Squeak DBX

"The complete and open-source solution to relational database access"

ESUG 2009 Innovation Technology Awards

Software: SquekDBX
Developers: Mariano Martínez Peck**,** Hernán Cassinelli**,** Germán Palacios, Alejandro Salanova. Coordinated by Esteban Lorenzano.

Affiliation and Country: Universidad Tecnológica Nacional, Buenos Aires, Argentina.
Dialect: Squeak & Pharo
Licence: M.I.T

URL: http://wiki.squeak.org/squeak/6052
Keywords: Squeak, Pharo, persistence, database, driver, OpenDBX

Abstract
SqueakDBX is a database driver that allows interaction with major database engines such as Oracle and MSSQL, apart from those which are open source, like PostgreSQL and MySQL. Moreover, integrating this plugin with GLORP, will enable a complete and open-source solution to relational database access. To do this, SqueakDBX uses a library called OpenDBX.

Short description

Present situation

Some of you may think that object databases are more suitable for store objects than its counterparts, relational databases. However, most of the information in the world is kept in relational storage.
Recently, Squeak has started being taken into account for enterprise projects probably thanks to projects like Pharo, Seaside, AidaWeb, etc. Therefore, many problems have not been considered….until now. Still nowadays most clients want to use a RDBMS. Indeed, they want to use a specific database. Mainly because they have licenses, they know it, they have it already installed, they do not want to spend time migrating data among other reasons.

Problem encountered

So, what's the problem? Well, Squeak only communicates natively to MySQL or PostgreSQL, but not to others so, in those cases where the client already has a RDBMS that Squeak does not support native access for, without SqueakDBX's existence, there was no way to do the project. In such cases, another technology had to be used. This is exactly what SqueakDBX does: communicate with major database engines through a simple, unified and object oriented interface, eliminating one of the most common restrictions people have when using Squeak for real enterprise applications.

Object Relational mapping

There is also an ORM (Object Relational Mapper) called GLORP (Generic Lightweight Object-Relational Persistence) which is an open-source object-relational mapping layer for Smalltalk. Its goal is to provide a simple, yet powerful framework for reading and writing objects from relational databases.

The magic

Nowadays, there is a library called OpenDBX, which is an extremely lightweight but extensible database access library written in the C programming language, and that aims to implement access to the native database interfaces efficiently, through a thin abstraction layer.

The OpenDBX library is open source and licensed via the LGPL license. Currently, it supports Firebird, Interbase, MS SQL Server, MySQL, Oracle, PostgreSQL, SQLite 2 and 3, ODBC and Sybase ASE.

Contrary to interfaces like ODBC, the OpenDBX library favors speed and flexibility and therefore doesn't try to unify the SQL query language. It enables the application developer either to use only standardized SQL elements or more efficient database specific language elements as well. Furthermore, this approach allows adding support for databases which don't implement SQL as query language.

The results of this work have enabled the community to interact with major database engines through a truly object oriented and open source solution.

Main targets

- Complete support of OpenDBX and all the databases it includes.
- Glorp integration
- Maintenance due to changes on OpenDBX that impacts on SqueakDBX, for example, API changes or new functionality.

Current features

- Cross-platform support: Linux, Windows (using MinGW) and Mac
- Mini VM locks when using FFI (asynchronous queries)
- Own SqueakDBX plugin
- Support for: Oracle, PostgreSQL, MySQL, MS SQL Server, ODBC and SQLite3
- Transactional concept: begin transaction, commit and rollback
- Mappings from String to specific types in selects
- Special OpenDBX options: multistatements, encryption, compression, paged results, mySQL modes, and more
- Automated database connection release (although manual disconnection is recommended)
- Automated results retrieving in order to do another query, after doing a query and not iterating ALL results

- Lots of unit tests that backup our project
- Lots of benchmarks and comparisons with native drivers (PostgreSQL and MySQL).
- Error handling: Not only errors, but levels associated with an error in order to avoid FFI calls (if you get a fatal error, it has no sense to do another query and the resources must be freed).
- Query timeout (this is very useful for webapplications) and pageSize can be set for each query.

Expected features

- Support for: Firebird, Interbase and Sybase
- Ability to be used as a GLORP driver

Examples

Using SqueakBDX is easy and pretty straight forward! Just connect to your database, send your query, retrieve the result sets and process the row values. If you've finished your job, you have to disconnect from the database and clean up all resources, really easy, right?

Now, let's see how this is done, first of all, you have to connect to the database:

Connection

Fill in the connection info:

```
| conn connectionSettings |
   connectionSettings := DBXConnectionSettings
                  host: 'localhost'
                  port: '5432'
                  database: 'sodbxtest'
                  userName: 'sodbxtest'
                  userPassword: 'sodbxtest'.
```

Then, there is the class DBXConnection that handles the connection to the database you want.

```
conn := DBXConnection platform: DBXPostgresPlatform new settings: connectionSettings.
```

In this example, we use a PostgreSQL database, but you can use any of the supported backends.

Untill now we have only created a connection object. You should now open the real TCP connection with the database. To do this, you will need to do two things: connect to database and then open the connection.

```
conn connect.
conn open.
```

Sending queries

Now, we are able to send queries. The only thing you need is the “execute” message in DBXConnection. Here are some examples:

```
conn execute: 'insert into DBTableTest(name, age) VALUES (''Mr. Squeak'', 22) '.
result := conn execute: 'delete from DBTableTest where name = ''Miss Java'''.
result := conn execute: 'select name from DBTableTest'.
result := conn execute: 'update DBTableTest set name = ''Mr. Squeak'' where id = 2'.
```

Retrieving results

The message execute of DBXConnection returns a DBXResultSet if it was a select statement, or a DBXResult in case the query was an update, delete, insert, create, drop or any other DML query.

DBXResult example:

```
result := conn execute: 'delete from DBTableTest where name = ''Miss .Net'''.
Transcript show: result rowsAffected asString, ' affected rows';.
```

DBXResultSet example:

```
result := conn execute: 'SELECT name FROM DBTableTest'.
columnDescription := result columnDescriptionAt: 1.
columnDescription := result columnDescriptionWithName: 'name'.
Transcript show: columnDescription dbxType, columnDescription name,  columnDescription    size,
columnDescription type.
columnCount := result columnCount.
aRow := result nextRow.
```

Processing results

Using nextRow or rowsDo enables you to obtain the rows and do anything you want with them. You have retrieved the results, so now you are able to process them. In order to do this, you may use some of the following classes:

DBXRow: Represents a row of a resultset. It has these useful messages:

valueAt: anIndex and valueNamed: aString. They return the value at that position or name. These values are automatically converted from String (OpenDBX returns String objects always) to the specific Squeak type.
values: Returns an Array with all the returned values.

Disconnection/Close

After you have done all you want with the database, you should now disconnect from it and release all resources. The correct way of doing this is:

```
conn close.
conn disconnect
```