Firebird Conference
Prague 2005

FlameRobin
Administration tool for Firebird

Milan Babuškov
www.flamerobin.org

About the author

Education:

2001 - B.Sc. In Business Information System Engineering

2003 - M.Sc. In Internet Technology at University of Belgrade

Started to program as a 14 year old, making simple games in BASIC and later assembler on Motorola's 680x0 series CPUs. Programmed in C, Perl and Java. Now writing the code in C++ and PHP. Started to work with Informix at University, after few experiments with Paradox, MySQL and MSSQL, finally switched to Firebird. Starting from 2002, developing various software using Firebird, PHP and Apache.

Developer of open source FBExport and FBCopy tools, for manipulation of data in Firebird databases. In 2003, started a project to build a lightweight cross-platform graphical administration tool for Firebird. The project was later named FlameRobin, and is built entirely with open source tools and libraries.

Hobbies include playing basketball and writing cross-platform computer games, some of them very popular (Njam has over 36000 downloads on sf.net):

http://njam.sourceforge.net

http://abrick.sourceforge.net

http://scalar.sourceforge.net

Born in 1977. Still single.

Live and work in Subotica, Serbia. Currently employed at large ISP company.

Introduction

What is FlameRobin?

FlameRobin is a graphical administration and data manipulation tool for Firebird DBMS. It features:

- small footprint
- ports to many platforms
- build requires only open source tools and libraries

Small footprint means many things. First, there is the executable size and download size of entire package. The goal is to follow the fashion of Firebird which gives us full featured database server in just a few megabytes. Current versions of FlameRobin are under 1.5 megabytes for Windows and around 2 megabytes for other ports. Secondly, there is the startup time. Many times a Firebird administrator (and even a developer) can get into situation where (s)he needs to run few quick queries and disconnect. Most of the currently available tools take few seconds to load and startup, so people mostly choose isql (Firebird's interactive command-line utility) for such tasks. FlameRobin starts almost instantly, so it's suitable for such tasks, and it does offer much more than isql. Being lightweight does not mean to be poor in features, they just have to be shown at right places and sometimes even loaded on demand (postponed until needed).

One of the project's goals is to make the tool available to as many platforms and operating systems as possible. If it's possible, we aim to cover all platforms where Firebird is available. Currently the ports to Windows, Linux, Mac OS X and FreeBSD are available. You can read more about those ports in a separate section of this document.

One of the requirements we set up when project started was not to require any closed-source software to build the tool. In order to build FlameRobin one only needs open sourced tools and libraries. The build process uses GNU or BSD make and GNU C++ Compiler. Of course, one can also use Microsoft's Visual C++ if (s)he desires, as it does work much faster than GCC. All libraries that are used are also released under open source licenses. Finally, the FlameRobin itself is released under Initial Developer Public License, IDPL.

What FlameRobin is NOT?

FlameRobin is not a full-featured, does-it-all tool. We partly follow the Unix philosophy to make simple tools that do specific tasks, but do it properly. It is “partly”, since if it would be “completely”, then FlameRobin itself would probably consist of many minor tools (one to run queries, other to backup and restore databases, etc.) However, short term project's goal is to provide a basic tool for beginners to get acquainted with Firebird. Such tool should provide access to all the features Firebird provides. That tool should be FlameRobin in version 1.0 once it's completed. Our long term goal is to provide a powerful tool for *nix systems, where lack of quality tools is evident. As many more people start to use Linux and MacOSX on desktops, FlameRobin would fill that gap and remove the need to run a separate Windows box to administer the server or run some Windows tool via emulator. Some may argue that great commercial and freeware Windows tools run fine in emulator, but that's really a second-class Windows environment, far from native one that FlameRobin offers.

Project's History

It all started at beginning of 2003. on some of Firebird's mailing lists. Once again someone noted that there is a need for graphical administration tool for *nix systems. Many people agreed about it, and finally there was a will to make something ourselves instead of just talking about it. Paul Schmidt started a FBManager project at sourceforge.net, and people from mailing list joined to see what we could come up with.

Choice of tools and libraries

The first task was to choose the programming language and library to communicate with Firebird. There were many suggestions and interesting alternatives. One of the things we first considered was Java and JayBird. JayBird offers great support for Firebird, and Java promises wide cross-platform support. However, Java doesn't fit in our lightweight philosophy. Some potential users who don't have Java Virtual Machine installed would meet a 14MB download. Also, Java runs slow on machines with low memory, so it was finally discarded.

The other choice was Python and KinterbasDB. This still looks like an interesting option, however, the lack of developers (only one person merely interested), made us avoid that option too. We also considered to use PHP and build a web-based tool. That approach has many drawbacks however: it requires installation of web server, and browser interface is still behind classic thick-client GUI when it comes to efficiency and user productivity.

Finally, that left us with C++. There were enough developers interested in programming and efficiency wasn't even questionable. As for connectivity library, the first thought was to use the Firebird C API. However, most of us did try out the excellent IBPP library (www.ibpp.org), which is a C++ thin wrapper around Firebird's C API. It gives us many benefits. We don't have to worry about error-prone C coding, and more importantly, it promises forward compatibility. In case Firebird's API change, IBPP would change, leaving the same interface to FlameRobin. If we weren't using IBPP, we would probably have to write a similar database access layer ourselves. During all this time of FlameRobin development, IBPP has proven as very stable and error-free solution, and support we get from lead developer Olivier Mascia is really excellent.

After we made a decision to use C++, we needed to decide which toolkit to use to provide the GUI. There are many toolkits available, but we only considered a few: nCurses, Qt, Gtk and wxWidgets. One of the first ideas was nCurses as it is a common toolkit on *nix systems. All these systems use the console/terminal, and this is a natural choice for cross-platform project. A good example of program that uses ncurses is the famous file manager, Midnight Commander. The problem with ncurses is that it is character based, and far from a modern GUI. It would surely scare away a typical Windows user that downloaded Firebird for the first time. Well, probably not any more than isql, but still. There are still plans to make a ncurses variant of FlameRobin one day.

Qt toolkit from Trolltech was an interesting alternative. The entire KDE environment is built upon it. It works fast and looks really nice. At the time the choice was made, the problem was non-Linux support. We didn't know if MacOSX port existed at all, and one was required to pay license for Windows port at that time. We also considered gtk, but nobody was familiar with it.

Finally, the decision was made to use wxWidgets. It's a true cross-platform toolkit that exists for twelve years and provides ports to many platforms. It's uncommon in sense that it wraps around native toolkit on a platform instead of drawing all the controls itself. On Windows it uses the Windows API. On Linux and FreeBSD it gives option between Gtk (both gtk1 and gtk2), motif and X11. On Mac OS X it uses Carbon, and Cocoa port is on the way. All this makes the applications using it look and feel native on each platform. In fact, those are native applications, which just use the additional layer that makes platform differences almost transparent. One of the benefits is also a Bakefile build system introduced in version 2.6, which lets us, the developers, use a single file to define all the project's files, allowing us to easily create makefiles for multiple compilers and platforms from a single source. Beside cross platform stuff, wxWidgets also offers various components needed in application development like: clipboard support, drag and drop, multi-threading, image loading and saving in various formats, HTML viewing support, printing, etc. If we were to start FlameRobin from scratch, we would surely pick it again.

A real start

In January 2004., Paul Schmidt decided to leave the project, so I took over as project manager. I hacked up a first code that actually did something and placed in on-line for download. At that time, it looked more like a one-man show than a great community effort. However, that was to change really quick. In February 2004., after another round of discussion on Firebird's mailing lists about the need for an open-source administration tool, Nando Dessena joined the project. He was soon followed by Gregory Sapunkov. We revisited the project's goals, decided upon initial bare-bones of code architecture and started building it. During the next few months, a skeleton of FlameRobin was made, with crucial new features added on a daily basis. It was evident that project has the future.

In July 2004., the first alpha version went out and got thousands of downloads, which really gave us the boost to keep going. We kept adding new features, and even started to fix some reported bugs. In October, Michael Hieke joined the project. Initially as a MacOSX porter, he turned out to be much more, giving us the insight not only in cross-platform development area, but also in GUI and wxWidgets usage area. During all this time we had great support by Marius Popa, who did a great PR job, just as he does for Firebird, and Alex Stanciu who designs and maintains the FlameRobin website.

The name

In may 2004, the project was renamed to FlameRobin, giving it distinctiveness it needed, as FBManager was really a general name, with potential to clash with many others. We were looking for potential name for some time. Various variants of "manager" were thought of, and also various acronyms, but none were good enough. We went on looking for a bird that would have some connection with fire, trying to resemble firebird in some way. FlameRobin is one that stuck. FlameRobin (petroica phoenicea) is small bird of the robins family. It can be found in southern Australia and Tasmania. The birds measure 12.5 to 14 cm long. The "flame" comes from the male birds, which have bright orange breast and throat when they grow up. The logo is designed by Barbara Del Vecchio and resembles the colors of a real bird, with head being black.

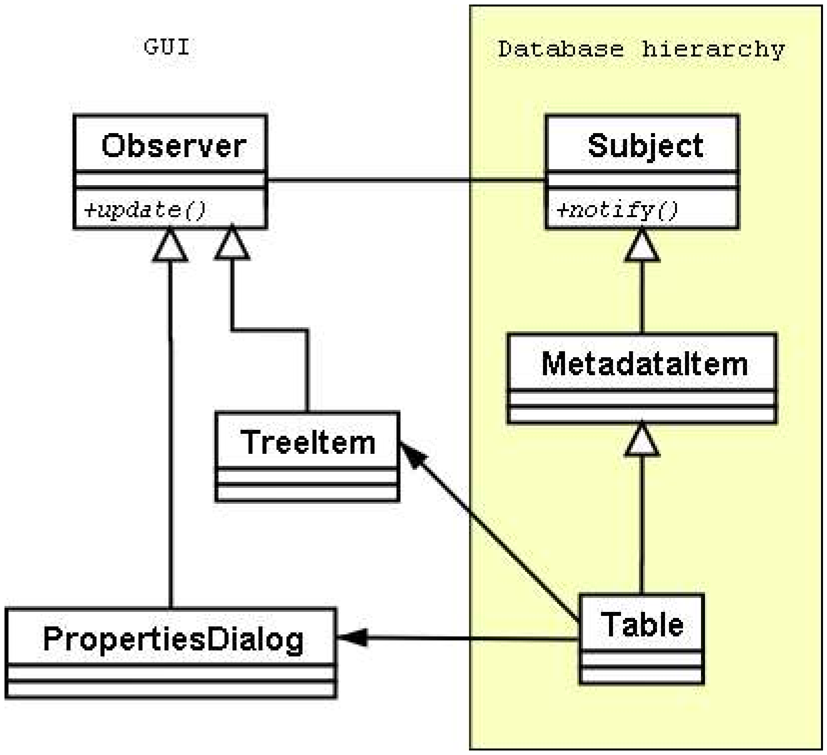
FlameRobin philosophy

From the start of the project, it was obvious that this isn't going to be a regular administration tool one may find on Windows platform. Most of the people on the project are not conservative by nature, so we usually don't decide to make something in a particular way just because every other administration tool does it like that. For example, FlameRobin uses SDI instead of MDI for window layout. Most other tools use MDI, which is quite common on Windows. However, you'll rarely see a MDI application on Linux. Then, there are the properties windows for database objects. While other tools use regular windows with various controls: buttons, options, checkboxes, grids, etc. we decided to use HTML. These are created at runtime using a set of provided template HTML files filled with placeholders where attributes of database objects are inserted. This gives great flexibility, as the user can easily modify the looks and content of those pages. For example, suppose you don't like the fact that table's constraints are on a separate page, and you'd like to see them as soon as you open the property page. Making this happen is a matter of simple copy & paste between two template files.

One of the things we are really proud of is the openness of the tool towards the user. Whichever action (s)he is taking, FlameRobin always displays the SQL statement it's going to run. All scripts are run in SQL Editor window, so one can easily copy them to clipboard or save to file. History of such statements is optionally kept in log files, which are user configurable with options to have separate log files for different databases, and also to use multiple log files – one for each statement, making it easy to build update scripts for production databases.

The Observer Pattern

Every operation that modifies the state of objects in database is instantly reflected on screen. FlameRobin keeps internal representation of state of database objects, which we call Database Object Hierarchy (DBH). DBH nodes are created upon connecting to the database, and updated when state in database changes. All this is done via Observer Pattern (one of the GoF patterns, see: http://hillside.net/patterns/DPBook/DPBook.html).



Graphical objects are observers of DBH objects. After some DDL (data definition language) statement is run and transaction is committed, the statement parser alerts the affected objects to update their state. In most cases, they reload the state from database, making it sure that the info is up-to-date and giving assertion to the user that operation really went through without problems. When DBH object changes its internal state it notifies all its observers to update themselves. Take a database table for example. Suppose we have connected and opened table's property page, and we're also viewing table's columns in tree view. We have two observers of table object in DBH. Running a statement like “ALTER TABLE x ADD column...” and committing it, starts the committed-statement parser which in turn alerts the table to update the info about it's columns. The table reloads the column info from database and then calls both observers to update themselves. The property page would reload itself, and display a new column, while the tree item would add a new sub-item to itself.

Current status

Although our plans were to come out with first beta version before the Conference, the development has been slower than expected, and we're still in alpha stage. The most recently released version is 0.4.0, which has been ported to Windows, Linux, Mac OS X and FreeBSD. Windows port works on all 32-bit Windows systems that Firebird supports: Windows 98, 2000, Me, NT, XP, 2003. Linux and FreeBSD ports are available for both 32 and 64 bit systems, although the 64-bit is not considered stable, since there is no stable 64-bit Firebird client library available yet. On these operating systems, one can choose between Gtk1 and Gtk2 version. Gtk1 is faster and has less dependencies, but it does not look so nice (no anti-aliased fonts, for example). It also uses wxWidgets replacements for some features that are otherwise native in Gtk2. Gtk1 is recommended for older systems and even on newer ones where speed is important. Mac OS X port uses wxCarbon, which uses Carbon toolkit as back-end. We are still looking for someone interested to try to build the port to Solaris. As far as we know, all tools and libraries used are already ported to it, we just need someone who has a working Solaris system to download and try to compile on it.

Currently implemented features

FlameRobin features many things one would expect from a decent database administration tool. Those "standard" features in version 0.4.0 include:

- Browsing databases
- View, edit and drop objects
- Backup and restore running in separate threads
- Powerful SQL Editor

User can browse databases in a regular tree view where nodes represent servers, databases and database objects. Each object can be manipulated via context menu or regular application menu. The tool is highly configurable to fit the users preferences.

SQL editor features many things one would want in such a tool. It provides syntax highlighting for keywords and database object names and auto-completion of those. It also features "call-tips" which show input and output parameters for user defined functions (UDF) and stored procedures. SQL statements can be entered, loaded, saved and executed. Also, a history of statements is kept persistently between sessions. A basic transaction control is present with commit and rollback options. Option to prepare the query and view the execution plan without executing it also comes in handy. One of the unusual features is the ability to select any object's name in editor and get properties page for that database object.

Some uncommon features

There are some specific features in FlameRobin which are rarely (if at all) found in other administration tools for Firebird:

Event monitor

Event monitor is a simple user interface to Firebird's events feature. It allows user to subscribe to one or multiple events, and even load the list of events to subscribe to from a textual file. Once the monitoring starts, the log window displays the timestamp and count of each event that got posted from the database.

Drag and drop query building

Drag & drop query building is a feature that allows the user to build SELECT queries without typing anything in SQL editor. Just open the SQL editor window and drag and drop table columns (or entire tables) onto it. It detects table relations by reading the foreign keys information and creates the appropriate JOIN clauses. If there are more ways to connect tables, it gives a short list and lets user pick the one (s)he needs. Currently this feature is very basic, but already usable.

“Connect as...” option

This option allows the user to use the different credentials when connecting to the database. Different from those typed in when registering the database. Suppose you sometimes need to log in as a different user, or role. Or you simply wish to log in using different character set. Instead of editing the database registration info, and reverting it back later, you can use this feature to temporarily log in with different parameters without changing anything.

FlameRobin future

So fat a lot of work has be done, and tool is very usable. In fact, some of us use it almost exclusively for our Firebird related work. However, there are still a lot of features that need to be implemented before it could be considered complete. Here are some of the important features planned for version 1.0:

- DDL extraction for all object types
- User management and granting privileges to objects
- Editable data grid
- Field editors for specific data types
- Better support for international (non-ASCII) characters
- Multi-threaded query execution

For more details, and up to date information about future goals, please review our Road-map page at:

http://www.flamerobin.org/dokuwiki/doku.php?id=wiki:roadmap

DDL extraction for all object types

This features allows the user to retrieve SQL script that could be run to create the object if it didn't exist. The script includes no only the appropriate CREATE statement, but also setting up privileges, constraints and relations to other objects (table foreign keys for example). Running it for all objects would give the SQL script to create entire database structure from scratch, similar to command "isql -a".

At later stages, we might also provide the so-called recreate scripts, which would drop the relations to existing objects and recreate them after original object is changed. It would be useful to cascading dependencies between tables, views, triggers and stored procedures, where altering a view is usually very error-prone job when using plain SQL.

User management and granting privileges to objects

It has two components. One is user management on database server, with ability to add and remove Firebird users, and also alter their passwords and data. This functionality will be implemented via Services API. The other component is grant manager, which is used to grant and revoke privileges on database objects, and also to grant roles to users.

At later stages, we might also implement a cascading grant feature. It is needed when, for example, user has INSERT privilege on a table, but there is also a trigger on that table which uses some other objects. Cascading grant would grant the user all the privileges required by the trigger at the time INSERT privilege on table is granted.

Editable data grid

Currently, the data grid in SQL Editor is read-only. We wish to implement a feature similar to one seen in Delphi's database grid components, where user can modify existing data inside the grid and also add and remove rows from the grid. When changes are made, it would run INSERT, UPDATE and DELETE statements in the background. Later we might implement special editors for blobs, arrays and timestamps, and also allow selection of data from other tables when columns with foreign keys are filled in.

Better support for international (non-ASCII) characters

FlameRobin already uses Unicode internally in some of its variants. All the strings in FlameRobin itself are translatable, and show properly. However, the data from databases is not represented correctly since the required conversion from connection's character set to Unicode is not done. We plan to alter our database access layer to make this work properly.

Multi-threaded query execution

In current version of FlameRobin, all the queries are run synchronously, i.e. the entire application is blocked until the query is executed. The plan it to make possible for queries to run in separate threads, leaving the GUI free for user to do other useful stuff while long-running query is running or a complex/heavy query being prepared. This is not such a simple task as Firebird's client library is not thread safe on the same connection, so each thread must start its own connection. This creates potential problems with foreign keys, as they can only be added when there are no other connections to the database (at least in current stable Firebird versions). Besides the query execution itself, we might make the connecting itself happening in a separate thread. It happens from occasionally that you try to connect to unavailable server (no network connection) and then you have to wait for connection timeout to pass before you can get the control back. It can be very frustrating, so it may be a worthwhile feature.

Useful links and contact information

Project's website

http://www.flamerobin.org

Souceforge.net project's page, feature requests and bug tracker

http://sourceforge.net/projects/flamerobin

http://sourceforge.net/tracker/?atid=699237&group_id=124340&func=browse

http://sourceforge.net/tracker/?atid=699234&group_id=124340&func=browse

Mailing list for FlameRobin users and developers

http://lists.sourceforge.net/lists/listinfo/flamerobin-devel

Libraries used to build FlameRobin

http://www.wxwidgets.org

http://www.ibpp.org

Author

mbabuskov@users.sourceforge.net

mbabuskov@yahoo.com

Table of Contents