haben Softwareentwickler das Problem, SQL-Statements aus SQL Traces oder aus Debugausgaben zu testen und meist sind diese Statements nicht formatiert, typischerweise ein langer String ohne Zeilenumbruch. Das Edit Extras Plugin bietet die Möglichkeit, SQL Code zu formatieren. Dabei werden Insert-Values-Statements so formatiert, dass eine direkte Zuordnung von Spalte und Werte sichtbar ist.

Viele Applikationen benutzen Views oder Stored Procedures. Diese können leicht in SQuirreL ausgeführt werden. Leider gibt es aber keinen SQL-Standard dafür wie der Quellcode eines Views oder einer Stored Procedure in der Datenbank abgelegt wird. Deswegen werden produktspezifische Plugins benötigt, um den Quellcode anzuzeigen. Einige dieser Plugins existieren bereits. Im Abschnitt „Programmierung von Plugins“ werden wir ein solches Plugin erstellen.

Als Randbemerkung sei erwähnt, dass SQuirreL Entwicklern von JDBC Treiben eine effektive Testumgebung bietet. SQuirreL kann auf jeden Treiber zugreifen und es benutzt einen Großteil der Funktionen, die ein Treiber zur Verfügung stellt.

Datenbankadministratoren
Für Datenbankadministratoren bietet SQuirreL einen einfachen und einheitlichen Zugriff auf alle lokalen und entfernten Datenbanken, für die sie verantwortlich sind.

Im Vergleich zu anderen Benutzergruppen sind Datenbankadministratoren öfter mit produktspezifischen Eigenheiten von Datenbanksystemen befasst. Diese Eigenheit können von SQuirreLs Standard Funktionen nicht berücksichtigt werden. Jedoch ist SQuirreLs Plugin API mächtig genug, um solche Eigenheiten in produktspezifischen Plugins abzubilden.

Es existieren bereits eine ganze Reihe produktspezifischer Plugins für SQuirreL. Nichtsdestotrotz ergeht an dieser Stelle der Aufruf an Datenbankadminstratoren, die Spass an der Java Programmierung haben: Sehen Sie sich das Beispiel „Programmierung von Plugins“ an und erstellen Sie produktspezifische Plugins für SQuirreL. Spätestens wenn Administratoren mehr als ein Datenbankprodukt betreuen müssen, werden sie die Vorteile eines Clients schätzen, der Gemeinsamkeiten vereinheitlicht und Spezifika angemessen berücksichtigt.

Menschen, die den Umgang mit relationalen Datenbanken erlernen
Für jemanden, der gerade SQL lernt, können die SQL Vorlagen, die das SQL-Bookmarks Plugin per Default mitbringt, eine Hilfe sein. Darüber hinaus erlaubt das Plugin dem Benutzer, eigene Vorlagen zu definieren.

Das Syntax Plugin führt on the fly eine SQL-Syntaxprüfung durch und weist auf Fehler durch eine entsprechende Färbung (defaultmäßig rot) hin. Wird der Mauszeiger über eine als fehlerhaft ausgewiesene Stelle geführt, so wird die Fehlermeldung als Tooltip angezeigt.

Das Graph Plugin (Abbildung 7) bietet die Möglichkeit, Online aus der Datenbank heraus die relationalen Strukturen der Tabellen zu visualisieren und damit für den Lernenden zugänglich zu machen. Die Scripting Funktionen innerhalb eines Graphen erlauben es, die visuelle Darstellung wieder in DDL Befehle zu übersetzen und so die Verbindung zwischen Visualisierung und abstrakten DDL Befehlen deutlich zu machen.

Für Lernende ist es wichtig zu verstehen, dass alle relationalen Datenbanksysteme Prinzipien folgen, die unabhängig vom jeweiligen Hersteller sind. Da SQuirreL einheitlichen Zugriff auf nahezu alle relationalen Datenbanken erlaubt, hebt es diese Prinzipien besonders hervor.

Menschen, die sich in die Struktur einer vorhandenen Datenbank einarbeiten müssen
Beim Kennenlernen einer bestehenden Datenbank kann das Graph Plugin eine Hilfe sein. Es erlaubt, die Tabellen und ihre Beziehungen beliebig gruppiert zu visualisieren. Die

Graphen können vom Benutzer mit Namen versehen werden und stehen beim nächsten Öffnen einer Session wieder zur Verfügung. Darüber hinaus ist es möglich, Graphen beliebig skaliert und über mehrere Papierseiten auszudrucken. Auf diese Weise stehen dem Neuling Übersichten über die für Ihn wichtigsten Tabellengruppen jederzeit zur Verfügung.

Gerade in der Phase der Einarbeitung in eine vorhandene Datenbank fällt es oft schwer SQLs zu schreiben, weil Tabellen und Spaltennamen noch nicht geläufig sind. Hier hilft das Code Completion Plugin. Es erlaubt die Vervollständigung von SQL und DDL Code wie man es in modernen IDEs kennt. Im Vervollständigungs-Popup für Spalten werden neben dem Namen einer Spalte auch ihre wichtigsten Eigenschaften dargestellt (Abbildung 6). Werden beim Erstellen von SQL-Statements detailierte Informationen zu einer Tabelle, einem View oder einem anderen Objekt benötigt, so kann der Cursor im SQL-Editor auf das Objekt platziert werden und dann über die rechte Maustaste oder die Tastenkombination Strg+B in den Baum des Objects-Tab auf das Objekt gesprungen werden.

Programmierung von Plugins

Einleitung

Wir möchten Sie in diesen Abschnitt ermutigen, Plugins für SQuirreL zu schreiben. Da Plugins unabhängig vom SQuirreL Kernprogramm sind, können Sie weitgehend frei entscheiden, welche Funktionen Sie implementieren möchten und wie Sie Ihr Plugin designen. Trotzdem wäre es eine gute Idee, wenn Sie zuvor mit uns SQuirreL Entwicklern in Kontakt treten würden, denn es könnte sein, dass bereits andere Entwickler an denselben Funktionen arbeiten, oder dass Sie Erweiterungen der Plugin-API benötigen. Wenn Sie ein Plugin entwickeln, ist es wahrscheinlich, dass Ihr Plugin auch für andere nützlich ist. Wir möchten Sie dazu ermutigen, Ihr Plugin, wie SQuirreL selbst, unter der LGPL Lizenz zu veröffentlichen. Aber auch wenn die LGPL für sie keine Option ist, können Sie auf unsere Unterstützung rechnen.

Wir erläutern die Programmierung von Plugins an einem einfachen Beispiel, das den Quellcode von Views und Stored Procedures im SQL-Editor anzeigt. Um Änderungen an einem View oder einer Stored Procedure vorzunehmen, müssen Sie auf Basis der Anzeige die entsprechenden DDL Befehle im SQL-Editor ausführen.

Da, wie bereites erwähnt, jedes Datenbanksystem Views und Stored Procedures auf properitäre Weise verwaltet, muss es sich bei dem Plugin um ein produktspezifisches Plugin handeln. Für unser Beispiel haben wir die IBM-DB2 Datenbank gewählt. Nichtsdestotrotz kann das Beispiel als Vorlage für andere Plugins dienen.

Das komplette Beispiel ist im sql12 Modul des SQuirreL CVS Repository verfügbar. Auf www.squirrelsql.org finden Sie detailierte Informationen darüber, wie sie das Modul auschecken können. Das Plugin-Beispiel finden Sie im plugins/example/ Unterverzeichnis des sql12 Moduls.

Techniche Details

Damit SQuirreL mit einem Plugin arbeitet, müssen die compilierten Klassen des Plugins in

eine Jar-Datei gepackt werden und in das Plugins-Verzeichnis der SQuirreL Installation gelegt werden. Beim Start von SQuirreL wird die Jar-Datei geladen und die Klassen nach einer Implementierung des Interface IPlugin durchsucht. Diese Implementierung ist die zentrale Klasse eines Plugins. In unserem Beispiel heißt diese Klasse ExamplePlugin und ist von DefaultSessionPlugin abgeleitet, das seinerseits IPlugin implementiert. Die zentralen Methoden von ExamplePlugin sind initialize() und sessionStarted() (Listing 1). Ob ein Plugin von SQuirreL erkannt wurde, können sie mit Hilfe des „Plugin Summary“ Dialoges überprüfen. Den Dialog können Sie über das Menu „Plugins“ --> „Summary“ öffnen.

Die Methode initialize(), wird einmal während des Starts von SQuirreL aufgerufen. In unserer Implementierung von initialize() werden die Resourcen, das heißt die Datei example.properties, geladen, in der sich die Bezeichnungen für die Menueinträge finden, die wir dem Kontextmenu des Object-Tree hinzufügen. Die Bezeichnungen lauten „(DB2) Script View“ und „(DB2) Script Procedure“.

Die Methode sessionStarted() wird aufgerufen, wenn eine Session geöffnet wird. In unserem Beispiel prüfen wir zunächst ob es sich um eine Session für eine DB2 Datenbank handelt. Falls nicht, dann wird das Plugin für die Session nicht aktiv. Handelt es sich um eine DB2 Datenbank, dann fügen wir den View- und Procedure-Knoten des Object-Baums die besagten Kontextmenueinträge hinzu.

Wird z.B. der Kontextmenueintrag „(DB2) Script View“ angeklickt, so wird die actionPerformed() Methode der Klasse ScriptDB2ViewAction aufgerufen (Listing 2). In der actionPerformed() Methode von ScriptDB2ViewAction werden zunächst die selektierten Objekte des Object-Tree ermittelt, um diesen weiter unten die benötigten Informationen, hier der „Simple Name“, zu entnehmen. Mit dieser Information geschieht der IBM DB2 spezifische Zugriff auf die Tabelle „SYSIBM.SYSVIEWS“. Der Rest des Codes dient dazu, die View-Definitionen im SQL-Editor auszugeben.

Ganz analog geschieht in ScriptDB2ProcedureAction (hier nicht gelistet) der DB2 spezifische Zugriff auf die Stored Procedures.

Listing 1 ExamplePlugin:

```
public synchronized void initialize() throws PluginException
{
   // Intialisierung der Resourcen, die in example.properties
   // abgelegt sind. Hier die Bezeichnungen der Menueinträge.
   _resources = new PluginResources
      ("net.sourceforge.squirrel_sql.plugins.example.example", this);
}


/**
 * Wird aufgerufen, wenn eine Session gestartet wird.
 * Hier werden die Einträge dem Kontextmenu hinzugefügt.
 *
```

```
 * @param   session  Die gestartete 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"))
      {
         // Das Plugin kann Views und Stored Procedures
         // nur für DB2 Datenbanken scripten.
         // Wenn es sich also nicht um eine DB2 Session handelt,
         // dann teilen wir SQuirreL mit, das das Plugin
         // nicht benutzt werden soll.
         return null;
      }

      // Einträge den Kontextmenus der View und
      // Stored Procedure Knoten des Object-Tree hinzufügen.
      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 spezifischer Code, um View Definitionen zu
            // lesen.
            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);
         }
      }


      /**
       * Liefert den vom Benutzer konfigurierbaren Trenner für
       * Statements mit passenden Zeilenumbrüchen.
       */
      private String getStatementSeparator()
      {
         String statementSeparator =
           _session.getProperties().getSQLStatementSeparator();

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

         return statementSeparator;
      }
}
```

Fazit

SQuirreL bietet eine einfache und einheitlich Oberfläche zum Arbeiten mit allen relationalen Datenbanken. SQuirreL lehnt sich in Punkto Bedienungskomfort und Funktionalität an moderne IDEs an. Es erlaubt einfachen Zugriff auf Daten, unterstützt den Benutzer beim Verstehen von Datenstrukturen sowie beim Erlernen von und beim Umgang mit SQL. SQuirreL hebt die Dinge hervor, die alle Datenbanksysteme, unabhängig vom Hersteller, gemeinsam haben, und erlaubt es gleichzeitig, Produktspezifika mit Hilfe von Plugins angemessen zu berücksichtigen.

SQuirreL hilft Benutzern relationaler Datenbanken produktiver zu sein. Die Benutzer sind für uns SQuirreL Entwickler die beste Quelle für Ideen und Verbesserungen. Jeder, der eine Anregung geben möchte, kann uns erreichen unter: squirrel-sql-users@lists.sourceforge.net

Zuletzt ein Wort an Softwareentwickler: Die SQuirreL-Entwickler werden motiviert durch den Spass an der Programmierung und durch das Verlangen, Funktionen hinzuzufügen, die sie selbst als Benutzer von SQuirreL brauchen. Darüber hinaus ist es interessant, die

Welt der relationalen Datenbanken mit Hilfe von Java zu erforschen. Nicht zuletzt ist es ein echter Kick zu sehen, wie Funktionen, die man entwickelt hat, rund um den Globus eingesetzt werden. Mit dem SQuirreL Quellcode zu arbeiten, ist ein guter Weg zu lernen, wie man mit Java äußerst flexible Anwendungen erstellen kann. Wir möchten Sie einladen an SQuirreL mitzuarbeiten. Sie können uns erreichen unter: squirrel-sql-develop@lists.sourceforge.net