ADVENIO

SQLGrinder
User Guide

1

1. Getting Started with SQLGrinder

This chapter introduces you to SQLGrinder.

Welcome to SQLGrinder! In this chapter, you'll find out the basics of connecting SQLGrinder to your database, executing SQL statements, and browsing your database.

SQLGrinder is a database development environment for people who need to build complex queries and browse their database schemas quickly and easily. With a SQL editor that features large editing areas, SQL syntax coloring, and code completion, developing and querying your database is made simple.

SQLGrinder's database browser makes navigating database schemas as easy as browsing in the Mac Finder. Column names, index and key information, and table import and export tools are all easily accessible.

Connecting to your database

In order to interact with your database, you first must open a new SQL editor or database browser.

To open a new SQL editor:

Choose File > New > Editor (⌘N).

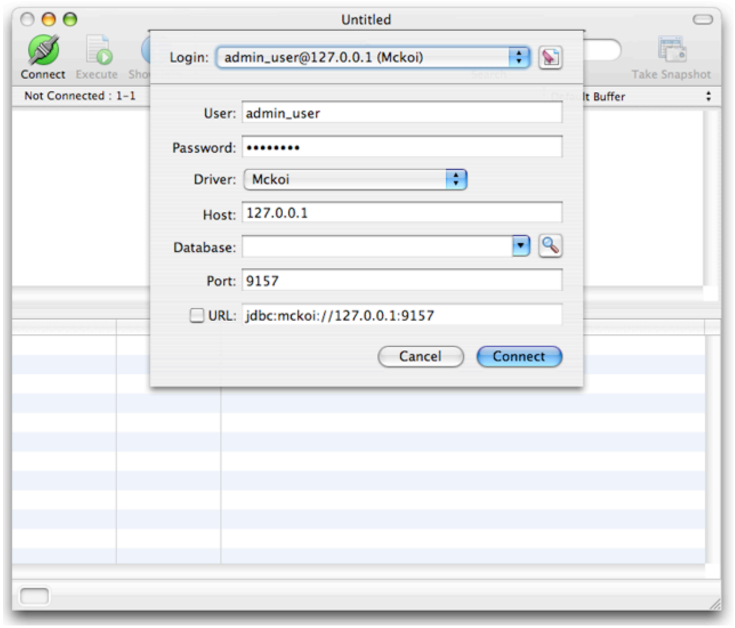
To open a database browser:

Choose File > New > Browser (⌘B).

By default, SQLGrinder will automatically open a SQL editor when the application starts, and you will be prompted for your database login information.

The information for connecting to your database in the sheet that slides down when a new window is opened, if that option is set in Preferences, or when you press the "Connect" button in the toolbar, or choose Connection > Connect (⌘K).

At the top of the sheet is the Login pop up menu with the list of your current saved log-ins. This menu will be empty the first time you run SQLGrinder. Next to the Login pop up menu is the “Add login to favorites” button, which saves the information entered in the sheet to the list of saved logins.



Login: admin_user@127.0.0.1 (Mckoi)
User: admin_user
Password: ••••••••
Driver: Mckoi
Host: 127.0.0.1
Database:
Port: 9157
URL: jdbc:mckoi://127.0.0.1:9157

To connect to your database:

1. Enter your user name into the User field.

2. Enter your password into the Password field

3. Choose the JDBC driver for your database from the Driver pop up menu. By default SQLGrinder comes with drivers for FrontBase, MySQL, OpenBase, Oracle, Post-greSQL and Sybase. If you need to connect to another database, like Microsoft SQL Server, follow the instructions in “Adding JDBC Drivers to SQLGrinder” below.

4. Enter the address or name of your database server in the Host field.

5. Enter the name of your database in the Database field. (Oracle users: see the re-lated note below.) If desired, you can look up the databases on a server by entering all of the login information except the database name, and pressing the “Lookup da-tabase names” button next to the Database field.

6. Enter the port for your server into the Port field. Default ports are listed in Appendix B.

7. Press the Connect button.

Oracle Note: Oracle's JDBC driver, by default, returns the entire database schema, which has more information than is usually desired. Because of this, the database for Oracle should be specified using SID:schema instead of just entering the SID. For example, if the database SID is DEVDB and you want to connect to the SCOTT schema, you should specify the database in the login sheet at DEVDB:SCOTT instead of just DEVDB.

Alternatively, you can press the checkbox next to the URL field to collapse the login sheet, and enter in the full JDBC URL by hand. This is useful when a database driver uses a non-standard URL that SQLGrinder doesn't support directly.

<LOGIN URL SHEET IMAGE>

If, after pressing the "Connect" button on the login sheet, you weren't able to connect to your database, you can check the Message Log for any error messages by choosing:

Tools > Show Message Log (⇧⌘M or shift-⌘-M).

Common connection problems are usually caused by incorrect port specifications, wrong user names or passwords, or attempts to connect to servers that are not running a database.

Once your editor or browser is successfully connected to your database, you can then enter a SQL query, or browse your schema.

SQL Editor Highlights

Code Completion

- To access code completion for keywords, table names, and column names, start typing something and press the Escape (esc) key. This will show the list of potential matches, based on the current word you were typing. To select something from the list, use the up and down arrow keys and press return. If there is only one match, that match will be entered into the editor for you automatically.

Searching

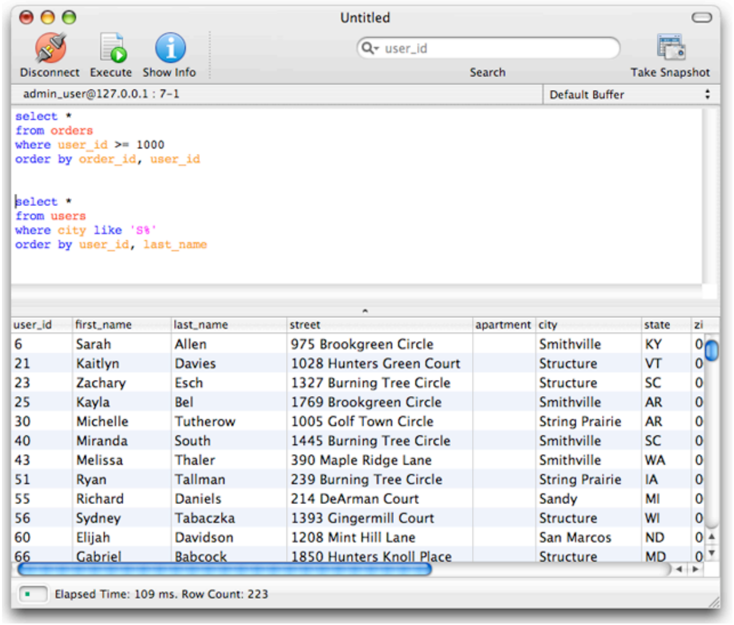
- You can search the current result set by selecting the column from the toolbar search field and typing the value you want to search for.

Sorting

- You can sort the result set by clicking on any of the result set column headers.

Exporting Information

- You can drag a selection from the result set table to another Mac application, or to the desktop to export the selection. You can also choose File > Export... to export the entire result set to a text file.



```
select *
from orders
where user_id >= 1000
order by order_id, user_id


select *
from users
where city like 'S%'
order by user_id, last_name
```

user_id	first_name	last_name	street	apartment	city	state	zi
6	Sarah	Allen	975 Brookgreen Circle		Smithville	KY	0
21	Kaitlyn	Davies	1028 Hunters Green Court		Structure	VT	0
23	Zachary	Esch	1327 Burning Tree Circle		Structure	SC	0
25	Kayla	Bel	1769 Brookgreen Circle		Smithville	AR	0
30	Michelle	Tutherow	1005 Golf Town Circle		String Prairie	AR	0
40	Miranda	South	1445 Burning Tree Circle		Smithville	SC	0
43	Melissa	Thaler	390 Maple Ridge Lane		Smithville	WA	0
51	Ryan	Tallman	239 Burning Tree Circle		String Prairie	IA	0
55	Richard	Daniels	214 DeArman Court		Sandy	MI	0
56	Sydney	Tabaczka	1393 Gingermill Court		Structure	WI	0
60	Elijah	Davidson	1208 Mint Hill Lane		San Marcos	ND	0
66	Gabriel	Babcock	1850 Hunters Knoll Place		Structure	MD	0

Elapsed Time: 109 ms. Row Count: 223

Database Browser Highlights
Browsing

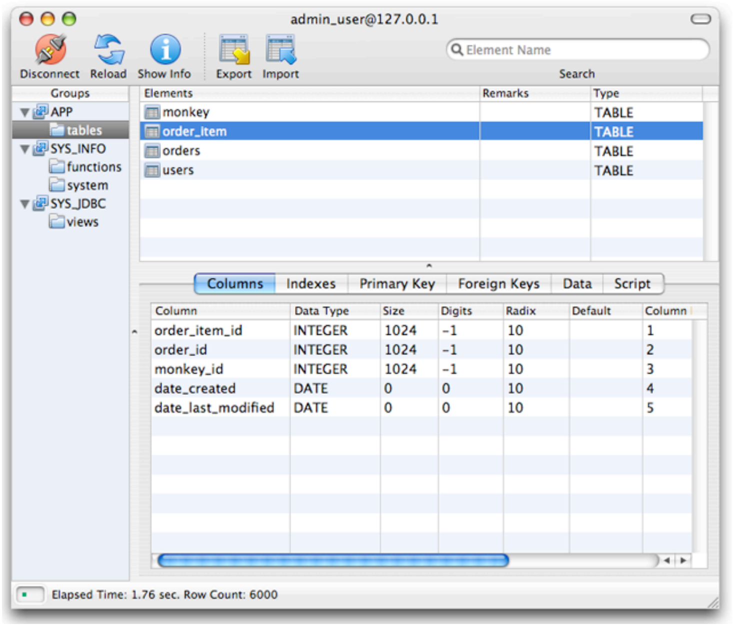
- You can browse the database structure in the list on the left. When a table is selected, the bottom panel will display a set of tabs that allows you to view the columns, indexes, columns making up the primary key, foreign keys, data and the script that can be used to create the table. When a procedure is selected, the bottom panel will display a set of tabs that allows you to view the parameters and text of the stored procedure.

Creating, dropping, and truncating tables

- This is done by selecting the table in the Elements list and choosing the appropriate menu or toolbar item.

Dropping procedures

- You can also drop a stored procedure by selecting it and choosing Browser > Drop Procedure.



Elements	Remarks	Type
monkey		TABLE
order_item		TABLE
orders		TABLE
users		TABLE

Column	Data Type	Size	Digits	Radix	Default	Column
order_item_id	INTEGER	1024	-1	10		1
order_id	INTEGER	1024	-1	10		2
monkey_id	INTEGER	1024	-1	10		3
date_created	DATE	0	0	10		4
date_last_modified	DATE	0	0	10		5

Elapsed Time: 1.76 sec. Row Count: 6000

Adding JDBC Drivers to SQLGrinder

For most common JDBC drivers that aren't already included, not much is required to add it them to SQLGrinder. SQLGrinder comes pre-configured with the information for loading the most commonly used drivers, so all you have to do is take the jar or zip file that contains the driver and put it in the

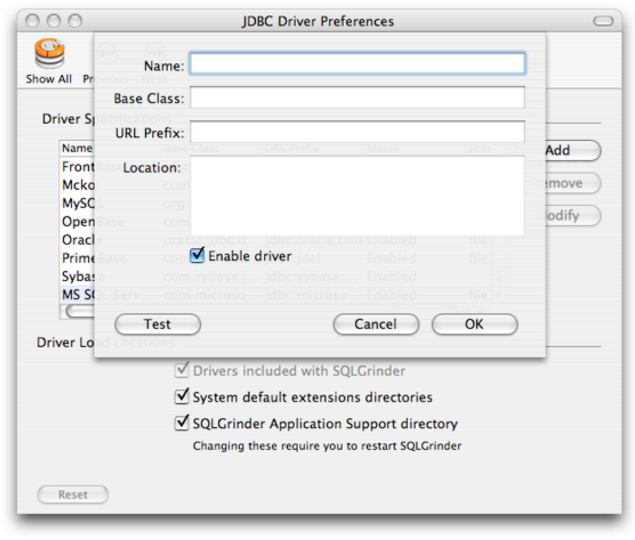
Library/Application Support/SQLGrinder

folder in your user folder. When you restart SQLGrinder, you should find your driver listed in the Driver pop up menu in the login sheet. Alternatively you can also place your driver in one of the following locations:

- User home: ~/Library/Java/Extensions
- Local domain: /Library/Java/Extensions
- Network domain: /Network/Library/Java/Extensions
- System domain: /System/Library/Java/Extensions

Adding a JDBC driver that SQLGrinder doesn't know about

From time to time you may need to use a JDBC driver that SQLGrinder isn't already configured to use. Adding this knowledge to the application requires adding a new entry in the JDBC preferences pane.


JDBC Driver Preferences
Show All
Name:
Base Class:
URL Prefix:
Location:
Enable driver
Test
Cancel
OK
Add
Remove
Modify
Driver Specifications
Driver Load Locations
Drivers included with SQLGrinder
System default extensions directories
SQLGrinder Application Support directory
Changing these require you to restart SQLGrinder
Reset

While SQLGrinder should work with any driver that supports at least JDBC 2.0, the built in driver configuration database doesn't know about every possible driver available. New drivers require that the configuration information be specified manually after adding the new driver file to one of the driver locations listed above.

Note: You will usually not be required to add a JDBC configuration yourself. SQLGrinder comes configured to load most common JDBC drivers. You only need to add a configuration yourself if you put the driver in one of the specified locations and it's not found and loaded.

To add a new driver:

1. After adding a new driver to one of the listed locations, launch SQLGrinder, if it's not running, and open Preferences by choosing SQLGrinder > Preferences. Click on the "JDBC Drivers" icon found in the Application section of the Preferences window.

2. Press the Add button to display the sheet shown above, and enter the values for name, base class, and url prefix. The Name is the label you want to to appear in the driver popup menu. The Base Class is the name of the main JDBC driver class. The URL Prefix is the first part of the JDBC url as specified by the driver developer. You can usually find this information in the developer documentation accompanying the driver.

3. After entering the information, you can try loading the driver by pressing the "Test" button. This will try and load the driver, and if it was configured correctly, display the location in the "Location" text area.

As an example, these are the entries for the Oracle 9i JDBC driver:

- Name: Oracle 9i
- Base class: oracle.jdbc.OracleDriver
- URL: jdbc:oracle:thin

For a full list of currently supported JDBC drivers, see Appendix A.

Shared Toolbar Tools

The following toolbar tools are common to both the SQL editor and the database browser. They can be added or removed by choosing View > Customize Toolbar... Choosing this menu item presents a sheet that allows tools to be dragged onto the toolbar to add them or dragged off to remove them.

Commit	Commits the current open transaction.
Connect	Displays the login sheet so that the login information can be entered.
Disconnect	Closes the current open connection.
Export	Exports the current result set or selected table to a file.
New Browser	Creates a new browser instance, connecting it to the same database as the parent.
New Editor	Creates a new editor instance, connecting it to the same database as the parent.
Rollback	Undoes any changes for the current open transaction.
Settings	Displays the settings for the current open connection.
Show Info	Displays the JDBC information for the current open database connection.
Show Library	Displays the SQL Statement Library window.
Show Message Log	Displays the Message Log window.

🔍	Search	Allows columns of a result set or table to be searched, showing all rows or elements that match.
🛑	Stop	Stops the current running process.

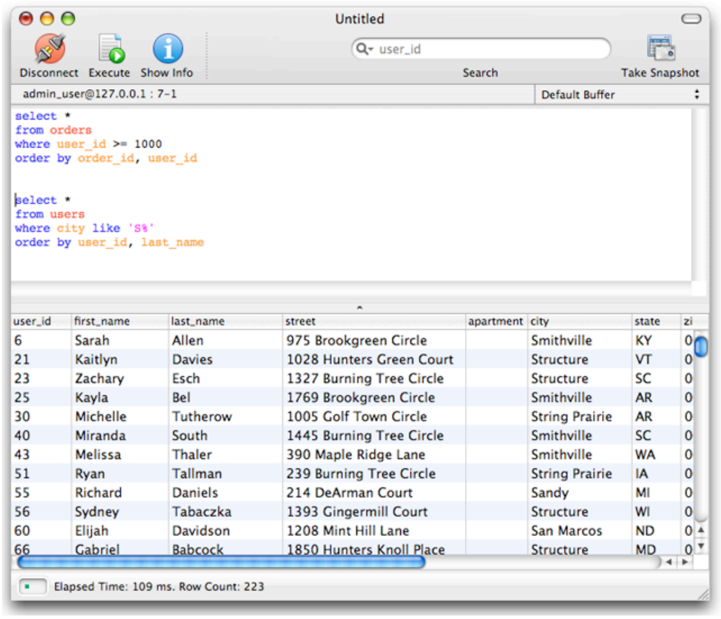
2

2. The SQL Editor

This chapter describes the features and usage of the SQL editor.

The SQLGrinder Editor is used to create, edit, save, and open your SQL queries. Editors can be saved and opened as ASCII text or as SQLGrinder documents that also save your tabs, syntax coloring etc. Each document can include one or many queries, and queries can either be sent to your database one at a time, or as a series of semicolon separated commands.

After executing the text of a query, the results are displayed in a table at the bottom of the editor. Once the table is populated, the data can be saved to a text file. A “snapshot” of the result set table contents can also be viewed in a table in another window by clicking the Snapshot button.



```
select *
from orders
where user_id >= 1000
order by order_id, user_id


select *
from users
where city like 'S%'
order by user_id, last_name
```

user_id	first_name	last_name	street	apartment	city	state	zi
6	Sarah	Allen	975 Brookgreen Circle		Smithville	KY	0
21	Kaitlyn	Davies	1028 Hunters Green Court		Structure	VT	0
23	Zachary	Esch	1327 Burning Tree Circle		Structure	SC	0
25	Kayla	Bel	1769 Brookgreen Circle		Smithville	AR	0
30	Michelle	Tutherow	1005 Golf Town Circle		String Prairie	AR	0
40	Miranda	South	1445 Burning Tree Circle		Smithville	SC	0
43	Melissa	Thaler	390 Maple Ridge Lane		Smithville	WA	0
51	Ryan	Tallman	239 Burning Tree Circle		String Prairie	IA	0
55	Richard	Daniels	214 DeArman Court		Sandy	MI	0
56	Sydney	Tabaczka	1393 Gingermill Court		Structure	WI	0
60	Elijah	Davidson	1208 Mint Hill Lane		San Marcos	ND	0
66	Gabriel	Babcock	1850 Hunters Knoll Place		Structure	MD	0

```
Elapsed Time: 109 ms. Row Count: 223
```

Connecting a SQL Editor to your Database

To connect to a database, choose File > New Editor (⌘N). This will open a new SQL Editor.

Alternatively, a new connection to a database can be created using the connection information from an already open connection. To do this, choose File > New > Editor Using Current (⇧⌘N or shift-⌘-N) or File > New > Browser Using Current (⇧⌘B or shift-⌘-B) commands. A tool can also be added to the toolbar that will do this as well.

Once connected to a database, an editor or database browser will show a green status indicator in leftmost position of the status indicator panel. This indicates that there is currently an open and functioning database connection.

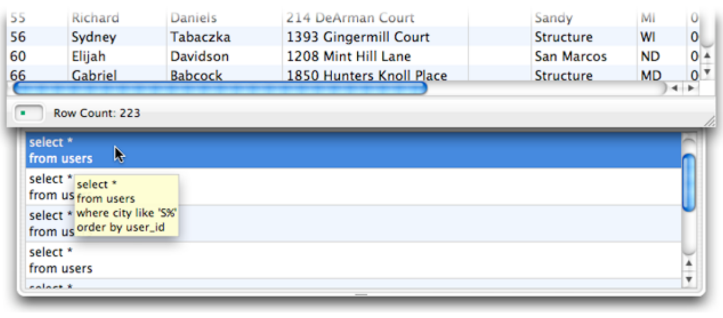
SQL Editor Toolbar Icons

The following tools are available on the SQL Editor toolbar. They can be added or removed by choosing View > Customize Toolbar... Choosing this menu item presents a sheet that allows tools to be dragged onto the toolbar to add them or dragged off to remove them.

Execute	Executes the contents of the editor, sending each semi-colon delimited statement to the database one at a time.
Execute Script	Executes the entire contents of the editor buffer as one script.
Take Snapshot	Creates a copy of the current result set, and displays it in another window.

Executing and Editing SQL Statements

There are many different ways to execute statements from the SQL Editor. When SQL code is sent to your database, semi-colon separated commands are sent one at a time, and each result set is added to the list of result sets in the editor Result Set Drawer. To see your results, show the drawer by choosing View > Show Result Set Drawer (⌥⌘R or option-⌘-R). Result sets are listed from most recent to least recent.



```
55   Richard   Daniels     214 DeArman Court          Sandy        MI   0
56   Sydney    Tabaczka    1393 Gingermill Court      Structure    WI   0
60   Elijah    Davidson    1208 Mint Hill Lane        San Marcos   ND   0
66   Gabriel   Babcock     1850 Hunters Knoll Place   Structure    MD   0
```

Row Count: 223

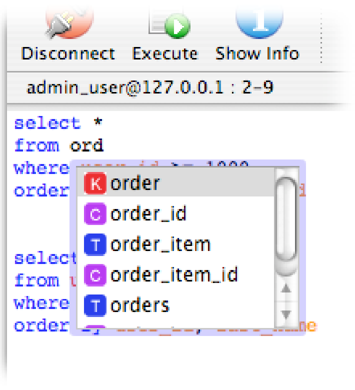
```
select *
from users

select *
from us

select *
from us

select *
from users
```

```
select *
from users
where city like 'S%'
order by user_id
```

Code completion
Keywords and table, column and procedure names are available while typing by pressing the escape key at any time while editing your SQL code. If there is more than one match, a window is displayed with your choices. Each type of match is marked with an image designating the type of command it is.

K	Database or SQL-99 keyword
C	Table column name
T	Table name
P	Procedure name

The choices can be narrowed down when the completion window is visible, by pressing the control (^) key to limit matches to table names, the command (⌥) key to limit matches to column names, and the shift (⇧) key to limit matches to keywords. While the completion window is open, you can also navigate the list of matches by using the up and down arrow keys, and you can select the string you want by pressing return. If there are no matches, you will hear your system alert sound when you press escape.



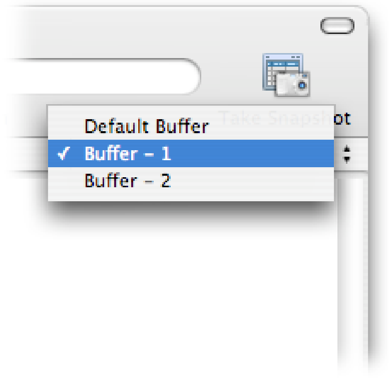
admin_user@127.0.0.1 : 2-9

```
select *
from ord
where
order


select
from
where
order
```

```
K order
C order_id
T order_item
C order_item_id
T orders
```

Finally, the code completion window can be dismissed by pressing the escape key a second time.

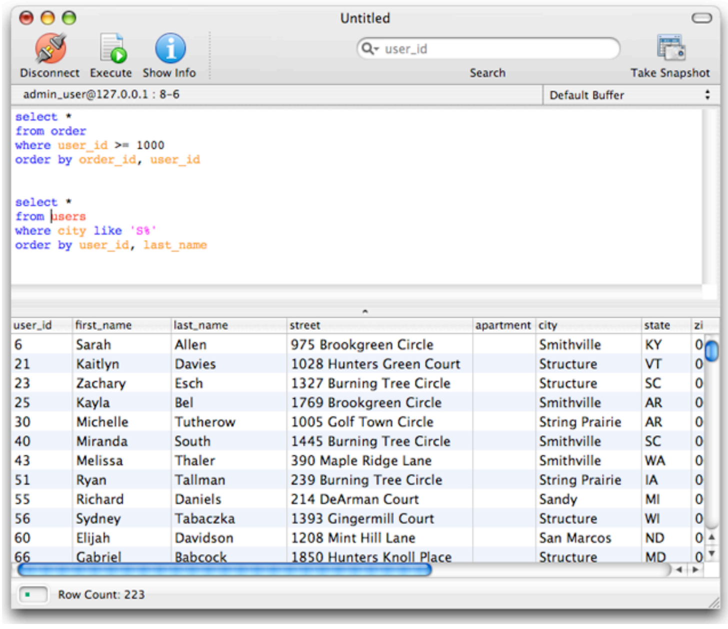
Editor buffers
Each editor allows you to work in one or many workspaces, or buffers. Each buffer has it's own result set list, and code space. This means that you can easily work on multiple blocks of SQL code in the same editor window. To create a new buffer, choose View > Tabs > New (⌘T). Switching from one buffer to another is done by choosing View > Tabs > Next (⌥⌘→ or option-⌘-right arrow) and View > Tabs > Previous (⌥⌘← or option-⌘-left arrow). Additionally you can close, rename and merge tabs.



Sending the editor contents to the database
To execute the entire contents of a SQL Editor as a query, enter the text into the editor text view and click the Execute toolbar button or choose Editor > Execute Statements (⌘-E). Doing any of these actions will send the entire contents of the editor window to the database to be executed. Additionally, pressing the Enter key will send the entire contents of the buffer to the database.

Sending a selection to the database
Selections in the editor window can be executed by selecting the text of a SQL statement and pressing the Execute toolbar button, or by choosing Editor > Execute Statements (⌘-E). Doing any of these with text selected will only send the text that is selected to the database.

Sending a block of code to the database
The current white space delimited block of text that contains the insertion point can also be sent to the database by choosing Editor > Execute Block (⌘⌅ or ⌘-enter). In the image below, the cursor is currently located in the second block of text. Using Execute Block now will send just the text of that block to the database. This makes it easy to work with many smaller code blocks in one editor without having to have multiple editor windows or buffers open.



```
select *
from order
where user_id >= 1000
order by order_id, user_id


select *
from users
where city like 'S%'
order by user_id, last_name
```

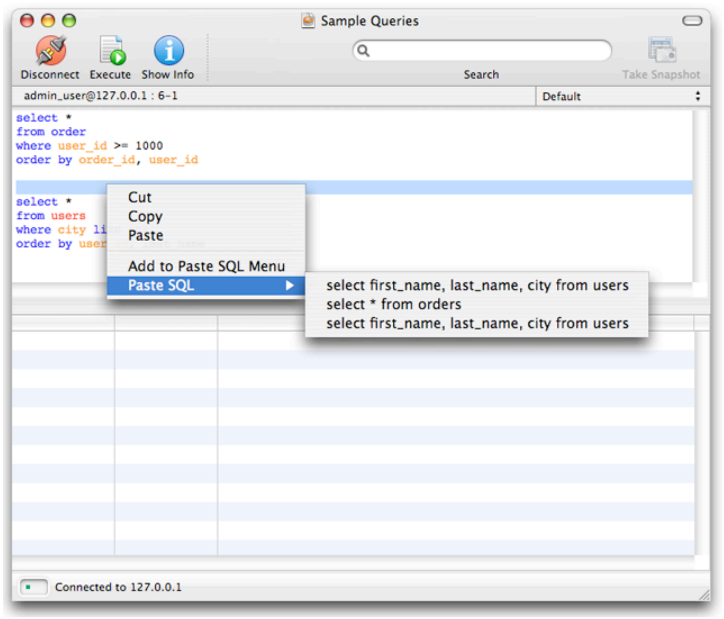
user_id	first_name	last_name	street	apartment	city	state	zi
6	Sarah	Allen	975 Brookgreen Circle		Smithville	KY	0
21	Kaitlyn	Davies	1028 Hunters Green Court		Structure	VT	0
23	Zachary	Esch	1327 Burning Tree Circle		Structure	SC	0
25	Kayla	Bel	1769 Brookgreen Circle		Smithville	AR	0
30	Michelle	Tutherow	1005 Golf Town Circle		String Prairie	AR	0
40	Miranda	South	1445 Burning Tree Circle		Smithville	SC	0
43	Melissa	Thaler	390 Maple Ridge Lane		Smithville	WA	0
51	Ryan	Tallman	239 Burning Tree Circle		String Prairie	IA	0
55	Richard	Daniels	214 DeArman Court		Sandy	MI	0
56	Sydney	Tabaczka	1393 Gingermill Court		Structure	WI	0
60	Elijah	Davidson	1208 Mint Hill Lane		San Marcos	ND	0
66	Gabriel	Babcock	1850 Hunters Knoll Place		Structure	MD	0

Row Count: 223

Building procedures and triggers - sending editor code as a script
Normally, sending the entire contents of an editor to the database will result in semi-colon delimited statements being split into separate commands and executed one at a time. When developing stored procedures and triggers however, this is not the desired behavior. Building these requires that the entire editor buffer get sent as one command. Accomplishing this is done by choosing Editor > Execute as Script (⌥⌘P or option-⌘-P) editor.

Displaying result set binary data
When your editor result set includes columns that have binary data, you can view this data using the Binary Data Drawer, which can be opened by choosing View > Show Binary Data Drawer (⌥⌘B or option-⌘-B). When this view is visible, you can display your binary data by selecting a row and choosing the column that you want to display from the Column pop up menu at the bottom of the view. If your data is an image, select the Image tab. If it is text, select the Text tab and then select the proper text encoding from the Encoding pop up menu.

The Paste SQL menu
The editor features a configurable menu that allows you to quickly paste often used SQL statements into the text view. To access this menu, right-click (or ctrl-click) anywhere in the editor text area. To paste something, choose one of the SQL statements from the menu or submenu.



```
select *
from order
where user_id >= 1000
order by order_id, user_id

select *
from users
where city li
order by user
```

To add a new statement to the Paste SQL menu:

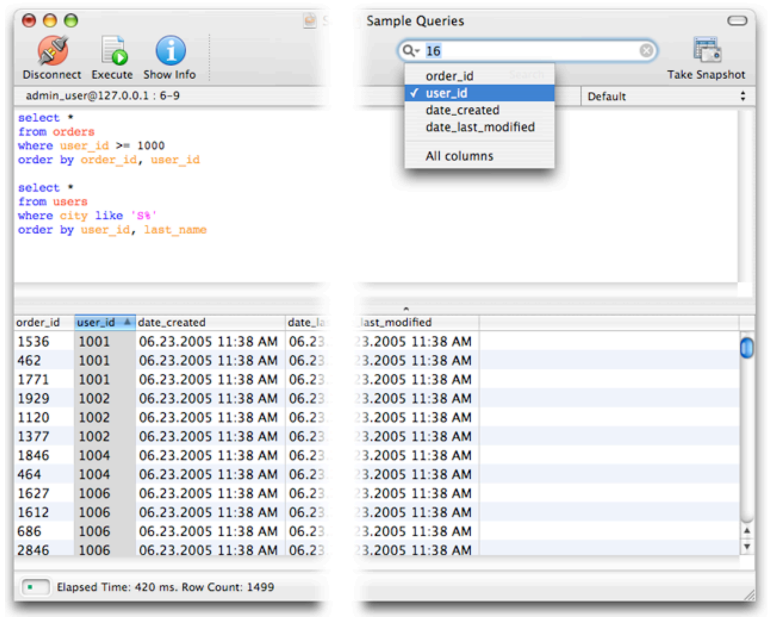
1. Select the Menu group in the SQL Statement Library.
2. Create a new statement in this group, or in a sub-group. New statements can be added to the group directly, or dragged in from other groups. Text can also be dragged to the group from an editor.
3. To set the title of the menu item, make the first line of the statement a comment such as: `-- Select all customers` and the menu item will have this as a title (minus the leading comment characters).

The following can also be done to add a menu item to this list:

1. Select text in the editor.
2. Right-click (or ctrl-click) in the editor to display the contextual menu.
3. Choose Add to Paste SQL Menu, and the text will be added to a new submenu, if necessary, with the name of the current login.

Searching and Sorting Editor Results

Each SQL editor features client-side searching and sorting. Once a result set has been loaded, searching is as simple as choosing a table column to search on in the toolbar Search field, and typing the string you want to search for. You can also choose "All columns" and search for a string across all of the columns in the result set, rather than limiting your search to one.



admin_user@127.0.0.1 : 6-9

```
select *
from orders
where user_id >= 1000
order by order_id, user_id

select *
from users
where city like 'S%'
order by user_id, last_name
```

order_id	user_id ▲	date_created	date_la	ast_modified
1536	1001	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
462	1001	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1771	1001	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1929	1002	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1120	1002	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1377	1002	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1846	1004	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
464	1004	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1627	1006	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
1612	1006	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
686	1006	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM
2846	1006	06.23.2005 11:38 AM	06.23	3.2005 11:38 AM

Elapsed Time: 420 ms. Row Count: 1499

To sort your result set, click on the table header of the column you want to sort, and the result set will be resorted, without having to communicate with your database.

Transactions: Committing or Rolling Back an Update

SQL statements can be selectively committed or rolled back. If the database an editor is connected to allows transactions, and the Preferences > Connections > Commit after every statement checkbox is checked, the Rollback and Commit buttons and the related menu items will be enabled. When an insert, update, delete or create is executed and a transaction is opened, a blue status indicator is shown in the center position of the status indicator panel. If an open transaction is closed, the open transaction indicator is then cleared. Note that the transaction behavior can also be set on a per-connection basis by setting it in the settings for an open window, by choosing Connection > Show Settings (⇧⌘K or shift-⌘-K).

Exporting Result Set Table Data

When a result set table is populated with data from a query, the data can be saved to a text file by pressing the Export toolbar button, or choosing File > Export...

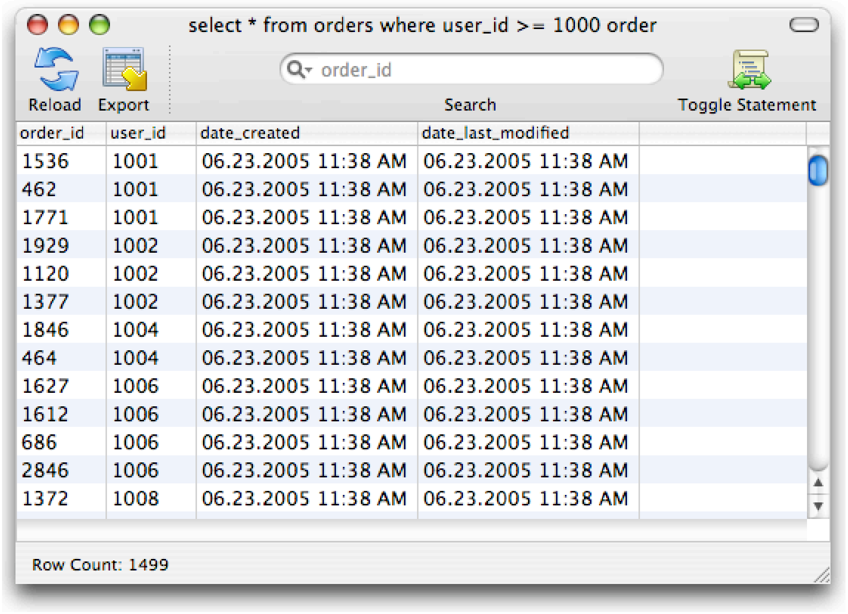
The results of a query can also be saved directly to a file by choosing Edit > Execute to File. If more than one semi-colon separated statement is sent to the database, a prompt for each result set will be displayed, so that the name and location of the export file can be entered.

Alternatively, you can select rows in the result set and either drag them to the desktop, or to another application. The data will be exported as text, using the column delimiter value of the “Copy column delimiter” setting in Import/Export preferences. Note that this method is best used for exporting small subsets of data, rather than the entire result set.

Taking a Snapshot of a Result Set

There are times when it is desirable to temporarily save the data of a result set “off to the side” of an editor window. The snapshot feature accomplishes this. If the SQL Editor has a current result set, the data in this result set can be copied to a temporary snapshot window. This is done by pressing the Snapshot button in the Toolbar or by choosing the Editor > Snapshot menu item (⇧⌘T or shift-⌘-T).

What good is a snapshot window? It’s a useful tool when you want to have many result sets open, say for database types or other related data, but don’t want to use whole editors and their associated database connections, to do it.



order_id	user_id	date_created	date_last_modified
1536	1001	06.23.2005 11:38 AM	06.23.2005 11:38 AM
462	1001	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1771	1001	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1929	1002	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1120	1002	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1377	1002	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1846	1004	06.23.2005 11:38 AM	06.23.2005 11:38 AM
464	1004	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1627	1006	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1612	1006	06.23.2005 11:38 AM	06.23.2005 11:38 AM
686	1006	06.23.2005 11:38 AM	06.23.2005 11:38 AM
2846	1006	06.23.2005 11:38 AM	06.23.2005 11:38 AM
1372	1008	06.23.2005 11:38 AM	06.23.2005 11:38 AM

Row Count: 1499

Snapshots queries can be displayed, refreshed and searched as well.

Calling Stored Procedures and Functions

When connected to Oracle, Sybase or Microsoft SQL Server, the SQLGrinder editor allows functions and stored procedures to be called by using the editor 'call' command:

Procedures

```
     call procedure_name(var1, var2, ... varN)
```

Functions

```
     call function_name(var1, var2, ... varN)
```

or

```
     call function_name var1, var2, ... varN
```

The previous descriptions are examples of procedures that return either nothing, or one return value.

You can, however also call procedures that have out or in/out parameters as in the following example which shows a simple Sybase stored procedure, with one return value, one int IN parameter, and one string IN/OUT parameter...

```
create procedure sp_callableSample(
    @p1 int, @p2 varchar(255) out) as
begin
    select @p1, @p2
    select @p2 = 'Monkey'
    return 42
end
```

calling this in the editor using the following 'call' command...

```
call sp_callableSample(20, 'testing')
```

results in three result sets being returned and added to the Result Set Drawer. The first result set, labeled "@p2" in the list of result sets, contains the updated p2 parameter value passed in. The second result set, labeled "Function return value" in the list of result sets, contains the return value, which in this example case is 42. Finally the third result set, labeled "Returned result set 0" contains the results of the first select statement.

Describing Database Tables and Procedures

The SQLGrinder editor allows the display of table column and procedure parameter information using the 'desc' command:

```
desc procedure storename_proc
```

will display a result set containing the parameters for `storename_proc`. This command:

```
desc table authors
```

will display the column information for the `authors` table.

Other Editor Features: Formatting and Reloading Keywords

Rounding out the editor features are commands to uppercase and lowercase selected text, and to shift text left and right. To apply any of these commands, just select the text that you want to change, and then choose the desired Editor menu command.

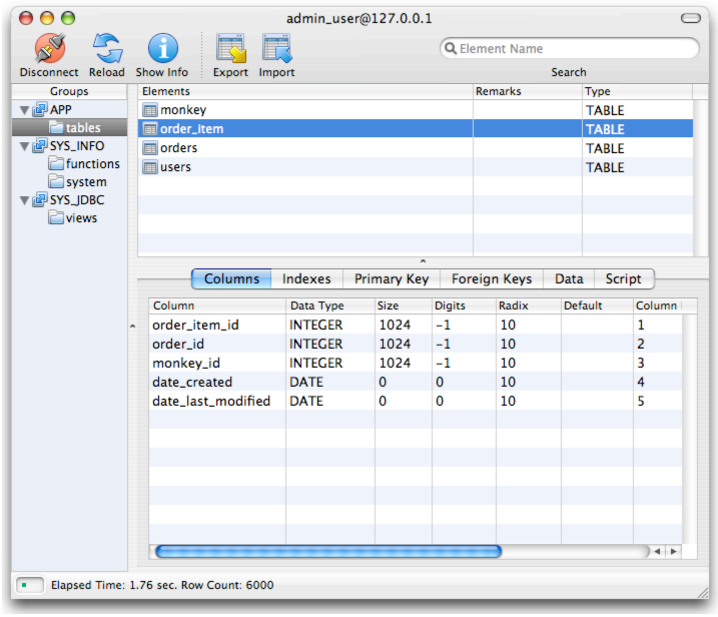
- To uppercase the selected text choose Editor > Make Uppercase (⌘+).
- To lowercase the selected text, choose Editor > Make Lowercase (⌘-).
- To shift the selected text left, choose Editor > Shift Left (⌘[).
- To shift the selected text right, choose Editor > Shift Right (⌘]).

3

3. The Database Browser

This chapter describes the features and usage of the database browser.

The SQLGrinder database browser allows easy navigation of a database schema's elements, such as tables, procedures, triggers and sequences. When a browser is connected to a database, the database structure is parsed to generate a browser representation. This representation is based on the JDBC schema of top level catalogs that include schemas, tables and procedures, and schemas that include tables and procedures.



Elements	Remarks	Type
monkey		TABLE
order_item		TABLE
orders		TABLE
users		TABLE

Column	Data Type	Size	Digits	Radix	Default	Column
order_item_id	INTEGER	1024	-1	10		1
order_id	INTEGER	1024	-1	10		2
monkey_id	INTEGER	1024	-1	10		3
date_created	DATE	0	0	10		4
date_last_modified	DATE	0	0	10		5

Elapsed Time: 1.76 sec. Row Count: 6000

Connecting a Database Browser to your Database

To connect to a database, choose File > New Browser (⌘N). This will open a new Database Browser.

Alternatively, a new connection to a database can be created using the connection information from an already open connection. To do this, choose File > New > Editor Using Current (⇧⌘N or shift-⌘-N) or File > New > Browser Using Current (⇧⌘B or shift-⌘-B) commands. A tool can also be added to the toolbar that will do this as well.

Once connected to a database, an editor or database browser will show a green status indicator in leftmost position of the status indicator panel. This indicates that there is currently an open and functioning database connection.

Database Browser Icons

The following icons represent nodes and elements in your database schema.

- Catalog
- Group
- Table
- Schema
- Procedure, trigger, sequence

Database Browser Toolbar Icons

The following tools are available on the database browser toolbar. They can be added or removed by choosing View > Customize Toolbar... Choosing this menu item presents a sheet that allows tools to be dragged onto the toolbar to add them or dragged off to remove them.

Drop Table	Drop the selected table from the schema.
Edit Table	Alter the selected table. (Not implemented)
Import	Import data into the selected table.
New Table	Add a new table to the schema.

	Reload	Reload the schema, or the selected schema element.
	Truncate Table	Truncate the selected table, removing all table rows.

Navigating and Changing Your Database Schema

After the structure is parsed, the database components can be browsed by clicking on a branch in the browser "Groups" tree on the left.

Clicking on an element in the browser that has no children, such as a table, will display a set of tabs for that element. Once a tab is displayed, clicking on any other database object in that column will update the data in the tab automatically, making it easy to browse the columns of a list of tables, for example. Additionally, if you double-click on a browser element, any associated script for the element will be opened in a SQL editor.

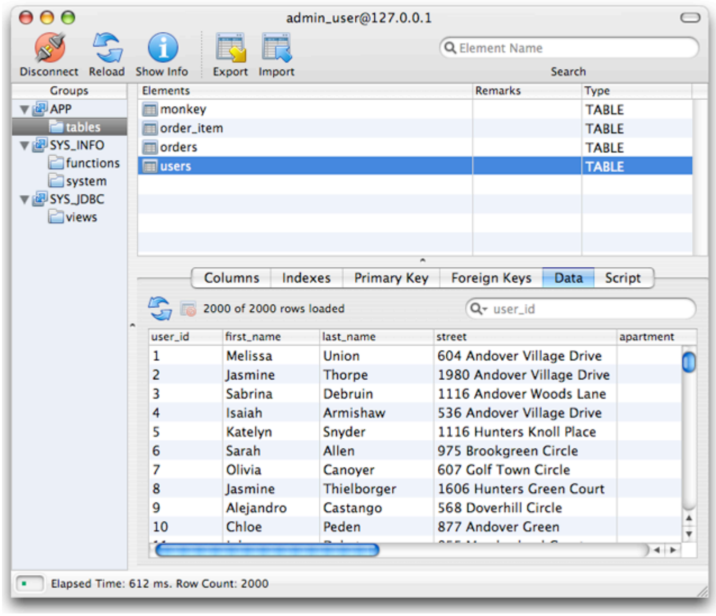
Table Meta Data Information

The browser allows you to easily view the meta data, such as column and primary key information, for a table.

The Columns tab, when visible, displays the columns and associated information for a table. The Indexes, Primary Key and Foreign Keys tabs are very similar to the Columns tab.

Displaying and Editing the Data in a Table

The Data tab displays the data in the table. To sort a column of data, click on the column header. One click on the column header will sort the table data by that column in ascending order. The next click will sort the data by the clicked column in descending order.



Elements	Remarks	Type
monkey		TABLE
order_item		TABLE
orders		TABLE
users		TABLE

2000 of 2000 rows loaded

user_id	first_name	last_name	street	apartment
1	Melissa	Union	604 Andover Village Drive	
2	Jasmine	Thorpe	1980 Andover Village Drive	
3	Sabrina	Debruin	1116 Andover Woods Lane	
4	Isaiah	Armishaw	536 Andover Village Drive	
5	Katelyn	Snyder	1116 Hunters Knoll Place	
6	Sarah	Allen	975 Brookgreen Circle	
7	Olivia	Canoyer	607 Golf Town Circle	
8	Jasmine	Thielborger	1606 Hunters Green Court	
9	Alejandro	Castango	568 Doverhill Circle	
10	Chloe	Peden	877 Andover Green	

Elapsed Time: 612 ms. Row Count: 2000

The data in the selected table can be edited if it has at least one primary key. When a table's data can be changed, an icon with a pencil is displayed next to the Load or Refresh button. To edit data, double-click on a cell, change the value and press "return" or "enter." Commit the transaction, to save any changes, if an transaction is open.

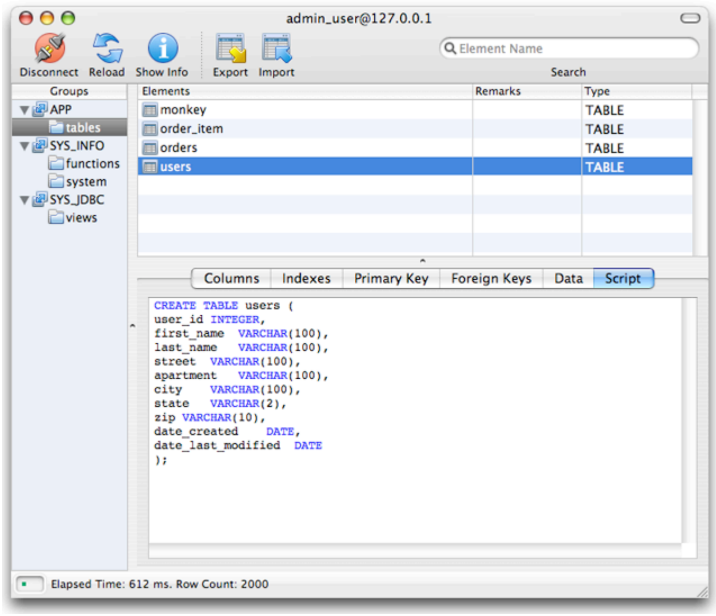
Like result sets in the editor, the data table can be searched by selecting a column in the Search field and typing a string.

To enter in a null value, or an empty string, clear the cell and press the "enter" key. This will display a prompt asking for the value to be inserted into the table.

Tables with no primary key are not able to be changed because there can be no guarantee that JUST the desired row will get updated.

Displaying the Script for a Table

The Script tab displays a script that can be used to describe and create the selected table.



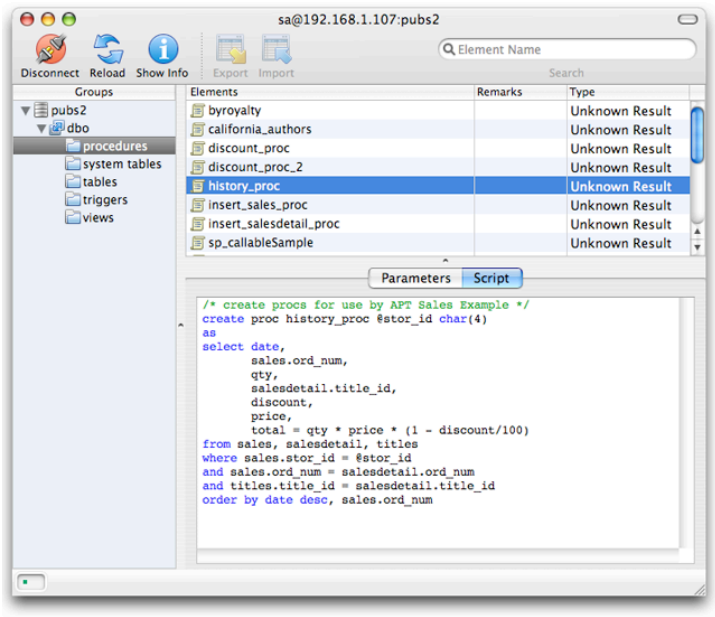
Elements	Remarks	Type
monkey		TABLE
order_item		TABLE
orders		TABLE
users		TABLE

```
CREATE TABLE users (
user_id INTEGER,
first_name  VARCHAR(100),
last_name   VARCHAR(100),
street  VARCHAR(100),
apartment   VARCHAR(100),
city    VARCHAR(100),
state   VARCHAR(2),
zip VARCHAR(10),
date_created    DATE,
date_last_modified  DATE
);
```

Elapsed Time: 612 ms. Row Count: 2000

Procedure Meta Data Information and Code

When a procedure is selected in the browser, you are presented options of viewing both the parameter information, and for Oracle, Sybase and Microsoft SQL Server, the actual source code for the procedure.



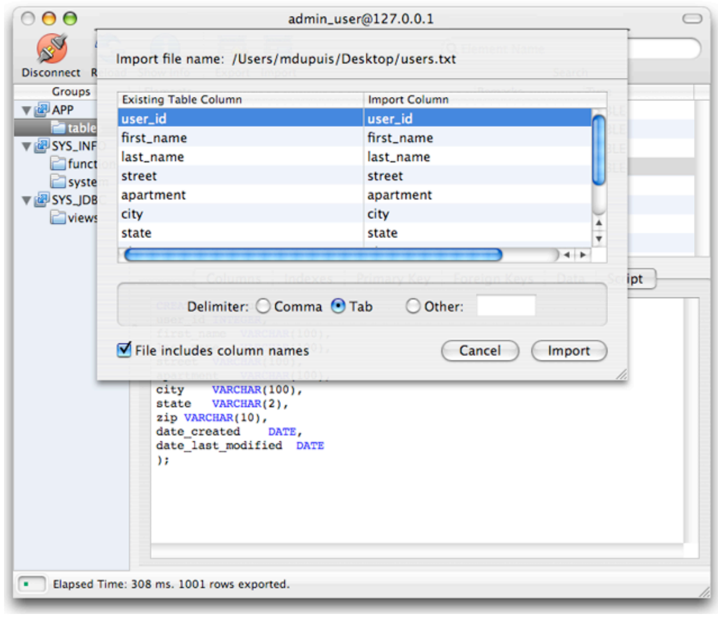
Elements	Remarks	Type
byroyalty		Unknown Result
california_authors		Unknown Result
discount_proc		Unknown Result
discount_proc_2		Unknown Result
history_proc		Unknown Result
insert_sales_proc		Unknown Result
insert_salesdetail_proc		Unknown Result
sp_callableSample		Unknown Result

```
/* create procs for use by APT Sales Example */
create proc history_proc @stor_id char(4)
as
select date,
       sales.ord_num,
       qty,
       salesdetail.title_id,
       discount,
       price,
       total = qty * price * (1 - discount/100)
from sales, salesdetail, titles
where sales.stor_id = @stor_id
and sales.ord_num = salesdetail.ord_num
and titles.title_id = salesdetail.title_id
order by date desc, sales.ord_num
```

Importing Data Into a Table

Data can be imported into a selected table by pressing the Import toolbar button or choosing File > Import... When the command is issued, you are first prompted for the file to import.

Next, a sheet is displayed with various import options. Here you must set whether or not the file has column names as the first row, and the delimiter that was used in the file. You also must make sure that the existing table columns match up with the import columns in the file.


admin_user@127.0.0.1

Import file name: /Users/mdupuis/Desktop/users.txt

Existing Table Column	Import Column
user_id	user_id
first_name	first_name
last_name	last_name
street	street
apartment	apartment
city	city
state	state

Delimiter: ( ) Comma [✓ Tab] ( ) Other:

[✓] File includes column names

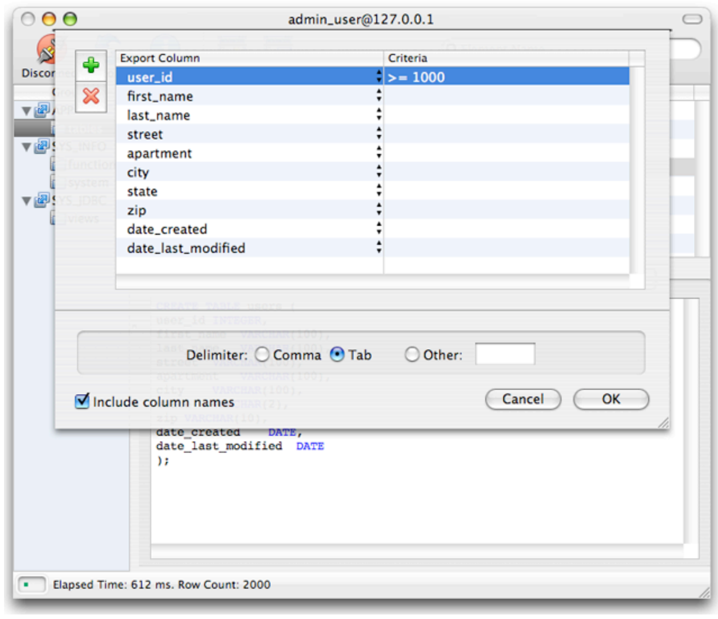
```
city      VARCHAR(100),
state     VARCHAR(2),
zip VARCHAR(10),
date_created      DATE,
date_last_modified  DATE
);
```

Elapsed Time: 308 ms. 1001 rows exported.

Once the information has been specified correctly, press the Import button to start the import.

Exporting Data From a Table

Data can be exported from the current table by pressing the Export toolbar button or choosing File > Export... When the command is issued, you are first prompted for the columns to export.


admin_user@127.0.0.1

Export Column	Criteria
user_id	>= 1000
first_name	
last_name	
street	
apartment	
city	
state	
zip	
date_created	
date_last_modified	

Delimiter: ( ) Comma [✓ Tab] ( ) Other:

[✓] Include column names

```
date_created      DATE,
date_last_modified  DATE
);
```

Elapsed Time: 612 ms. Row Count: 2000

To export data, select the desired columns for export from the pop up menus in the first column of the export sheet. If desired, you can add and remove columns from your export set by pressing the Add or Remove buttons to the left of the table.

Simple criteria can also be specified for your export. To specify a restriction on an export column, enter the restriction in the table column labeled “Criteria”. The following are examples of the kinds of export restrictions that can be entered:

< 100	Export only rows where the integer column values are less than 100
>= 1000	Export rows where the integer column values are greater than or equal to 1000
= 'new'	Export rows where the column values are equal to 'new'
like 'Fred%'	Export rows where the string column values are similar to any value that starts with the substring 'Fred'

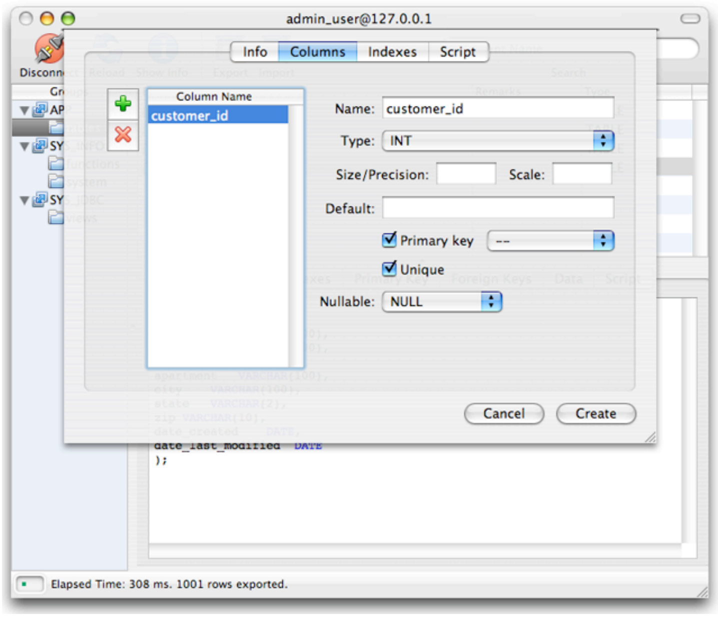
All of the common operators can be used as a restriction, such as <, >, <=, >=, !=, =, like, etc.

Additionally, you can specify the delimiter to get used between columns, and whether or not to include the column names as the first data row.

Once the columns have been selected, and any restrictions specified, press OK to begin the export.

The Table Builder

The browser includes an assistant, called the Table Builder, that makes it easy to create new database tables.



customer_id

Name: customer_id

Type: INT

Size/Precision: Scale:

Default:

[✓] Primary key --

[✓] Unique

Nullable: NULL

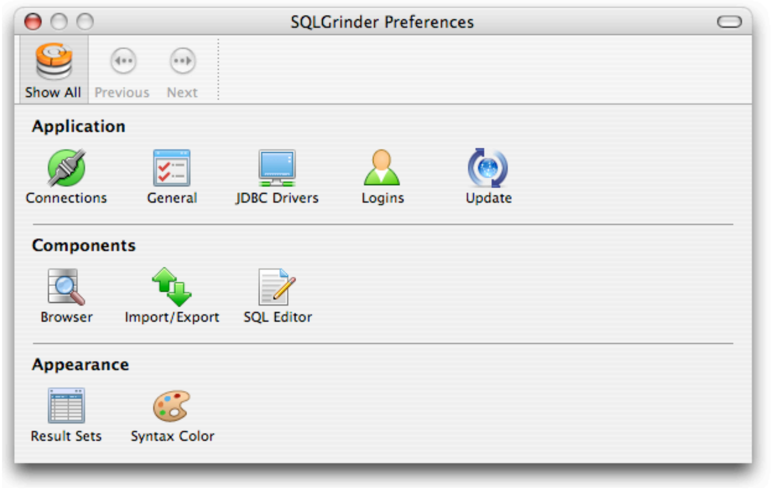
Elapsed Time: 308 ms. 1001 rows exported.

In the table builder, the table name, columns and indexes can be specified to create a new table. After specifying the name for the table, columns can be added and deleted, and the meta data entered, on the “Columns” tab. Indexes can be specified on the Indexes tab, after columns have been added, by selecting the columns to be indexed. Finally the Script tab displays the script that will be used to create the new database table.

4

4. Setting SQLGrinder Preferences

This chapter discusses preferences and explains how to change the default preference settings.



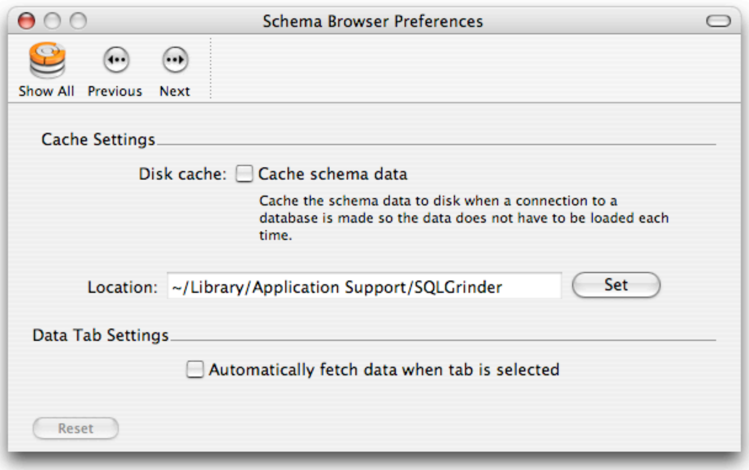
Preferences Overview

The following is an overview of the SQLGrinder preferences categories.

🔍	Browser	Set preferences for the database browser.
🔌	Connections	Preferences for connections. Set statement, commit and wake from sleep behavior.
✓	General	General application preferences. Specify Java VM and window settings.

	Import/Export	Import and Export preferences. Set column delimiters and batch settings.
	JDBC Drivers	JDBC driver configurations. View, edit and add JDBC drivers.
	Logins	View, edit and add database logins and change login settings.
	Result Sets	Preferences for result set tables. Set the appearance and date format.
	SQL Editor	Set preferences for the SQL editor. Set the editor font, command history size and tab close confirmation.
	Syntax Coloring	Enable and disable syntax coloring and set the colors for keywords.
	Update	Update check settings.

Browser Preferences
Set preferences for the database browser.



Cache Settings

Disk cache: ☐ Cache schema data

Cache the schema data to disk when a connection to a database is made so the data does not have to be loaded each time.

Location: ~/Library/Application Support/SQLGrinder

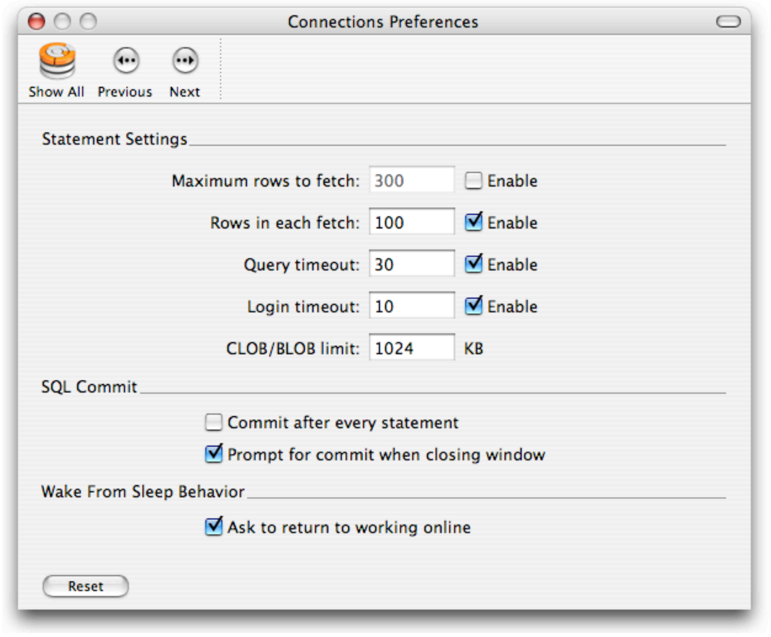
Data Tab Settings

☐ Automatically fetch data when tab is selected

- Cache Settings
    - Disk Cache - Save a cached version of the schema to disk for browsing offline.
- Data Tab Settings

- Automatically fetch data when tab is selected - When the data tab is selected in the browser, automatically load all of the rows for the table.

Connections Preferences
Preferences for connections. Set statement, commit and wake from sleep behavior.



Statement Settings

Maximum rows to fetch: 300 [ ] Enable

Rows in each fetch: 100 [✓] Enable

Query timeout: 30 [✓] Enable

Login timeout: 10 [✓] Enable

CLOB/BLOB limit: 1024 KB

SQL Commit

[ ] Commit after every statement

[✓] Prompt for commit when closing window

Wake From Sleep Behavior

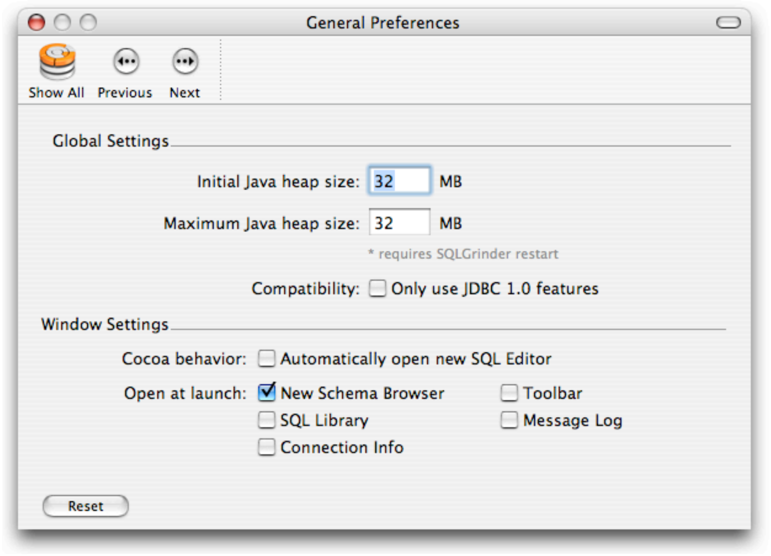
[✓] Ask to return to working online

- Statement Settings
  - Maximum rows to fetch - If enabled, limits the number of rows fetched to the specified value.
  - Rows in each fetch - If enabled, grabs the specified number of rows in each data fetch. A larger number can reduce database data “roundtrips” and speed data retrieval.
  - Query timeout - If enabled, times out a query after the specified number of seconds.
  - Login timeout - If enabled, times out a login attempt after the specified number of seconds.
  - CLOB/BLOB limit - If enabled, limits CLOBs and BLOBs to the size specified.
- SQL Commit
  - Commit after every statement - If checked, automatically commits after every statement (auto commit).

- Prompt for commit when closing window - If a transaction is open, window prompts for commit when it is closed.

- Wake From Sleep Behavior
  - Ask to return to working online - When the application is activated after computer is woken from sleep, ask to reconnect any connections that were open when computer was put to sleep.

General Preferences
General application preferences. Specify Java VM and window settings.



Global Settings

Initial Java heap size: 32 MB

Maximum Java heap size: 32 MB

* requires SQLGrinder restart

Compatibility: ☐ Only use JDBC 1.0 features

Window Settings

Cocoa behavior: ☐ Automatically open new SQL Editor

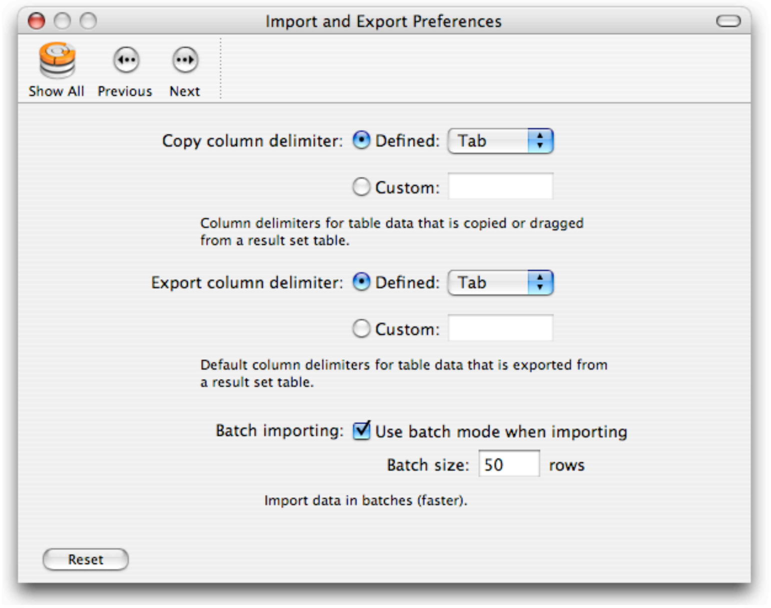
Open at launch: ☑ New Schema Browser ☐ Toolbar ☐ SQL Library ☐ Message Log ☐ Connection Info

- General Settings
  - Initial Java heap size - Sets the initial size of the memory heap that the Java VM uses.
  - Maximum Java heap size - Sets the maximum size of the memory heap that the Java VM uses. Increase this value for larger data sets.
  - Compatibility - Set application mode to be compatible with JDBC drivers that do not support JDBC 2 features, using only simple JDBC calls.

- Window settings
  - Cocoa behavior - Automatically open a new SQL editor when the application starts or comes to the foreground and there is no open editor.
  - Open at launch - Open the specified windows at application start.

Import/Export Preferences

Import and Export preferences. Set column delimiters and batch settings.



Copy column delimiter: [✓ Defined:] Tab
Custom:

Column delimiters for table data that is copied or dragged from a result set table.

Export column delimiter: [✓ Defined:] Tab
Custom:

Default column delimiters for table data that is exported from a result set table.

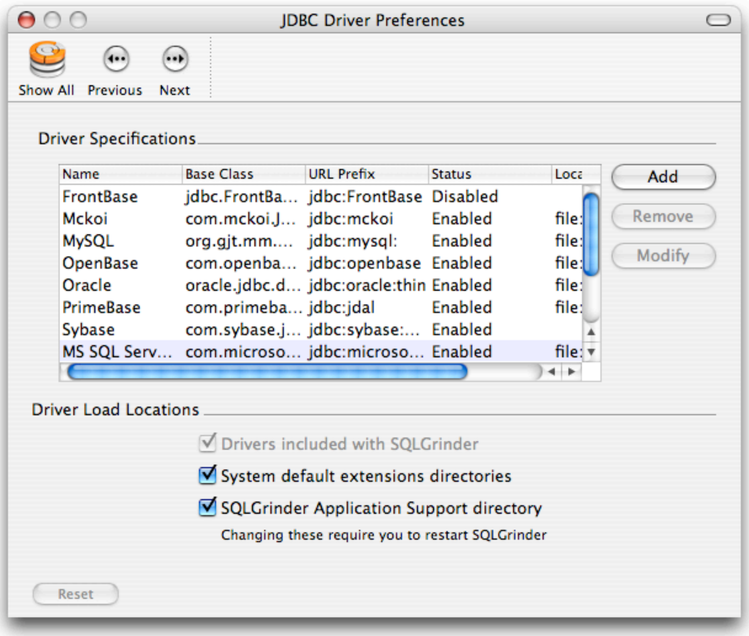
Batch importing: [✓ Use batch mode when importing]
Batch size: 50 rows

Import data in batches (faster).

- Copy column delimiter - The column delimiter for result set table data that is copied, by choosing Edit > Copy, or dragged.
- Export column delimiter - The default column delimiter for result set table data that is exported by choosing File > Export...
- Batch importing - When checked, sets the row batch size sent to the database in each “roundtrip.” Using batches for imports is often faster, and limits data transmission time.

JDBC Driver Preferences

JDBC driver configurations. View, edit and add JDBC drivers.



Driver Specifications

Name	Base Class	URL Prefix	Status	Loca
FrontBase	jdbc.FrontBa...	jdbc:FrontBase	Disabled	
Mckoi	com.mckoi.J...	jdbc:mckoi	Enabled	file:
MySQL	org.gjt.mm....	jdbc:mysql:	Enabled	file:
OpenBase	com.openba...	jdbc:openbase	Enabled	file:
Oracle	oracle.jdbc.d...	jdbc:oracle:thin	Enabled	file:
PrimeBase	com.primeba...	jdbc:jdal	Enabled	file:
Sybase	com.sybase.j...	jdbc:sybase:...	Enabled	
MS SQL Serv...	com.microso...	jdbc:microso...	Enabled	file:

Driver Load Locations

- [✓] Drivers included with SQLGrinder
- [✓] System default extensions directories
- [✓] SQLGrinder Application Support directory

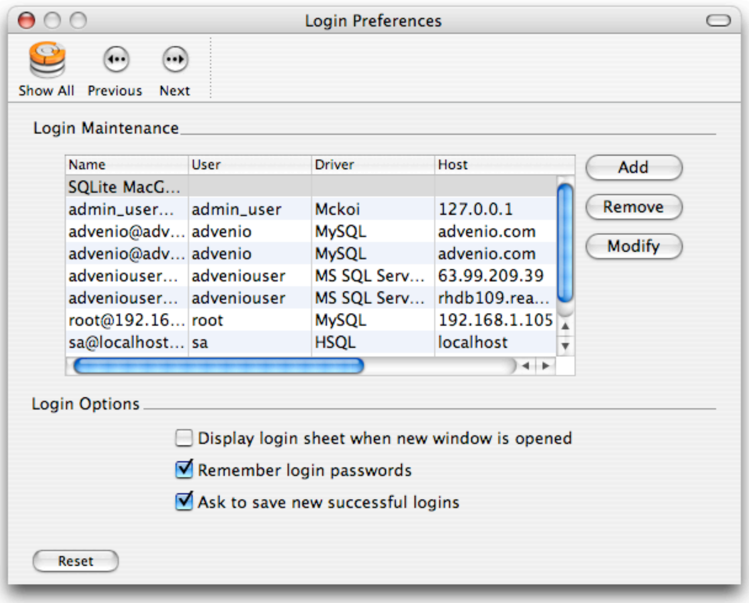
Changing these require you to restart SQLGrinder

- Driver Specifications - Allows JDBC driver configurations to be created, removed and modified. Allows the JDBC driver base class, url prefix and status to be set. Specifications colored in blue are built-in, and cannot be modified or deleted.

- Driver Load Locations - Specifies the locations used to look for and load JDBC drivers.

Login Preferences

View, edit and add database logins and change login settings.



Login Maintenance

Name	User	Driver	Host
SQLite MacG...			
admin_user...	admin_user	Mckoi	127.0.0.1
advenio@adv...	advenio	MySQL	advenio.com
advenio@adv...	advenio	MySQL	advenio.com
adveniouser...	adveniouser	MS SQL Serv...	63.99.209.39
adveniouser...	adveniouser	MS SQL Serv...	rhdb109.rea...
root@192.16...	root	MySQL	192.168.1.105
sa@localhost...	sa	HSQL	localhost

Login Options

- [ ] Display login sheet when new window is opened
- [✓] Remember login passwords
- [✓] Ask to save new successful logins

- Login Maintenance - Allows logins to be created, removed and modified. Setting the name of a login changes the label used in the login sheet, SQL statement library and Paste SQL menu.

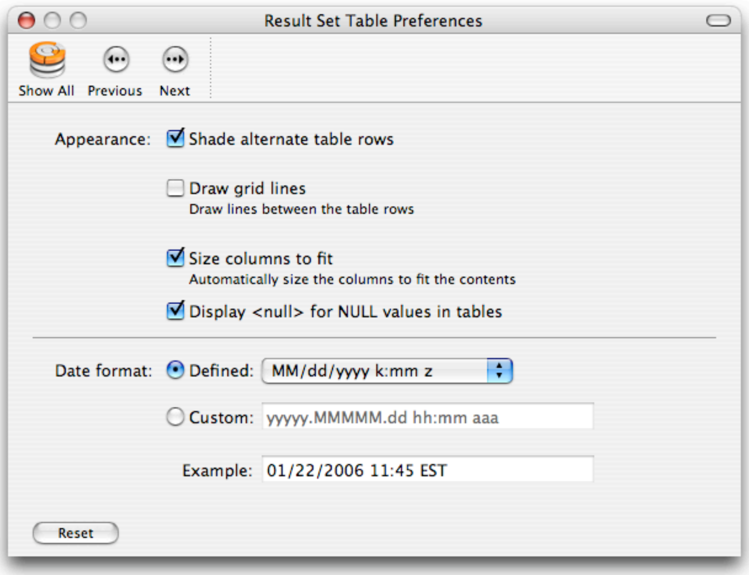
- Options

  - Display login sheet when new window is opened - If checked, automatically displays the login sheet when a new editor or browser is opened.

  - Remember login passwords - If checked, saves login passwords entered to the system keychain.

  - Ask to save new successful logins - If checked, a prompt is displayed that asks to save the login information when a new connection to a database is successful.

Result Set Preferences
Preferences for result set tables. Set the appearance and date format.



Appearance: [✓] Shade alternate table rows

[ ] Draw grid lines
Draw lines between the table rows

[✓] Size columns to fit
Automatically size the columns to fit the contents

[✓] Display <null> for NULL values in tables

Date format: (●) Defined: MM/dd/yyyy k:mm z

( ) Custom: yyyyy.MMMMM.dd hh:mm aaa

Example: 01/22/2006 11:45 EST

- Appearance

  - Shade alternate rows - If checked, every other row in a result set is colored.

  - Draw grid lines - If checked, lines are drawn between each row in a result set.

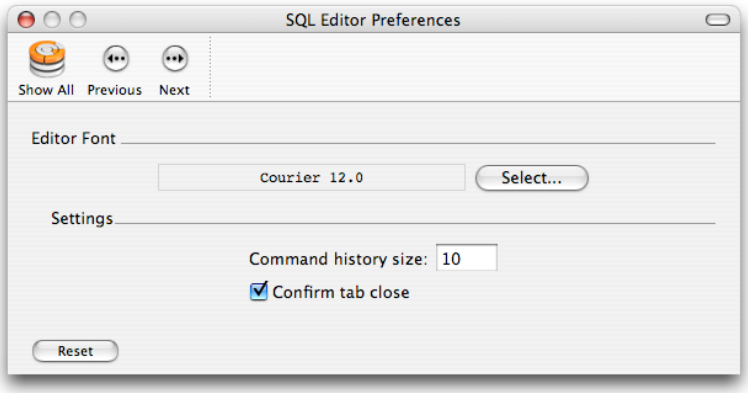
  - Size columns to fit - If checked, columns are sized to their contents, if possible, up to a maximum size.

  - Display <null> for NULL values in tables - If checked, when a cell value is NULL, the string <null> is displayed to show that this is the value.

- Date format - Displays data objects in result set columns using the specified predefined formats, or the specified custom format.

SQL Editor Preferences

Set preferences for the SQL editor. Set the editor font, command history size and tab close confirmation.



Editor Font

Courier 12.0

Settings

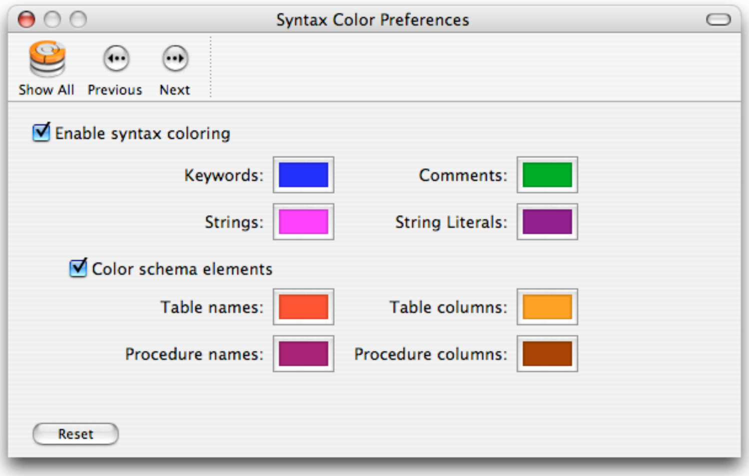
Command history size: 10

☑ Confirm tab close

- Editor Font - Specifies the font to be used in any SQL editor or SQL display view.
- Settings
  - Command history size - Specifies the number of result sets to maintain in the result set history drawer.
  - Confirm tab close - If checked, prompts the developer before a tab is closed.

Syntax Coloring Preferences

Enable and disable syntax coloring and set the colors for keywords.



☑ Enable syntax coloring

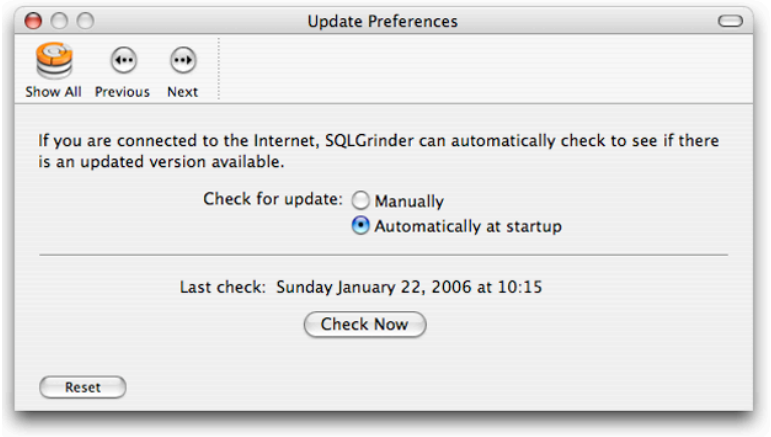
Keywords:	🟦	Comments:	🟩
Strings:	🟪	String Literals:	🟪

☑ Color schema elements

Table names:	🟥	Table columns:	🟧
Procedure names:	🟪	Procedure columns:	🟫

- Enable syntax coloring - If checked, enables syntax coloring for standard SQL syntax.
    - Keywords - Specifies the color for the standard SQL keywords.
    - Comments - Specifies the color for comments.
    - Strings - Specifies the color for strings.
    - String Literals - Specifies the color for string literals.
- Color schema elements - If checked, enables syntax coloring for schema table and procedure elements. If checked, table names, table columns, procedure names, and procedure parameters are all fetched from the database in a background thread, and used for colorization.
    - Table names - Specifies the color for table names.
    - Table columns - Specifies the color for table column names.
    - Procedure names - Specifies the color for procedure names.
    - Procedure parameters - Specifies the color for procedure parameters.

Update Preferences
Update checking settings.



If you are connected to the Internet, SQLGrinder can automatically check to see if there is an updated version available.

Check for update: ( ) Manually [✓ Automatically at startup]

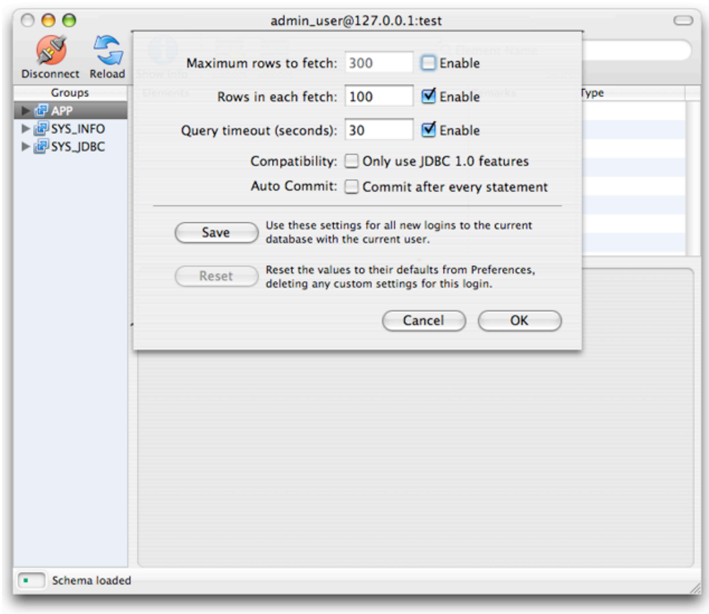
Last check: Sunday January 22, 2006 at 10:15

Specifies whether or not to check for a newer version of the application when it starts. No information is sent. The only data transmission is the retrieval of the latest version of the application from the Advenio web server, which is then compared to the current application version.

If set to Manually, the check is only done when the “Check Now” button is pressed. If set to “Automatically at startup” then the server is checked every time the application is started. If set to check, the check is only done once for the current 24 hour time period, rather than checking each time the application is launched in a single day.

Specifying the Settings for a Connection

SQLGrinder allows connection settings to be specified on a per-window basis for the life of the connection. This can be done by choosing Connection > Show Settings (⇧⌘K or shift-⌘-K) when an editor or browser with an active connection is the frontmost window.



Maximum rows to fetch: 300 ☐ Enable

Rows in each fetch: 100 ☑ Enable

Query timeout (seconds): 30 ☑ Enable

Compatibility: ☐ Only use JDBC 1.0 features

Auto Commit: ☐ Commit after every statement

Use these settings for all new logins to the current database with the current user.

Reset the values to their defaults from Preferences, deleting any custom settings for this login.

- Maximum rows to fetch - If enabled, limits the number of rows fetched to the specified value.
- Rows in each fetch - If enabled, grabs the specified number of rows in each data fetch. A larger number can reduce database data “roundtrips” and speed data retrieval.
- Query timeout - If enabled, times out a query after the specified number of seconds.
- Compatibility - Set application mode to be compatible with JDBC drivers that do not support JDBC 2 features, using only simple JDBC calls.
- Auto Commit - If checked, automatically commits after every statement (auto commit).

To use the settings once, set the values. They will be used just for the current window as long as that window remains open. To use these settings for any window that connects to a database using the user name and server information, press the Save button.

To clear the settings, and go back to using the application defaults, press the Reset button. The Reset button being enabled signifies that custom settings are currently being used for the user name and database specified in the login sheet.

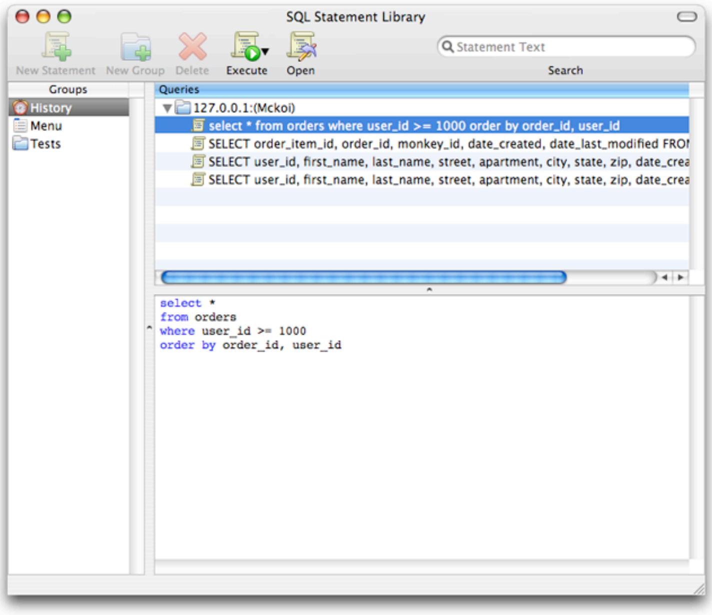
5

5. SQLGrinder Tools

This chapter describes the SQL statement library and the message log.

The SQL Statement Library

SQLGrinder provides an easy way to save and reuse SQL statements: the SQL statement library. The library allows SQL statements and any other text to be stored, grouped, and edited. Additionally, it also maintains the history list that displays all of the SQL statements sent to databases during a session. Finally, the library provides the means to configure the SQL Paste menu for the SQL editors.



127.0.0.1:(Mckoi)

select * from orders where user_id >= 1000 order by order_id, user_id

SELECT order_item_id, order_id, monkey_id, date_created, date_last_modified FRO

SELECT user_id, first_name, last_name, street, apartment, city, state, zip, date_crea

SELECT user_id, first_name, last_name, street, apartment, city, state, zip, date_crea

```
select *
from orders
where user_id >= 1000
order by order_id, user_id
```

Adding SQL statements to the library

There are various ways to add SQL statements to the library:

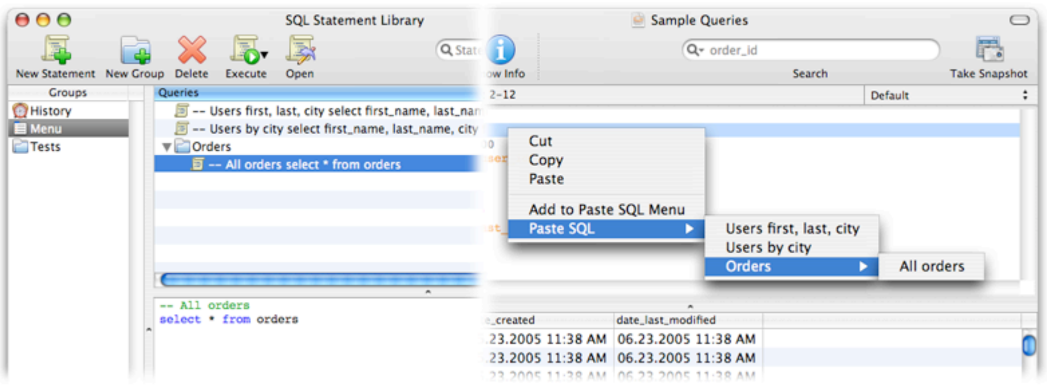
- Drag text from an editor or another application, to a selected library group.
- Select a library group and choose Edit > Paste to add text copied from an editor, or another application.
- Create a new statement by choosing Tools > SQL Statement Library > New Statement or by pressing the “New Statement” toolbar tool.

The Library History Group

The SQL statement library maintains a group with all of the SQL statements send successfully to any database during a current SQL session. Within the “History” group are subgroups with the names of the database logins that were used to send the statements. For instance, if a login named “Customer Database” was used to connect to a database and send a SQL statement, then that SQL statement would be found in a History subgroup named “Customer Database.” The History group cannot be modified, and it is cleared each time SQLGrinder is restarted. Statements can, however, be dragged from the History group to other groups and can be copied to the system pasteboard.

The Library Menu Group

SQL editors have a menu that is displayed whenever the right mouse button is clicked (or the ctrl and mouse button are clicked) on the text area, named “Paste SQL.” This menu is configured using statements saved in the library. Modifying this menu is done by modifying the Menu group. To make changes, add and remove SQL statements to this group. Subgroups can also be added to the menu group. These subgroups will appear as submenus in the Paste SQL menu. The menu item name displayed for a statement can be set by making the first line of the SQL statement a comment.



-- Users first, last, city select first_name, last_nar
-- Users by city select first_name, last_name, city
-- All orders select * from orders

```
-- All orders
select * from orders
```

_created	date_last_modified
23.2005 11:38 AM	06.23.2005 11:38 AM
23.2005 11:38 AM	06.23.2005 11:38 AM
23.2005 11:38 AM	06.23.2005 11:38 AM

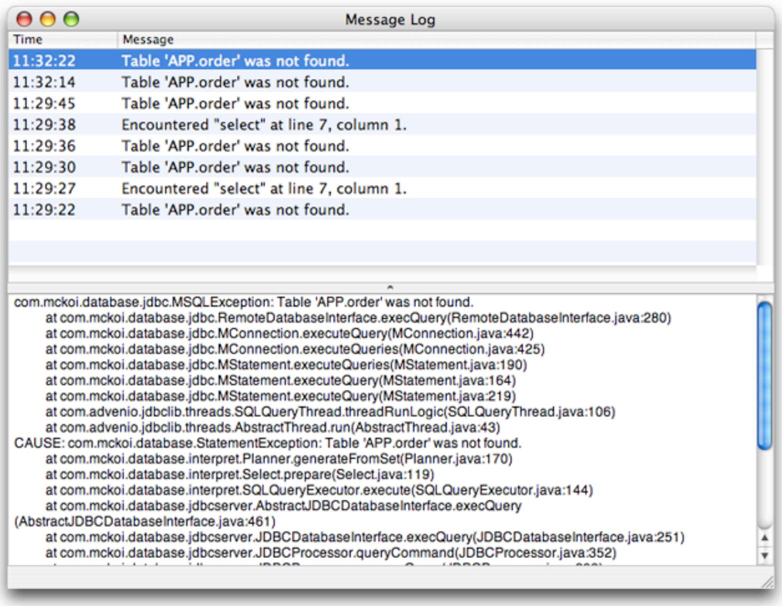
SQL Statement Library Toolbar Icons

The following tools are available on the SQL statement library toolbar. They can be added or removed by choosing View > Customize Toolbar... Choosing this menu item presents a sheet that allows tools to be dragged onto the toolbar to add them or dragged off to remove them.

Delete	Deletes the selected SQL statement or group from the library.
Execute	Opens a new editor setting the contents to the selected statement and executes it displaying any results.
New Statement	Add a new statement to the library.
New Group	Add a new group to the library.
Open	Opens a new editor setting the contents to the selected statement.

The Message Log

The Message Log displays messages generated while using SQLGrinder. The messages can be error messages from the database or application generated messages. When a message is added to the log, the time it was generated is stored along with it.



Time	Message
11:32:22	Table 'APP.order' was not found.
11:32:14	Table 'APP.order' was not found.
11:29:45	Table 'APP.order' was not found.
11:29:38	Encountered "select" at line 7, column 1.
11:29:36	Table 'APP.order' was not found.
11:29:30	Table 'APP.order' was not found.
11:29:27	Encountered "select" at line 7, column 1.
11:29:22	Table 'APP.order' was not found.

```
com.mckoi.database.jdbc.MSQLException: Table 'APP.order' was not found.
        at com.mckoi.database.jdbc.RemoteDatabaseInterface.execQuery(RemoteDatabaseInterface.java:280)
        at com.mckoi.database.jdbc.MConnection.executeQuery(MConnection.java:442)
        at com.mckoi.database.jdbc.MConnection.executeQueries(MConnection.java:425)
        at com.mckoi.database.jdbc.MStatement.executeQueries(MStatement.java:190)
        at com.mckoi.database.jdbc.MStatement.executeQuery(MStatement.java:164)
        at com.mckoi.database.jdbc.MStatement.executeQuery(MStatement.java:219)
        at com.advenio.jdbclib.threads.SQLQueryThread.threadRunLogic(SQLQueryThread.java:106)
        at com.advenio.jdbclib.threads.AbstractThread.run(AbstractThread.java:43)
CAUSE: com.mckoi.database.StatementException: Table 'APP.order' was not found.
        at com.mckoi.database.interpret.Planner.generateFromSet(Planner.java:170)
        at com.mckoi.database.interpret.Select.prepare(Select.java:119)
        at com.mckoi.database.interpret.SQLQueryExecutor.execute(SQLQueryExecutor.java:144)
        at com.mckoi.database.jdbcserver.AbstractJDBCDatabaseInterface.execQuery
(AbstractJDBCDatabaseInterface.java:461)
        at com.mckoi.database.jdbcserver.JDBCDatabaseInterface.execQuery(JDBCDatabaseInterface.java:251)
        at com.mckoi.database.jdbcserver.JDBCProcessor.queryCommand(JDBCProcessor.java:352)
```

The message log can be shown by choosing Tools > Message Log. It can be cleared by selected any number of messages and pressing the Delete key on the keyboard.

When a new message is added to the message log, and the log is not in the foreground, a red status indicator is displayed in any open editor or browser. This is to show that new messages were added to the log. Bringing the Message Log to the foreground will clear the status indicator until another new message is added. The unread messages status indicator can also be cleared by choosing Tools > Clear Unread Messages.

6

6. AppleScript and Automator

This chapter introduces you to SQLGrinder's AppleScript and Automator support.

SQLGrinder features full support for AppleScript, using AppleScript Studio and Automator.

Using AppleScript and SQLGrinder

SQLGrinder provides an AppleScript interface that allows windows and data to be manipulated using AppleScript, AppleScript Studio and Automator actions. Using AppleScript, editors and browser windows can be opened, connected, disconnected, and closed, and data can be fetched and manipulated. SQLGrinder can also be used as a faceless conduit that AppleScript applications can use to communicate with any database with a JDBC driver. With SQLGrinder running, AppleScript scripts, applications and Automator actions can use it to send and receive data without opening any windows or requiring any user actions.

Scripts using the SQL editor

This section describes ways to use and manipulate an open, existing and connected SQL editor. Information for making database connections is always specified using the name of a Login.

The editor supports the editor row count command, from the SQLGrinder suite, which returns the number of rows in the current result set.

```
(* Get the count of the rows in the result set *)
set rowCount to editor row count
```

The editor also supports the editor result set columns command, also in the SQLGrinder suite, which returns the names of the columns in the result set.

```
(* Get the column names list *)
```

```
set columnNames to editor result set columns
```

The editor result set command returns a reference to the result set itself, in the form of an array of arrays. The outer array is the array of rows. The inner arrays are the column values. You can loop through the rows and columns of the result set arrays using the following example:

```
(* Get the current result set from the editor *)
set resultSet to editor result set
set columnCount to length of columnNames

(* Loop through the columns of the result set, building a string of comma
delimited rows separated by returns *)
repeat with i from 1 to rowCount
      repeat with j from 1 to (columnCount - 1)
            set row to row & item j of item i of resultSet & ", "
      end repeat
      set row to row & item (j + 1) of item i of resultSet & return
end repeat
```

A file with this example, named “Process Result Set Example”, can be found in the SQLGrinder AppleScript Dev Kit.

Scripts using the database browser
This section describes ways to retrieve schema data using the database browser.

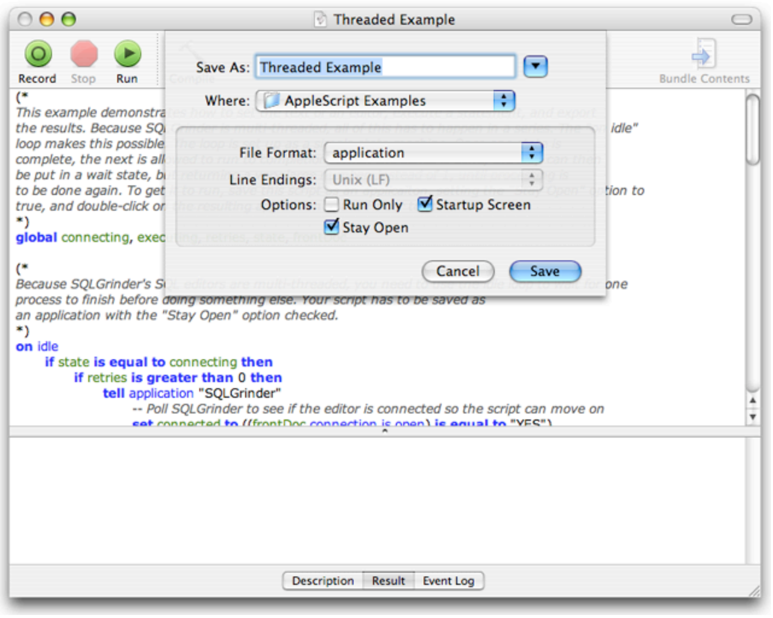
```
(* Select the object in the browser with the specified path. Path looks like:
   /category/schema/type group/object. NOTE: case matters
*)
select object with path "/APP/tables/orders"


(* Get the list of table names. % is a wild card character. *)
get table names catalog "%" schema "%" table name "%"

(* Get the list of the schemas *)
get schemas

(* Get the names of the columns for the table 'customer'. % is a wild card
   character *)
get table column names catalog "%" column name pattern "%" schema "%"
      table name pattern "customer"

(* Get all of the column information for the table 'customer'. % is a wild card
    character *)
get table columns catalog "%" column name pattern "%" schema "%" table
      name pattern "customer"
```

Scripts using the SQL commander
This section describes using the SQL commander to get data. The SQL commander allows the use of SQLGrinder as an AppleScript client, without needing any open editors or browsers to do the work. This can be thought of a the “faceless” mode for SQLGrinder. The following command sends a select statement to the database, connecting using the login named “My Database Login” and using the “,” character as a delimiter between columns.

```
execute sql with text "select * from customer where LAST_ORDER_ID =
'4636'" using "My Database Login" delimiter ","
```

The SQL commander is used primarily to issue SQL commands to your database and get the resulting data in a the form of a string, with the specified delimiter between columns, and return characters between the rows.

Because SQLGrinder is threaded, the commander is the easiest way to send SQL to the database and get back results. Using an editor window to do the same thing requires more advanced AppleScript that uses timers to wait for each step to finish before continuing. You can find an AppleScript file that shows how this is done, called “Threaded Example” in the SQLGrinder AppleScript Dev Kit. This example file uses the **on** idle command, so to run it, you’ll need to save this script as an AppleScript application, setting the Stay Open option to true, and then double-click on the application in the Finder.



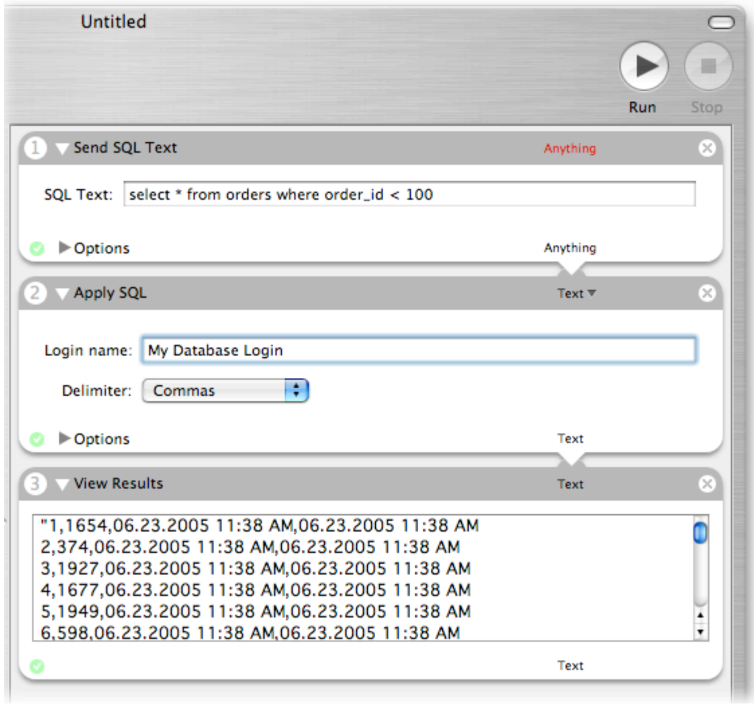
Save As: Threaded Example
Where: AppleScript Examples
File Format: application
Line Endings: Unix (LF)
Options: ☐ Run Only ☑ Startup Screen
☑ Stay Open

```
(*
This example demonstra
the results. Because SQ                                                        idle"
loop makes this possible
complete, the next is all
be put in a wait state, b
to be done again. To get                                                   tion to
true, and double-click or
*)
global connecting, exec

(*
Because SQLGrinder's S                                                          one
process to finish before doing something else. Your script has to be saved as
an application with the "Stay Open" option checked.
*)
on idle
    if state is equal to connecting then
        if retries is greater than 0 then
            tell application "SQLGrinder"
                -- Poll SQLGrinder to see if the editor is connected so the script can move on
                set connected to ((frontDoc connection is open) is equal to "YES")
```

Using Automator and SQLGrinder

This section describes using Automator with SQLGrinder. Available separately and as part of the SQLGrinder AppleScript Dev Kit are 3 Automator actions for communicating with any JDBC database using SQLGrinder as a conduit.

The "Apply SQL" action takes a query in the form of a string from another action and passes it to the database associated with the SQLGrinder login that is specified. Results in the form of a string are returned, using the specified delimiter between columns and return characters between the rows.



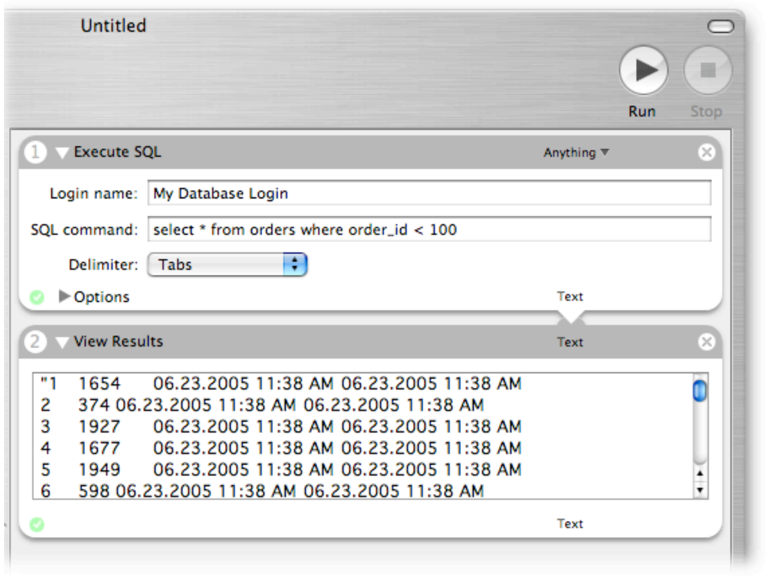
SQL Text: select * from orders where order_id < 100

Login name: My Database Login

Delimiter: Commas

```
"1,1654,06.23.2005 11:38 AM,06.23.2005 11:38 AM
2,374,06.23.2005 11:38 AM,06.23.2005 11:38 AM
3,1927,06.23.2005 11:38 AM,06.23.2005 11:38 AM
4,1677,06.23.2005 11:38 AM,06.23.2005 11:38 AM
5,1949,06.23.2005 11:38 AM,06.23.2005 11:38 AM
6,598,06.23.2005 11:38 AM,06.23.2005 11:38 AM
```

The "Execute SQL" action takes the specified SQL statement and passes it to the database associated with the SQLGrinder login that is specified. Like the "Apply SQL" action, results in the form of a string are returned, using the specified delimiter between columns and return characters between the rows.



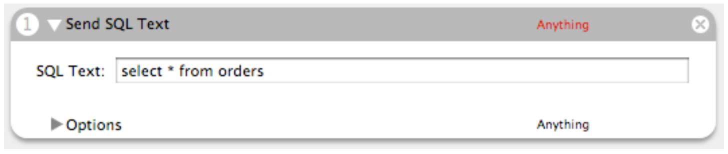
Login name: My Database Login

SQL command: select * from orders where order_id < 100

Delimiter: Tabs

```
"1	1654	06.23.2005 11:38 AM 06.23.2005 11:38 AM
2	374 06.23.2005 11:38 AM 06.23.2005 11:38 AM
3	1927	06.23.2005 11:38 AM 06.23.2005 11:38 AM
4	1677	06.23.2005 11:38 AM 06.23.2005 11:38 AM
5	1949	06.23.2005 11:38 AM 06.23.2005 11:38 AM
6	598 06.23.2005 11:38 AM 06.23.2005 11:38 AM
```

Finally, the “Send SQL Text” action allows any entered SQL text to be passed to another action. This action is primarily for testing your actions before adding them to workflows.



SQL Text: select * from orders

The source code and projects for all of these actions can be found in the SQLGrinder AppleScript Dev Kit.

A

Appendix A

Supported JDBC Drivers

This is the list of database drivers that have been verified to be compatible with SQLGrinder. If your database is not found in the list, that doesn't mean that it is incompatible, it just means that it has not been officially verified and tested by Advenio.

SQLGrinder requires drivers to be at least JDBC 2 compliant to enable all functionality. If a driver you are using is not at least 2 compliant, features of SQLGrinder will not work. In some cases, the driver will be unusable. SQLGrinder provides a compatibility mode, both in preferences and for connections, that limits SQLGrinder to only using features found in JDBC 1.

Drivers bundled with SQLGrinder

- FrontBase
- MySQL
- OpenBase
- Oracle
- PostgreSQL
- Sybase

Other supported drivers

- DB2
- Hypersonic SQL
- Mckoi
- Microsoft SQL Server
- Primebase
- SQLite

DB2

IBM Toolbox for Java, JTOpen

“The "Toolbox" JDBC driver. This is shipped as part of the IBM Toolbox for Java (57xxJC1). It is implemented by making direct socket connections to the database host server. This happens to be the same route that the Client Access/400 ODBC driver takes. However, Client Access/400 is NOT required. The Toolbox runs on any JVM. The class name to register is com.ibm.as400.access.AS400JDBCDriver . The URL subprotocol is as400.”

http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm

FrontBase

FrontBase JDBC Driver

www.frontbase.com

Hypersonic SQL (HSQL)

“HSQLDB is Java developers' best choice for development, testing and deployment of database applications. The latest HSQLDB 1.8.0 is fast, powerful and reliable -- more so than ever.”

http://hsqldb.org/

Mckoi

“Mckoi SQL Database is an SQL (Structured Query Language) Database management system written for the JavaTM platform. Mckoi SQL Database is optimized to run as a client/server database server for multiple clients, however it can also be embedded in an application as a stand-alone database. It is highly multi-threaded and features an extendable object-oriented engine.”

http://mckoi.com/database/

Microsoft SQL Server

• jTDS

“TDS is an open source 100% pure Java (type 4) JDBC 3.0 driver for Microsoft SQL Server (6.5, 7, 2000 and 2005) and Sybase (10, 11, 12). jTDS is based on FreeTDS and is currently the fastest production-ready JDBC driver for SQL Server and Sybase. jTDS is 100% JDBC 3.0 compatible, supporting forward-only and scrollable/updateable ResultSets, concurrent (completely independent) Statements and implementing all the DatabaseMetaData and ResultSetMetaData methods.”

http://jtds.sourceforge.net/

• Microsoft SQL Server 2000 Driver for JDBC

> "The Microsoft® SQL Server™ 2000 Driver for JDBC™ is a Type 4 JDBC driver that provides highly scalable and reliable connectivity for the enterprise Java environment. This driver provides JDBC access to SQL Server 2000, both 32 bit and 64 bit editions, through any Java-enabled applet, application, or application server."

http://tinyurl.com/3v7tc

MySQL

MySQL® Connector/J

> "MySQL Connector/J is a native Java driver that converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database. It lets developers working with the Java programming language easily build programs and applets that interact with MySQL and connect all corporate data, even in a heterogeneous environment. MySQL Connector/J is a Type IV JDBC driver and has a complete JDBC feature set that supports the capabilities of MySQL."

http://www.mysql.com/products/connector/j/

OpenBase

OpenBase JDBC 3.0 Driver Client

http://store.openbase.com/downloads-Instructions.221.html

Oracle

• Oracle Database 10g Release 2 (10.2.0.1.0) drivers

• Oracle9i Release 2 (9.2.0.5) (9.2.0.4) (9.2.0.3) & (9.2.0.1) drivers

• Oracle8i Release 2 (8.1.7)

http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

PostgreSQL

PostgreSQL JDBC Driver

> “Allows Java programs to connect to a PostgreSQL database using standard, database independent Java code. It is a pure Java (Type IV) implementation, so all you need to do is download a jar file and you're on your way.
>
> The driver provides are reasonably complete implementation of the JDBC 3 specification in addition to some PostgreSQL specific extensions.”
>
> http://jdbc.postgresql.org/

Primebase

http://www.primebase.com/en/index.html

SQLite

“SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine.”

http://www.sqlite.org/

Notes on using SQLite type 3 driver with SQLGrinder:

There is a very good SQLite driver made available by Tim Anderson here:

http://www.itwriting.com/sqlitenotes.php

To use it, go to the web page above and click the "Mac OS X 10.4 JNI wrapper" link. Take the two files in the downloaded archive, named sqlite.jar and libsqlite_jni.jnilib and put them in the Library/Java/Extensions (you may have to create these folders) folder in your home folder and restart SQLGrinder. SQLGrinder will automatically discover the driver and "SQLite" will appear in the drivers list in the login sheet.

A couple of caveats:

1. Your database file cannot have spaces in the name, this is a JDBC thing, not a Mac or SQLite thing.

2. The URL field in the sheet when SQLite is selected should look something like "jdbc:sqlite://Users/username/pathtodatabasefile, the URL checkbox should be checked, and no username or password are required.

Sybase

jConnect for JDBC

“jConnect provides high performance native access to the complete family of Sybase products including Adaptive Server Enterprise, Adaptive Server Anywhere, Adaptive Server IQ, and Replication Server. Through ASE/CIS (formerly OmniConnect), it provides transparent connectivity to more than twenty five enterprise and legacy database servers. It can also directly access Oracle, AS/400 and others via DirectConnect.”

http://www.sybase.com/products/informationmanagement/softwaredeveloperkit/jconnect

B

Appendix B

Default Database JDBC Ports

Listed below are some of the default ports for common databases.

Database	Port
Microsoft SQL Server	1433
MySQL	3306
Oracle	1521
PostgreSQL	5432
Sybase ASA	2638/11222
Sybase ASE	2048