

## MySQL: the Company, the Product

Let's begin with how to pronounce it. The American National Standards Institute (ANSI) wants SQL pronounced *ess-kew-ell*. The International Standards Organisation (ISO) takes no position on pronunciation. Many database professionals and most Microsoft SQL Server developers say *see-kwel*. MySQL founders prefer *my-ess-kew-ell*. We think they get the last word on the matter.

In a way, MySQL<sup>®</sup> the product created MySQL AB the company. Michael “Monty” Widenius and David Axmark formed it as a Swedish open source company in 1995. It remained entirely open source through 2006, turning annual profits from 1996 through 2007 with a business model based more on support, consulting and training services than on product licencing. In 2001 MySQL added Heikki Tuuri's [InnoDB](#) transactional engine. Oracle bought InnoDB in 2005. Sun Microsystems bought MySQL AB in 2008, and Monty soon left to form his own company. Oracle bought Sun in 2009, promising to continue to develop and support MySQL. Oracle's first moves have been to improve MySQL performance and integrate the MySQL and InnoDB development teams.

The MySQL product remains open source, but Oracle's acquisition of MySQL leads many to question the future of MySQL. EU regulators extracted written pledges from Oracle to “continue to enhance” MySQL for five years. What do these pledges mean? It seems unlikely, for example, that Oracle would develop MySQL in a direction that would make it more competitive with Oracle's other relational database products.

### Beginnings

The story begins in 1979, in Helsinki, with Widenius and Allan Larsson, owners of a consulting firm called TcX. In that year Monty wrote his own in-house database tool called UNIREG. Based on *curses* (a Unix program), UNIREG made it easy to search, update and report from databases. This was just one year after IBM had begun testing its first relational database at customer sites.

Monty wrote UNIREG in BASIC on a Swedish computer called an ABC800, with a whopping 32K of RAM and a 4.0 MHz Z80 processor. Two years later, Monty rewrote it in C on a much more powerful computer, a Swedish DS90 (16 MHz MC68010) with 4MB RAM, running an operating system called Denix, a Swedish variant of Unix.

In 1983 Monty met David Axmark, who worked at Detron HB. They have worked closely together since then, traded code and ideas, often late into the night on the phone.

From 1985 through 1994, TcX moved into data warehousing, ported UNIREG to more powerful computers (mainly Suns), and extended its power to handle huge databases.

As software goes, UNIREG has had a remarkably long life. Monty has never changed the original user interface, and its users are still easily moving reports and data from any UNIREG version to the current one.

In 1994, TcX began developing web-based applications, still relying on UNIREG. Unfortunately, each time a user hit a dynamic page, another copy of UNIREG was loaded. Under a heavy load, this was unacceptable. Meanwhile, SQL as a database language was by now a standard. TcX decided to start using SQL as the interface to UNIREG.

They tested commercial SQL servers but found large table performance unacceptable.

*Databases are often described in two dimensions: largeness and richness. Large databases have many rows in a relatively few tables. Rich databases have many tables with relatively few rows.*

*The terms are imprecise and relative to experience. Our convention is that a rich database has more than 300 tables, and a large database has millions of rows. The first MySQL client's database was both large and rich—60,000 tables, 5 billion rows.*

Monty downloaded a copy of mSQL, a relational database written by David Hughes. Monty found mSQL wanting in several critical areas, and asked Hughes if he would be interested in connecting mSQL to the UNIREG B+ ISAM table handler. David declined, and TcX decided to write their own SQL server, making it compatible with the UNIREG format and meeting their performance requirements.

A year later Monty rolled out MySQL 1.0. David was an enthusiastic supporter of the open source movement, and almost immediately began to pressure TcX to release MySQL into the wild, making it freely downloadable on the Internet. He also wanted a much freer copyright policy than mSQL, allowing anyone to use it commercially so long as they didn't distribute it. They chose Solaris and Linux as major platforms, and Monty began porting the code to Linux.

By making the MySQL API almost identical to that of mSQL, Monty made it trivial to port the rapidly increasing number of free mSQL clients to MySQL. Something of a groundswell emerged, and many developers began to contribute new code.

*Threads permit a program to perform more than one task at once. In truth, a single CPU can perform only one task at a time, but threads allow a developer to create the illusion of multi-tasking. Each thread is its own task or sequence of tasks; the CPU distributes slices of time to each thread in rapid succession, much like a casino blackjack dealer passing cards to each player in turn.*

One problem remained—threading. Monty had designed MySQL using a Posix-compliant library developed by Chris Provenzano, which ran on numerous platforms. Chris no longer had time to support his product, so Monty took it over. He fixed a few bugs and added some missing Posix features that MySQL needed. In January, 1997, the pthreads code was included in the MySQL source distribution, paving the way for it to be ported to numerous platforms.

In 2001, Mårten Mickos joined MySQL AB as CEO. Mårten and Monty have known each other more than twenty years. They studied together, and kept in close contact since then, but he was not involved in MySQL work until the middle of 2001.

## The product

MySQL is published under the GNU General Public Licence (GPL), so it is free to [download](#) and use for databases of any size if you use it in-house, or if you distribute it unembedded and only with other open-source software.

As Monty [describes](#) it, MySQL entered the database market as a “low-end market disruptor”, meeting demand for low-cost web databases, then moved upmarket. It is far and away the most popular open source database. It runs on dozens of platforms and has APIs to every important programming language including C, C++, Java, Perl, PHP, Python, Eiffel and Microsoft .NET languages. A *Join Vision E-Services GmbH* 2006 survey found 29% of “IT experts” using MySQL, 24% using SQL Server/Access, 23% using Oracle and 10% using DB2. In 2005-2007 MySQL market share grew 25%.

Table 2-1 shows the six layers of MySQL, two of which set it apart: open source APIs for *multiple programming languages* and *multiple database storage engines*. There are freely downloadable community and commercial MySQL builds for several operating environments including many flavours of Linux, Windows, Sparc/Solaris, IBM AIX, Mac OS X and FreeBSD. For how to install MySQL, see the [next chapter](#). You can also download [free GUI tools](#): MySQL Administrator, Query Browser, Database Workbench for database design, and the Migration Toolkit for database migration.

**Table 2-1: MySQL Layers**

<b>Layer</b>	<b>Description</b>
<i>Client programs</i>	MySQL client, Administrator, Query Browser; <a href="#">utilities</a>
<i>Programming language connectors</i>	C, JDBC, ODBC, .NET, PHP, Perl, Ruby, Cobol
<i>Connection pool</i>	Connection management
<i>Kernel server modules</i>	Process management, SQL interface, parser, optimiser, caching and buffering
<i>Pluggable database storage engines</i>	ARCHIVE, CSV, FEDERATED, INNODB, MEMORY, MYISAM, MERGE, others
Operating system interface	Management of files and logs.

## Licence

The main points of MySQL licencing are:

- You must pay for MySQL only if you are selling MySQL directly, or selling a product which is not open source and includes the server (you may not include MySQL in a distribution if you charge for some part of it).
- MySQL client code is in the Public Domain or under the General Public Licence.
- The company may restrict some functionality to commercial editions of MySQL.
- The company "encourages" users to buy licences or support.

## Features

[Chapter 4](#) describes MySQL data types, [Chapter 6](#) and [Chapter 8](#) cover MySQL's version of the SQL language, [Chapter 7](#) describes MySQL database engines, [Chapter 9](#) covers query building with MySQL, chapters 10 through 16 cover MySQL connectors, [Chapter 18](#) covers MySQL administrative utilities, [Appendix B](#) covers MySQL administration and control variables, and [Appendix C](#) describes MySQL error handling.

The first client to use MySQL was a firm that used it to store interviews. Their database grew to 60,000 tables comprising 5 billion rows because this client used it to store interview questionnaires, and the system created tables for each new questionnaire. From the beginning, then, MySQL was designed to handle large numbers of tables with large numbers of rows.

Your view of the architecture of MySQL will be tinted by your familiarity with the other major databases—Oracle, DB/2 and SQL Server. If you come from such a universe, you may miss some functionality you expect to have. On the other hand, compare licencing and support costs; if you can do without a shrinking list of big-iron features, you can save a substantial amount of money. If your universe consists of web site design, and you're responsible for dozens or hundreds of pages, you may find the power delivered by MySQL astounding.

Compared with the Big Three, MySQL does a lot of things differently. Its developers have not been slaves to the ANSI standard. They follow the standard in many crucial respects, but remain free to add whatever extensions to the standard language that they feel they need. The Big Three do not release their source code. Their business model is based on proprietary code. By contrast, MySQL AB makes its money through training, consulting and programming, more than from selling the product.

The architecture extends these privileges to you, the developer. You are free to create new functionality in several ways:

- *Write Stored Procedures and Functions:* As of MySQL 5, you can write your own server-side procedures and functions.
- *Write plugins* and install them via the [plugin interface](#).
- *Write original MySQL code:* At the most fundamental level, you can download the source code and, if you're a serious developer, fix any bugs you find, or code your own function server functionality.
- *Code through an interface:* MySQL is designed to permit other programs to communicate directly with it. Several application program interfaces (APIs) are available for MySQL, so programmers can interact with it from languages such as Java, C, PHP, Python, Microsoft Access and others. These interfaces are in no small part responsible for the tremendous success of MySQL, and are covered in Part III, Chapters 10 through 16.