1 MySQL

It was not too long ago that you were simply out of luck if you wanted to build an application backed by a robust database on a small budget. Your choices were quite simple: shell out thousands—or even tens or hundreds of thousands—of dollars on Oracle or Sybase or build the application against "toy" databases like Access and FileMakerPro. Thanks to databases like mSQL, ProgreSQL, and MySQL, however, you now have a variety of choices that suit different needs. This book, of course, is the story of MySQL.

The History of MySQL

This story actually goes back to 1979 when MySQL's inventor, Michael Widenius (a.k.a. Monty) developed an in-house database tool called UNIREG for managing databases. UNIREG is a tty interface builder that uses a low-level connection to an ISAM storage with indexing. In the years that have followed, UNIREG has been rewritten in several different languages and extended to handle big databases. It is still available today, but it is largely supplanted by MySQL.

The Swedish company TcX* began developing Web-based applications in 1994 and used UNIREG to support this effort. Unfortunately, UNIREG created too much overhead to be successful in dynamically generating Web pages. TcX thus began looking at alternatives.

TcX looked at SQL and mSQL. mSQL was a cheaply available database management system that gave away its source code with database licenses—almost Open Source. At the time, mSQL was still in its 1.x releases and had even fewer features

* For most of its existence, TcX had a single employee: Monty.

than the currently available version. Most important to Monty, it did not support any indices. mSQL's performance was therefore poor in comparison to UNIREG.

Monty contacted David Hughes, the author of mSQL, to see if Hughes would be interested in connecting mSQL to UNIREG's B+ ISAM handler to provide indexing to mSQL. Hughes was already well on his way to mSQL 2, however, and had his indexing infrastructure in place. TcX decided to create a database server that was more compatible with its requirements.

TcX was smart enough not to try to reinvent the wheel. It built upon UNIREG and capitalized on the growing number of third-party mSQL utilities by writing an API into its system that was, at least initially, practically identical to the mSQL API. As a result, an mSQL user who wanted to move to TcX's more feature-rich database server would only have to make trivial changes to any existing code. The code supporting this new database, however, was completely original.

By May 1995, TcX had a database that met its internal needs—MySQL 1.0. A business partner, David Axmark at Detron HB, began pressing TcX to release this server on the Internet and follow a business model pioneered by Aladdin's L. Peter Deutsch. Specifically, this business model enabled TcX developers to work on projects of their own choosing and release the results as free software. Commercial support for the software generated enough income to create a comfortable lifestyle. The result is a very flexible copyright that makes MySQL "more free" than mSQL. Eventually, Monty released MySQL under the GPL so that MySQL is now "free as in speech" and "free as in beer".

As for the name MySQL, Monty says, "It is not perfectly clear where the name MySQL derives from. TcX's base directory and a large amount of their libraries and tools have had the prefix 'my' for well over 10 years. However, my daughter (some years younger) is also named My. So which of the two gave its name to MySQL is still a mystery."

A few years ago, TcX evolved into the company MySQL AB at *http://www.mysql.com*. This change better enables its commercial control of the development and support of MySQL. MySQL AB is a Swedish company run by MySQL's core developers. MySQL AB owns the copyright to MySQL as well as the trademark "MySQL". Since the initial Internet release of MySQL, it has been ported to a host of Unix operating systems (including Linux, FreeBSD, and Mac OS X), Win32, and OS/2. MySQL AB estimates that MySQL runs on about 500,000 severs.

MySQL Design

Working from the legacy of mSQL, TcX decided MySQL had to be at least as fast as mSQL with a much greater feature set. At that time, mSQL defined database per-

www.mysql.com vs. www.mysql.org

As this book is being written, MySQL AB has no control over the mysql.org domain. Until June 2001, mysql.org was owned by someone else who had simply pointed it to the mysql.com domain. On June 4, 2001, a company called NuSphere bought the mysql.org domain and put up a different site. MySQL AB requested the domain be transferred to them, but NuSphere refused.

MySQL AB claims that NuSphere is distributing non-GPL, proprietary code with MySQL in violation of MySQL's GPL license in addition to hijacking the MySQL trademark. NuSphere, on the other hand, claims it is not violating the GPL and that MySQL signed a contract to allow NuSphere to use the MySQL trademark. Both sides have sued each other, and the matter is now before the courts.

Regardless of where your sympathies lie on the issue or the eventual outcome, the official source of MySQL is MySQL AB at *www.mysql.com*.

formance, so TcX's goal was no small task. MySQL's specific design goals were speed, robustness, and ease of use. To get this sort of performance, TcX decided to make MySQL a multithreaded database engine. A multithreaded application performs many tasks at the same time just as if multiple instances of that application were running simultaneously. Fortunately, multithreaded applications do not pay the very expensive cost of starting up new processes.

In being multithreaded, MySQL has many advantages. A separate thread handles each incoming connection with an extra thread always running in order to manage the connections. Multiple clients can perform read operations simultaneously without impacting one another. Write operations, on the other hand, only hold up other clients that need access to the tables being updated. While any thread is writing to a table, all other threads requesting access to that table simply wait until the table is free. Your client can perform any allowed operation without concern for other concurrent connections. The connection-managing thread prevents other threads from reading or writing to a table in the middle of an update.

Figure 1-1 illustrates the multithreaded nature of a MySQL database server. Another advantage of this architecture is inherent to all multithreaded applications: even though the threads share the same process space, they execute individually. Because of this separation, multiprocessor machines can spread the load of each of the threads across the many CPUs.

In addition to the performance gains introduced by multithreading, MySQL has a richer subset of SQL than mSQL. MySQL supports over a dozen data types and additionally supports SQL functions. Your application can access these functions through ANSI SQL statements.

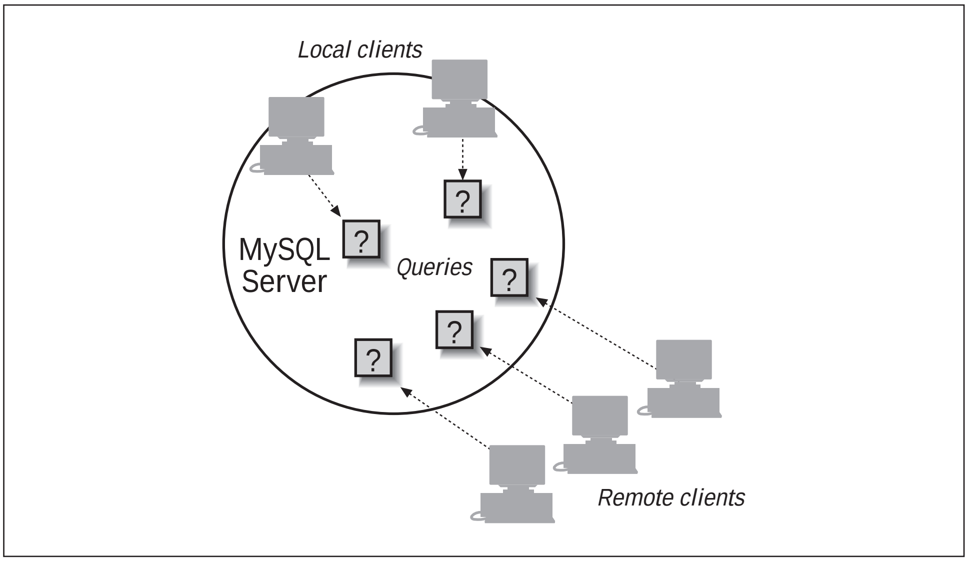

Figure 1-1. The client/sever design of MySQL

The Great Transaction Debate

Transactions are a relatively new feature of the MySQL database engine. It is, in fact, a feature that is not present unless you set up your tables to support it. A lot of people wondered for a long time what use MySQL was without transactions and why they would set up tables without support for transactions. The answer is one word: performance.

The minute you introduce transactions into the picture, the database takes a performance hit. Transactions add the overhead of complex locking and transaction logging. The complex locking includes support for something called transaction isolation levels. We will discuss transaction isolation levels in Chapter 9. Basically, however, increasing transaction isolation levels require an increasing amount of work by the database to support the same functionality. The more work the database has to do for a task, the slower it performs that task.

Transactions are important for applications that support complex rules for the concurrent updating data. They prevent concurrent updates from leaving the database in an inconsistent state at any point in its life. The most common use for MySQL, on the other hand, is to drive dynamic Web content. This use is "heavy read" in that 80% or more of the work is reading from the database. In such an environment, support for transactions is generally not needed. Performance, however, is.

MySQL actually extends ANSI SQL with a few features. These features include new functions (`ENCRYPT`, `WEEKDAY`, `IF`, and others), the ability to increment fields (`AUTO_INCREMENT` and `LAST_INSERT_ID`), and case sensitivity.

MySQL did intentionally omit some SQL features found in the major database engines. For the longest time, transactions support was the most notable omission. The latest releases of MySQL, however, now provide some support for transactions. Stored procedures, another notable omission, are scheduled for a future release. Finally, MySQL does not support most additions to the SQL standard as of SQL3. The most important SQL3 feature missing from MySQL is support for object-oriented data types.

Since 1996, MySQL AB has been using MySQL internally in an environment with more than 40 databases containing 10,000 tables. Of these 10,000 tables, more than 500 contain over seven million records—about 100 GB of data.

MySQL Features

We have already mentioned multithreading as a key feature to support MySQL's performance design goals. It is the core feature around which MySQL is built. Other features include:

Openness
: MySQL is open in every sense of the term. Its SQL dialect uses ANSI SQL2 as its foundation. The database engine runs on countless platforms, including Windows 2000, Mac OS X, Linux, FreeBSD, and Solaris. If no binary is available for your platform, you have access to the source to compile to that platform.

Application support
: MySQL has an API for just about any programming language. Specifically, you can write database applications that access MySQL in C, C++, Eiffel, Java, Perl, PHP, Python, and Tcl. In this book, we cover C, C++, Java, Perl, and PHP.

Cross-database joins
: You can construct MySQL queries that can join tables from different databases.

Outer join support
: MySQL supports both left and right outer joins using both ANSI and ODBC syntax.

Internationalization
: MySQL supports several different character sets, including ISO-8859-1, Big5, and Shift-JIS. It also supports sorting for different character sets and can be customized easily. Error messages can also be provided in different languages.

Above all else, MySQL is cheap and it's fast. Other features of MySQL may attract you, but you are probably looking in the wrong direction if cost and performance are not issues for you. The other relational databases fall into two categories:

- Other low-cost database engines like mSQL, PostgresSQL, and InstantDB.
- Commercial vendors like Oracle, Microsoft, and Sybase.

MySQL compares well with other free database engines. It blows them away, however, in terms of performance. In fact, mSQL does not compare with MySQL on any level. InstantDB compares reasonably on a feature level, but MySQL is still much faster. PostgresSQL has some cool SQL3 features, but it carries the bloat of the commercial database engines. If you are looking at the low-cost database engines, then you probably want PostgresSQL if you are using advanced SQL3 features and MySQL if you are doing anything else.

Oddly enough, comparing MySQL with Oracle or some other commercial database is a lot like comparing MySQL with PostgresSQL. The commercial database engines support just about every feature you can think of, but all those features come at a performance cost. None of these database engines can compete with MySQL for read-heavy database applications. They certainly cannot compete on price. They only really compete in terms of SQL3 feature set and commercial support. MySQL AB is working to close the gap on both counts.

Like many applications, MySQL has a test suite that verifies that a newly compiled system does indeed support all of the features it is supposed to support without breaking. The MySQL team calls its test suite "crash-me" because one of its features is to try to crash MySQL.

Somewhere along the way, someone noticed that "crash-me" was a portable program. Not only could it work on different operating systems, but you could use it to test different database engines. Since that discovery, "crash-me" has evolved from a simple test suite into a comparison program. The tests encompass all of standard SQL as well as extensions offered by many servers. In addition, the program tests the reliability of the server under stress. A complete test run gives a thorough picture of the capabilities of the database engine being tested.

You can use "crash-me" to compare two or more database engines online. The "crash-me" page is *http://www.mysql.com/information/crash-me.php*.

What You Get

MySQL is a relational database management system. It includes not only a server process to manage databases, it also provides tools for accessing the databases and building applications against those databases. Among these tools are:

mysql
: Executes SQL against MySQL. You can also use it to execute SQL commands stored in a file.

mysqlaccess
: Manages users.

mysqladmin
: Enables you to manage the database server, including the creation and deletion of databases.

mysqld
: The actual MySQL server process.

mysqldump
: Dumps the definition and contents of a MySQL database or table to a file.

mysqlhotcopy
: Performs a hot backup of a MySQL database.

mysqlimport
: Imports data in different file formats into a MySQL table

mysqlshow
: Shows information about the MySQL server and any objects like databases and tables in that server.

safe_mysqld
: Safely starts up the *mysqld* process on a UNIX machine.

Over the course of this book, we will go into the details of each of these tools. How you use these tools and this book will depend on the way you want to use MySQL.

Are you a database administrator (DBA) responsible for the MySQL runtime environment? The chief concerns of a DBA is the installation, maintenance, security, and performance of MySQL. We tackle these issues in Part II.

Are you an application architect responsible for the design of solid database applications? Architects focus on data modeling and database application architecture. We address the impact of MySQL on these issues in the first few chapters of Part III.

Are you a database application developer responsible for building applications that rely on a database? A database application developer needs tools for providing their applications with data from MySQL. Most of Part III covers the various programming APIs that support application interaction with MySQL.

No matter who you are, you need to know the language spoken by MySQL—SQL. Like most database engines, MySQL supports the American National Standards

Institute (ANSI) SQL2 standard with proprietary extensions. Chapter 4 is a comprehensive tutorial on MySQL's dialect of SQL. The details of the language are covered in several chapters in Part IV.