MySQL: Past, Present, Future

It's named after Michael "Monty" Widenius's first daughter My. How to pronounce it? The American National Standards Institute (ANSI) wants SQL pronounced *ess-kew-ell*. The International Standards Organisation (ISO) says nothing about it. Many database professionals and Microsoft SQL Server developers say *see-kwel*. The owners of MySQL haven't stated a preference. MySQL founders preferred *my-ess-kew-ell*. We follow them.

In a way, MySQL® the product created MySQL AB the company. Monty and his partner David Axmark formed it in 1995 as a Swedish open source company. It stayed entirely open source through 2006, turning profits from 1996 through 2007 on a business model based more on support, consulting and training than on product licencing. In 2001 they added Heikki Tuuri's *InnoDB* transactional engine.

Oracle bought InnoDB in 2005. The next year, it tried to buy MySQL. In 2008 Sun Microsystems did buy MySQL. Monty soon left to form his own company. Oracle bought Sun in 2009. Its focus since has been to improve MySQL performance and scaling, to grow sales of the value-added Enterprise Edition, and lately to market an online MySQL database service in competition with the likes of Amazon and SAP.

MySQL *ranks* a shade behind Oracle as the world's most popular database engine, well ahead of Microsoft SQL Server. In this tabulation, SQL Server and Oracle have been losing ground while MySQL's score has grown slowly. Oracle's purchase of MySQL was widely unpopular. Aganst expectations, Oracle stewardship has left MySQL bigger, more SQL-compliant, more robust, more reliable. But there are issues: enhancements have come slowly, Oracle's efforts to make money from MySQL Enterprise have closed some source, its GUI utility MySQL Workbench is less than entirely reliable, and MySQL community support fora are too often overrun by spam.

With his new company, Monty created a drop-in replacement for Oracle MySQL—MariaDB, named after another daughter. It's robust, has impressive users *e.g,*. Google and Wikipedia, but though many internet service providers offer MariaDB, its market share does not begin to challenge MySQL. More recent versions have diverged from drop-in replaceability with a plethora of added features.

Beginnings

The story begins in 1979, in Helsinki, with Monty 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 worked closely together, 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 had a remarkably long life.

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.

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.

The product

MySQL Community Edition is published under the GNU General Public Licence (GPL), so it's free to *download* and use for databases of any size if you use it in-house, or if you distribute it unembedded and with no closed-source software. Version 8.0 discontinued the embedded version.

As Monty *says*, MySQL entered the database market as a "low-end market disruptor", meeting demand for low-cost web databases, then moved upmarket. For years it has been the world's most popular open source relational database. It runs on dozens of platforms, with APIs to many important programming languages including C, C++, C#, Java, Javascipt, Perl, PHP, Python, Ruby, Eiffel. A 2020 survey by Datanyze found MySQL holding an overall lead in domain database market share. The *2022 survey* (Table 2-1) shows Microsoft SQL Server slightly ahead of MySQL, those two well ahead of all others, PostgreSQL and MongoDB market shares growing, and MariaDB far down the list.

Table 2-1: Database domains

Rank	*DBMS*	*Domains*	*Pct*
1	Microsoft SQL Server	34,607	15.73
2	MySQL	33,686	15.31
3	Microsoft Access	24,577	11.17
4	PostgreSQL	18,250	8.30
5	MongoDB	12,483	5.67
6	NoSQL	12,094	5.50
7	Oracle RDBMS	8,792	4.00
8	Oracle Database	5,802	2.54
9	Redis	4,716	2.14
10	Sybase	4,308	1.96
…			
28	MariaDB	1,148	0.52

Table 2-2 shows MySQL's six layers, 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 incuding many flavours of Linux, Windows, Sparc/ Solaris, IBM AIX, Mac OS X and FreeBSD. For how to install MySQL and its tools including the GUI Workbench for server administration, data modelling, database migration, and management of data and queries, see the *next chapter*.

Table 2-2: MySQL Layers

Layer	*Description*
Client programs	MySQL client, Workbench, utilities
Programming language connectors	C, JDBC, ODBC, .NET, PHP, Perl, Ruby, Cobol
Connection pool	Connection management
Kernel server modules	Process management, SQL interface, parser, optimiser, caching and buffering
Pluggable database storage engines	ARCHIVE, CSV, FEDERATED, INNODB, MEMORY, MYISAM, MERGE, others
Operating system interface	Management of files and logs.

Licence

The main points of MySQL Community Edition 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.
- Some functionality may be restricted to commercial editions.

- Users are encouraged 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 with 5 billion rows because this client created new 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 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 is website design and you're responsible for dozens or hundreds of pages, you may be astounded by MySQL's power.

Like Oracle and Microsoft, MySQL AB skipped some SQL standards and implemented a few non-standard SQL features of its own. The open-source database model has long been gaining world market share against the commercial proprietary code model; in 2021 it drew even. Of course Oracle Corp. has tried to grow the MySQL Enterprise market from the success of the Community Edition. As a MySQL user you are still free to create new MySQL 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 bugs you find, or code the additional server functionality you desire.
- *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, Visual Studio and others. Such interfaces are in no small way responsible for the tremendous success of MySQL, and are covered in Part III, Chapters 10 through 16.