MySQL® RoadMap

What we have now & Where we are heading

Arjen Lentz, MySQL AB

I will first outline the current key features of the MySQL RDBMS, supported by a little history and an overview of the MySQL development and release philosophy. We then go on to look at what new funky stuff the MySQL development team is working on. There will also be time for open questions.

About MySQL

MySQL is a relational database management system (RDBMS), which can be used in either the familiar client/server architecture, or as an embedded library. Internally, the MySQL server is multi-threaded and is written in C/C++. The lower layers of the MySQL code have history going back to the early 80s, with the SQL layer added around 1995. MySQL runs on pretty much any platform for which the the gcc compiler toolset is available, as well as a few others. This of course includes all the regular Unix variants, plus native Microsoft Windows (no Cygwin), and Novell NetWare.

On the client side, the primary API is written in C (for greater portability), and most application and scripting languages use this library under the hood. The MySQL client/server protocol is, of course, public and there are secondary implementations of the client API. Interesting examples are the native Java driver (Connector/J) and a .Net driver written in C#. Surely someone with too much time on their hands will implement it in a plain shell script, too...

MySQL AB

MySQL AB is the company behind the MySQL. The “AB” stands for “aktiebolag” which is simply “Pty Ltd” in Swedish. MySQL AB was founded by MySQL's principal author Michael “Monty” Widenius, together with David Axmark and Allan Larsson. Over the last few years there have been two venture capital funding rounds, primarily to support accelerated growth and speed up development. Our biggest investor is Benchmark Capital (Red Hat, eBay).

While MySQL AB has local subsidiaries and offices in Sweden, Finland, Germany and the USA, most employees actually work from home. The big advantage of this model is that, apart from lowering the operational overhead, it allows us to seek out talent anywhere on the planet. Near the end of 2003, there were over 100 employees worldwide, spread across more than a dozen countries. There are currently two employees in Australia.

MySQL AB makes its money selling commercial licenses, support (from e-mail to 24/7 phone support, by developers) and professional services such as consultancy, training and certification.

Dual Licensing and Warranty

The MySQL server, tools and libraries are all released under the GNU General Public License. This is a vital aspect of our business model, and all investors have to explicitly underwrite this commitment. Free and Open Source software is better: there is public scrutiny of the code, third party security audits, feedback throughout the development process, and battle-testing in many different environments. What other database company has a quality assurance department with millions of different installations of its software?

As noted, MySQL AB also sells commercial licenses. We can do this because we own all the code and documentation, plus the name and the logo. That is rare, as the intellectual properly for most

open source software projects is owned by a meriad of people (like the Linux kernel). But there is other software that also uses the dual licensing concept, for instance BerkeleyDB from Sleepycat Software. It is not suitable for all software, but in the right market it provides a solid basis for a sustainable business model.

The GNU General Public License is often regarded as confusing, but the fundamentals are actually quite simple. The GPL operates within the scope of regular copyright law. The original author of a work (automatically) has the sole right to copy his/her work. This applies to programming code just like it does to a book. The owner may of course sub-license their right to copy/distribute. So without permission from the owner, you're not allowed to copy it. Similarly, the GPL merely talks about distribution (copying), not use. If you legally acquire GPL licensed software, the GPL will be of no further concern to you unless you wish to distribute it further, either on its own or as part of a larger package. The GPL stipulates that all parts of such a package must also be GPL licensed. So the GPL basically gives you additional freedom, with certain conditions. If someone finds these conditions not to their liking, they are still bound by regular copyright law (i.e. they are not allowed to distribute/copy without explicit permission from the owner).

In practise, most people can use MySQL under the GPL license. For some purposes, the GPL is not suitable. This typically happens with embedded applications, where (parts of) the MySQL code is bundled with proprietary software, and then sold (distributed). A company wishing to do that needs to acquire a license from the owner, MySQL AB. The code in the commercially licensed MySQL software is almost identical to the GPL version, it is only the license that is different. Licenses are cheap (only a few hundred dollars for a single license, and of course there is specially negotiated pricing for higher volumes).

Another very important aspect of licensing is warranty. GPL provides none: “if you break it, you own all the pieces.” If a company requires warranty, then a commercial MySQL license is the way to go: they can effectively indemnify themselves against liability in case anything goes wrong with the MySQL component of their package. “Regular” open source software cannot offer this.

Development and Release Philosophy

MySQL is developed using multiple parallel development trees, currently the following:

Series	***Status***
3.23.x	Stable, old (only critical fixes)
4.0.x	Stable, current production (only bugfixes)
4.1.x	Alpha, binaries available
5.0.x	Development, sources only

Development follows various stages. There can be multiple releases (binary builds) within a single stage. Releases are indicated by the number after the series (e.g., version 4.0.16 is the 16th release in the 4.0 series). New features are only added in the *development* and *alpha* stage. From the *beta* stage there is a feature freeze, allowing the code to “settle”. Generally, there will be a few binary releases during this time, so that a wide audience can test any fixes. If no new critical bugs are found for a while, a version moves into the *gamma* stage. You could compare the gamma stage to a “release candidate”. If no major problems show up, then the series can be declared *stable* (production).

We aim to make each binary release free from all known -reproducible- bugs. Currently, while the 4.0 series still sees regular releases with bugfixes, the developers are busy in the 4.1 and 5.0 trees

working on new features. Applicable changes are also merged into higher series. The 5.0 tree will move into the alpha stage (with binary builds) when the 4.1 tree goes beta. This shift may already have happened by now.

So how do you decide which series to use? We generally recommend trying the latest alpha series for new application development. This allows you to try out all new features. Often, a series has either already reached the production stage when an application is ready to be deployed, or you will be so familiar with the specific MySQL series that you know it will run solidly with your application. The stage tag says nothing about actual *instability*, or suitability for your specific application. Only you can make that decision. Do keep the MySQL release system in mind when asking a question or reporting a problem. Simply referring to "the latest version" is clearly not enough ;-)

Existing and Upcoming Features

As 4.0 is current production series, we will use this as the basis for the feature overview. MySQL supports ACID transactions, foreign keys and row-level locking with an "Oracle style" table space using multi-versioning concurrency control (MVCC). There is no need for "vacuum". The unique *storage engine* concept allows a developer to select the format most appropriate for each table. Currently there are transactional (high concurrency), non-transaction (smaller, ideal for some logging operations), as well as HEAP (in-memory) table types. There is support for >4GB tables and BLOBs. Large datasets (into the terabyte realm) are operational in production environments. Other key features are asynchronous replication, full-text indexing, query caching, multi-table deletes and updates, and the dynamic setting of server variables.

When implementing new features, we aim towards SQL standards compliance. Version 4.1 adds a range of major new features such as sub-queries, Unicode (UCS2 and UTF8) character sets configurable down to the column level, OpenGIS spatial extensions, prepared statements, INSERT ... ON DUPLICATE KEY UPDATE ..., GROUP BY ... WITH ROLLUP, pre-loading of indexes, client/server connections using SSL, warnings and standard error codes, and online help in the MySQL command-line client.

In the 5.0 development tree, the main focus is on the implementation of stored procedures (Persistent Stored Modules, as SQL99 calls it). If you download the 5.0 source tree, you can see them working. The SQL99 syntax is similar -but not identical- to Oracle's PL/SQL. We are also working on a new recursive descent parser which will replace the old Bison-based one and provide more flexibility with fewer reserved words. Views and server-side cursors are also being worked on, and will be added into either 5.0 or 5.1. Triggers are scheduled for 5.1.

Parallel to the development of the server, we are working on better (graphical) tools for installation, administration and development. During my talk, I will elaborate further on individual features mentioned above, depending on the specific interests of the audience and the latest news at that time.

The primary goals of MySQL are speed, stability and easy of use. A default installation of MySQL has a minimal footprint on disk and in memory, and is therefore geared to allow you to start "playing" quickly. Naturally, you will want to tune various buffers and other settings for optimal performance with your specific applications in the target environment. Interestingly, we find that many production environments have not been tuned at all, and MySQL still runs surprisingly well with the minimalistic defaults.

MySQL does not aim to implement every possible feature, or clone another database system. The vast majority of applications (say 95%) use only a tiny subset of the vast feature set encompassed by modern RDBMSs. MySQL operates in this *commoditised* segment of the market. We are quite happy to leave the last few percent to others, particularly since we are in a huge (and growing!) market anyway. If you don't use a particular feature in a new version of MySQL, you should see no speed decrease compared to earlier versions. We always try to find the most optimal way to implement something. This is why, sometimes, things takes a while. It is very important to prevent "bloating", and we keep these objectives in mind from the first design stage of a feature.

Important Developments

In June 2003, MySQL AB acquired full commercial rights to SAPdb (now named "MaxDB by MySQL") in a cross-licensing deal with SAP. MaxDB is an enterprise grade database system, which SAP already published as open source. The deal gives us access to a lot of expertise. We will leverage these new resources to develop the next generation of our software, let us call it "MySQL Enterprise". As MaxDB was written in a Pascal dialect, the two servers cannot be merged and our existing MySQL code base will provide the foundation for adding the necessary features for further SQL standards compliance.

Also in 2003, MySQL AB acquired Alzato from Ericsson Business Innovation. Alzato developed NDB Cluster, a clustered database engine designed for real-time telco use. The Alzato team is now part of MySQL AB, and working on integrating the NDB Cluster technology with MySQL.

So where are we heading?

We are sticking close the goals that we set for ourselves 3 years ago, making MySQL:

- The best and the most used database in the world
- Available and affordable for all
- Easy to use
- Continuously improved while remaining fast and safe
- Fun to use and improve
- Free from bugs

By the way, we are working to support SAP R/3. Certainly, a bold statement. But it is good to have ambitious plans, and now that you know more about who and where we are, you may appreciate that our goals are quite attainable. As mentioned early on, the open source model and the MySQL community play vital roles. The commercial aspects help the community, and vice versa. Our model requires both. Thank you for your continued support.

Arjen Lentz (arjen@mysql.com), December 2003