

CHAPTER 1

Overview of MySQL

Data storage and retrieval is a core element of most applications today. In the early days of software development, programmers wrote their own low-level code to accomplish this. However, they quickly realized that in each application they were essentially reinventing the wheel. Through the usual cycle of trial, error, and subsequent refinement a solution was developed: the data storage and retrieval engine was abstracted into a stand-alone database server with the clients connecting to it and sending requests in a custom language called SQL (Structured Query Language).

Today, developers can choose from many data storage and retrieval products that use SQL. These products are usually referred to as SQL database servers, or sometimes relational database management systems (RDBMSs). Strictly speaking, an RDBMS system must comply with a set of formal requirements. It does not necessarily implement the SQL language, and vice versa—an SQL server may comply only partially with a set of formal RDBMS requirements. However, for practical purposes, the terms are frequently used interchangeably; most RDBMS products implement the SQL standard, and an SQL server that complies only partially with the formal requirements of an RDBMS will still be regarded by many IT specialists to be in the RDBMS league.

Products such as Oracle, DB2, Informix, and Microsoft SQL Server implement the SQL standard and are widely used in the industry. Even if you know nothing about SQL and relational databases, you have no doubt heard of these products—they are the well-known giants in the world of SQL servers. This book, however, is not about the giants of the SQL world. It is about MySQL—a feisty,

lightning-fast underdog of Scandinavian origin that has surprised many developers with its capability to outperform many of the giants.

Unlike most database servers, MySQL is an open source product: its source code is freely available for download to anyone. Programmers can modify the source code to tailor MySQL to their needs. One of the values of open source products is that a wide range of professional developers and users contribute their experience to the software, making it better. As a prominent open source project, MySQL has a large community of loyal supporters. MySQL has benefited in many ways from the contributions of the community, making it more than just a piece of software.

Decision makers in the IT industry are sometimes wary of open source products. The most common concern is that open source products do not have a commercial entity behind them that will take responsibility for supporting the software. In this respect, MySQL is different from most open source products. MySQL AB is a full-fledged company that, at the time of this writing, employs some 50 people all over the world who are responsible for development, support, sales, consulting, training, documentation, and other business functions. It is not the purpose of this book to discuss MySQL AB's business model, but if you are interested in knowing how a basically free open source product can be commercially viable, go to www.mysql.com for more information.

How Is MySQL Most Commonly Used in the Enterprise?

MySQL has several million users, among them many corporate users. In this section, I discuss the most common uses of MySQL in a corporate environment. This summary is based on my own observations while handling commercial support requests at MySQL AB.

Database Backend for a Web Site

Many Web sites have to provide dynamic content (e.g., a news site) and/or collect some data from visitors (e.g., an online store). Thus, there arises the need to have some data storage/retrieval functionality in the Web application. As many Web developers have discovered, MySQL is a perfect tool for this kind of job.

Free to obtain, easy to install and configure, and providing excellent performance and stability, MySQL has been a lifesaver for more than one CTO floundering in the perilous waters of the dot-com world. Some often hesitate to bypass a more expensive alternative, somehow thinking that if MySQL is free it

cannot be good. Nevertheless, when they finally make the decision they are often surprised to discover that MySQL is not only able to handle the load, but can often handle a load that none of the database “giants” they’ve tested has been able to.

Usage Logger

Another common problem in the IT industry is logging events of various types for the purpose of subsequent statistical analysis or simply for record retrieval in the future. This could be, for example, a network traffic monitor, an ISP keeping track of dial-up users, a cell phone provider logging calls, or a Web-usage counter.

MySQL’s speed on insert and select queries makes it an attractive choice for this kind of application. And, of course, the other advantages of MySQL mentioned earlier make it only more attractive.

Data Warehousing

Various technologies today enable the accumulation of large collections of data. For example, a business could have a list of purchase records accumulated over the years, or a computer chip manufacturer could have collected a large dataset of test results. It could be very useful for various purposes to drill through the data and produce a number of statistical reports.

MySQL’s speed on select queries makes it an excellent choice for many such problems. In fact, MySQL was originally written for the specific purpose of solving a particular data-warehousing problem more efficiently than what the market could offer at the time.

Integrated Database

More and more often, software vendors are finding it necessary to integrate a database into their commercial products. For example, a desktop phone book application with various search capabilities will be much easier to write if a lightweight SQL server has been integrated into the system.

The main considerations for a database server in this situation are the cost and the resource requirements. MySQL makes the grade in both aspects. Although not free in this case, the license cost per copy could very well be below \$10 if the volume is large enough. And, of course, MySQL is very frugal about the resource utilization, the binary itself being small in size and the server configuration options allowing it to use no more than a few kilobytes of system memory while still maintaining a decent performance.

Embedded Database

Sometimes an application must process large amounts of data. A low-cost, lightweight database server is the ideal solution for an application programmer working under the restrictions of the embedded environment.

In addition to the advantages mentioned in the previous section, all of which apply here, the portability of MySQL makes it an attractive choice. MySQL can already run on a large number of architectures. Even if it has not yet been ported to the target architecture, the high coding standards that diligently address potential portability issues make it very likely that the port could be done with minimal effort.

Strengths and Weakness of MySQL

The primary question I will try to answer for you in this book is: “Will MySQL solve my needs, and if so, how?” The first step in answering this question is to examine MySQL’s strengths and weaknesses; with a more complete understanding of MySQL’s functionality, you will be able to decide whether MySQL is a good solution for your needs.

Strengths:

- Speed
- Reliability
- Low system resource requirements
- Scalability
- Platform diversity
- Support for a large number of host languages
- ODBC support
- Free or low-cost licensing
- Inexpensive commercial support
- Strong user community backing
- Availability of the source code

Weaknesses:

- Lack of certain SQL features
- Lack of thorough testing on certain platforms
- Difficulty of working with the source code

This list is not comprehensive. I have selected the most common factors in the decision making based on my experiences in working with MySQL users. Let's now discuss each of the strengths and weaknesses in more detail to help you understand its implications for your particular problem.

Strengths

Following is a discussion of MySQL's strengths.

Speed

The core MySQL code was written from the ground up with excellent performance as a primary goal. In fact, Monty Widenius, the original creator of MySQL, was frustrated with the relatively slow speed of databases available on the market at the time, so he wrote MySQL. In my role of enterprise support at MySQL AB, I have often had to deal with the concern on behalf of a potential user that MySQL might not be fast enough for what the user needs, or it will not be able to handle the load. The MySQL support team response is always, "Why don't you write a benchmark that will simulate a part of your application and see it for yourself?" The results of these benchmarks often stun customers and make them converts on the spot.

Many production database systems run a load of 1000 to 2000 queries per second on commodity x86 hardware (dual Pentium 800 with 1–2GB of RAM). I have taken MySQL "for a spin" on several occasions and was able to get 13,000 queries per second on a quad Pentium 700 selecting one record on a key from a table with one million records. So the answer to the question "Will MySQL be fast enough for me?" in most cases is "Yes, and probably much faster than you will ever need it to go." Benchmarking MySQL performance is discussed in Chapter 4, where you can learn how to see for yourself what MySQL is capable of.

A word of caution: Like any database product, MySQL can be slow in some cases if you are not careful when writing your queries. You can avoid this problem by understanding how the server works. One of the goals of this book is to give you the information you need to write efficient queries and keep MySQL running at top speed (see Chapters 12 through 15 for tips and techniques).

Reliability

MySQL has earned a reputation for being able to run unattended for days—even months—after initial setup. Here and there, of course, various issues arise and various bugs are discovered, just like in any other database server, but

overall it is very uncommon for MySQL to go down—and when it does, it is usually able to recover gracefully from the crash. This reputation for reliability got MySQL noticed by a number of enterprise users, who decided it was a great product for their needs. The list includes Yahoo! Finance, Cisco, Texas Instruments, the United States Census Bureau, NASA, Novell, Blue World Communications, Motorola, and many others.

The development team members are extremely focused on making MySQL reliable; they are obsessed (at least by industry standards) with ridding betas of bugs. I have seen MySQL releases postponed in numerous instances just because a single and rather insignificant bug had not yet been resolved. The discovery of one serious bug is reason to build a whole new release and issue a public apology.

Low System Resource Requirements

MySQL is able to make the best of the resources you give it. Of course, the greater the resources, the better the performance you can expect, but minimal resources will not put MySQL out of commission as it does some other database servers. I have successfully run MySQL on a 32MB RAM, 166MHz Pentium system that is not fully dedicated to MySQL. There have been reports of running MySQL under even smaller configurations. The footprint of a MySQL process with a “lowfat diet” configuration is 2MB to 3MB, so it is theoretically possible to run MySQL with as little as 4MB of RAM on the system. (For more information on server configuration, see Chapter 14.)

Scalability

Practical experience shows that MySQL scales well on systems with up to four CPUs, and up to 4GB of RAM, fully taking advantage of the system resources. It is known to work well with tables containing several billion records and has been reported to handle up to 1500 concurrent users without a notable performance degradation.

It is very possible that the true limits of MySQL scalability have not yet been fully explored. At the time of this writing, the MySQL development team has not yet been able to find the time or the hardware to create tests that will do so. They do rely to a great extent on user reports to learn how MySQL performs under load.

If your scalability needs ever exceed the capabilities of a single server, you can use the internal replication capabilities of MySQL to create a cluster of systems and distribute the load by directing the writes to the master host and sending the reads to the slaves. (Replication is discussed in Chapter 16.)

Platform Diversity

MySQL runs on a wide variety of architectures and operating systems. Among the most frequently used are Linux, Windows, Solaris, and FreeBSD. MySQL also runs on Irix, HP-UX, AIX, SCO, Tru64, OpenBSD, NetBSD, and Mac OS X.

Support for a Large Number of Host Languages

When you're developing a database application, one of your primary concerns is the ability to interface with the database server using a particular programming language, which is often referred to as the *host language*. This is another area of strength for MySQL; programmers can communicate with MySQL using C/C++, PHP, Perl, Java, Python, TCL, Ruby, and Eiffel.

ODBC Support

In addition to multiple host language support, MySQL includes an ODBC driver. This gives the programmer the ability to write vendor-independent database applications using the Open Database Connectivity (ODBC) standard.

ODBC connectivity support also allows MySQL to be used with a large number of data management ODBC-capable applications, such as Microsoft Access, Microsoft Excel, Crystal Reports, and many others. ODBC support allows MySQL to be used in Visual Basic and Delphi applications; in ASP (Active Server Pages); as well as with ColdFusion, Borland Builder, and many other development tools and environments.

Free or Low-Cost Licensing

MySQL is distributed under the terms of the General Public License (GPL) created by the Free Software Foundation (FSF). This license allows you to use the software free of charge for both commercial and noncommercial purposes on the condition that any derived product must be distributed with its entire source code under the terms of the same license. More information about GPL, including the full text of the license agreement can be found at www.gnu.org/licenses/gpl.html. For MySQL, the terms of the license mean that in most cases—except when MySQL is included as part of a proprietary product that the vendor is distributing to its customers—MySQL can be used free of charge.

In the case when the license is required, or when the organization policies do not permit the use of a GPL-licensed product, a license can be purchased from MySQL AB at reasonably low cost. At the time of this writing, the price for a

single license is \$200, but drops dramatically as the number of licenses increases, all the way to \$20 per copy for 10,000 or more licenses. The license is issued per server, and does not restrict the number of users.

You can find more information on MySQL licensing at www.mysql.com/support/arrangements.html.

Inexpensive Commercial Support

For those planning to run MySQL in a mission-critical environment, the issue of high-quality commercial support is very important. Even if no problems develop, the CTO needs to know that there is a competent source to turn to if any questions or problems arise. Not having this kind of support available could be a serious concern in making the decision to adopt the use of a product.

MySQL AB provides a wide range of commercial support at a reasonable price, including 24x7 telephone support. The prices range from \$1500 per year for entry level to \$48,000 per year for deluxe. The core developers participate in handling support requests—this means you don't have to jump through several hoops before you start talking with the person who wrote the code that has something to do with your issue.

Strong User Community Backing

As mentioned earlier, MySQL is more than just a database. MySQL's founders have always focused on giving the community more than they take from it. The community has responded with loyalty, hard work, and camaraderie.

How does this strong user community affect a potential enterprise user? The most obvious effect is that it is possible to get free support from the community in addition to the support provided by MySQL. If you post to a newsgroup or a mailing list, or get on IRC and ask a question, it is likely that you will get an answer. Unlike with commercial support, the answer is not guaranteed by MySQL AB, of course, but there are many experts on these lists who give excellent advice. Consider this support option as going fishing—if you know how to fish, you can get free fish from a lake. If you do not know, or do not feel like driving out to the lake, you can simply go to the store and buy some.

Another aspect of a strong user community is that it is relatively easy to find a dedicated MySQL expert who will work for you. What does this mean for an enterprise manager? In addition to the natural objective strengths of MySQL, you will get what one might call “the self-fulfilling prophecy effect.” We usually think of self-fulfilling prophecies as something negative, but in this case it is a positive force and is exactly what an enterprise manager would want. The developer who knows the capabilities of MySQL and likes to work with it

predicts that MySQL will do the job and will then go to work, and for the exact same amount of pay, will put forth the kind of effort it takes to make it happen. You rarely find this kind of enthusiasm and commitment to other database products.

One other aspect of community support worth mentioning here is the “community insurance.” A typical concern about the products of small software companies is the future of the product in case the company itself fails. Even if MySQL AB stopped selling support tomorrow and went back to being solely a group of dedicated developers, the community of users, the source code, and the open source process that built MySQL would still be alive and well.

Availability of the Source Code

Access to MySQL source code is an important advantage for businesses that employ experienced C/C++ programmers. It provides an opportunity for various customizations, improvements, extensions, and bug fixes to be done without having to wait for the vendor to do it.

Another advantage of having the source code available is increased peer scrutiny, which tends to lead to higher code quality. The driving factor in this process is the MySQL developer’s sense of professional honor and reputation. When the source is going to be seen only by few coworkers, there is a temptation to start cutting corners—for example, failing to check for rather uncommon error conditions, avoiding security issues, or writing inefficient code in some places, hoping that the customer hardware is fast enough anyway. However, if the source is going to be seen by a large number of competent programmers across the globe, the attitude is totally different.

Weaknesses

Following is a discussion of MySQL’s known weaknesses.

Lack of Certain SQL Features

The most serious weakness of MySQL is that it currently does not support subqueries, views, stored procedures, triggers, and foreign-key enforcement. This presents a number of issues, perhaps the most important of which is porting existing applications to MySQL. If your database application contains any of the features not supported by MySQL, you will need to rewrite those portions before porting it to MySQL. In some cases, this can be a daunting task.

There exists a strong school of thought in the IT world that some of the features not yet supported by MySQL are an absolute must. Many programmers have learned to depend on those features, and it has become an essential part of their

programming repertoire. It is little surprise, therefore, that this group is somewhat wary of using MySQL, and in some cases might argue quite strongly against it. Their reaction is perhaps somewhat similar to that of a person who has used automatic transmission his entire life and who is now asked to learn to drive a stick-shift.

Avid MySQL users, on the other hand, have learned to live without those features and even enjoy the challenge of having to get around them. Pushed by the challenge, they manage to find elegant solutions that they would have otherwise missed. They learn to emphasize MySQL strengths in their code and work around the weakness when necessary, and in their hands MySQL is able to perform just about any job they set out to accomplish. They tend to argue strongly for the use of MySQL. Using the car metaphor again, their skill allows them to shift gears to get the car to top speed, which they could not have done using automatic transmission.

Because of this difference of opinion, you will frequently encounter a division among computer professionals on whether MySQL would be an appropriate choice for a particular application. If you are a decision maker who has to trust the opinion of your experts and they disagree, then you are in a difficult situation. Can MySQL really meet your needs?

The reality is that in most cases (with the exception of porting some existing applications) MySQL will meet your needs for building database applications, given a good combination of skill, creativity, and motivation on the part of your developers. The problem is that if your database programmers are strongly affiliated with the “true RDBMS religion,” they would be reluctant to use MySQL. Because of this reluctance, they will not put forth as much effort in providing a MySQL-based solution as they would have if you chose another database that they find more suitable; as a result, it will cost you more. If this is the case in your organization, forcing people to use MySQL before you get them to like it would not be a good idea from the pure economic standpoint. However, if you have some folks who are excited about working with MySQL, they would quickly find a way to make things work without the “absolutely necessary” features of the enterprise-level RDBMS.

It should also be mentioned that although MySQL lacks the above-mentioned features, they are currently in development. The goal of MySQL AB for the next two years is to implement all of the missing features and become fully SQL-compliant.

Lack of Thorough Testing on Some Platforms

To understand this weakness, you need to first understand a quality assurance phenomenon that is probably unique to MySQL. MySQL AB has strict coding

standards and a set of thorough testing procedures, but this level of quality assurance can go only so far. The next stage of testing happens when a new version is downloaded from the Internet at the rate of about 20,000 a day, gets installed on a very large number of systems, and is exposed to various combinations and sequences of queries on a rich variety of systems. This process will expose various bugs and system issues that could not have possibly been discovered even in the most rigorous in-house testing. Bugs are reported, and usually quickly fixed. MySQL AB depends on the field testing done by the users for quality assurance.

As a result of this process, it is apparent that the stability and performance of MySQL on a particular platform will be greatly influenced by the size of the install base. The larger the install base, the less of a chance that a critical bug could hide for long periods of time.

Although MySQL AB puts forth a valiant effort to be as cross-platform as possible and not favor one platform above another, the skew in the install base distribution causes some platforms to be much better tested than others. Although it is difficult to know the precise MySQL usage patterns, based on the download statistics, survey results, and a feel of commercial support and public mailing list traffic, I estimate that about 40 percent of installations are on x86 Linux, 25 percent on Microsoft Windows, 15 percent on FreeBSD, 15 percent on SPARC Solaris, and 5 percent on all other platforms.

Due to the quality assurance phenomenon, the remaining 5 percent will always be in the role of poor cousin compared to the dominant platforms, unless they manage to get a larger share of the market in terms of the number of units sold. This is not to say that MySQL is somehow broken on those platforms. It runs very well, and quite a number of people report success in using them, which is due in a great measure to the high focus on portability issues in the code: well-written code has a higher chance of running problem-free on a not-so-well-tested platform. However, a smaller install base will translate into a higher chance of running into a surprise, and you must be aware of that when choosing the platform to run MySQL on.

This situation will change in the future as MySQL AB continues to grow and is able to obtain resources to perform more in-depth internal testing of the platforms that are not extremely popular in the community, such as AIX and HP-UX. This disadvantage may very well be gone in a year or so from the time of this writing.

Difficulty of Working with Server Source Code

We have already discussed the value of having MySQL source code available. Having the source code in your hands definitely gives you a lot of flexibility. The

caveat of working with somebody else's source is that you have to understand it in order for it to be of any practical value. The MySQL server (not client) source is fairly difficult to get into, even for a skilled and experienced C/C++ programmer. Relatively few people dare.

I think there are two major reasons for this difficulty. First, it is a database server, which means it has to have code to efficiently organize data on the disk, cache it in memory to minimize disk access, parse queries, and select a strategy for resolving a particular query. This kind of code by its very nature would be complex, regardless of how clearly it is written and how well it is documented.

The second reason is what one could describe as the “genius code effect.” The core code has been written by Monty Widenius, who in my opinion deserves to be called “the Mozart of computer programming.” A programmer with a sense of taste will have an experience similar to listening to a beautiful piece of music while studying Monty's code. While this beauty is nice and wonderful, the challenge is that even when that beauty is documented, it requires the kind of inspiration that Monty had when he wrote it to understand it and to be able to add on to it without breaking the server. Experience shows that even the best programmers will find this challenging.

The challenge, though, is manageable. MySQL development team members somehow managed to master it, as well as several users who have contributed various patches to the code. Chapter 18 discusses server internals, partially in an attempt to encourage brave souls to learn MySQL server code and contribute to the code base.

MySQL from the Application Developer's Perspective

What options are available to a developer who wants to write a client application interfacing with a MySQL server? Client API libraries exist for C/C++, PHP, Perl, Python, TCL, Ruby, and Eiffel. Java is supported through a JDBC driver. You can also connect to MySQL using the ODBC standard, which opens up MySQL for use with Visual Basic, ADO, ASP, Delphi, ColdFusion, and any other ODBC-capable language, protocol, or development environment. If you are wondering about the use of a particular tool with MySQL and there is no explicit mention of MySQL support anywhere in this book, in the MySQL online documentation, or in the documentation of the tool itself, check to see whether the tool is ODBC-capable. If it is, it can be used with MySQL.

If a certain language or tool currently does not have native or ODBC support to connect to MySQL, there is still hope. If the language or tool is capable of accessing routines in an external library, you can invoke the code from the

MySQL C API library to get the job done. Even in the case of not being able to interface with external libraries at all, you can implement the MySQL client-server protocol on the application level, thus creating a low-level MySQL driver for that language or tool. In practice, the latter has happened only once in the history of MySQL, when Java support was created. APIs for other languages were implemented by simply calling routines from the MySQL C API.

What about the option of extending the server? Unlike many other database servers, MySQL has its source code available, so this is actually an option. As we mentioned earlier, work on MySQL server source can be a challenge. Luckily, the flexibility of MySQL allows you to solve most problems without ever having to consider extending the server. However, in some cases adding code to the server could be beneficial. We discuss this option in more detail in Chapter 18.

Overview of MySQL Integration with Other Industry-Standard Software

How well does MySQL coexist with other industry-standard software? Instead of merely rattling off a long list of products it can work well with, let's examine the principles of integration that determine how well application *X* will integrate with MySQL.

The key word to remember and focus on is *standards*. What standards does the application support? Does it support ODBC? If so, MySQL will integrate with it rather smoothly. This means MySQL will work with Access, Excel, Visual Basic, ASP, Crystal Reports, Delphi, ColdFusion, and many other applications, languages, and development tools.

Although ODBC is a great portability standard, the price to pay for portability is performance. For a desktop application connecting to a remote MySQL server, the performance reduction can be neglected for all practical purposes. However, if ODBC connections are established from a busy application server—in which case performance might be more important than portability—it would be advisable to consider an alternative that would use the native MySQL API.

How well will Web server *X* integrate with MySQL? Again, we go back to the concept of a standard. Most Web servers support the CGI standard, which permits an application executable residing on the Web server to be invoked through a Web request. Thus, the problem is reduced to being able to create an executable on the Web server that can talk to MySQL, which can be done using your language of choice. MySQL, therefore, can be integrated with any Web server that supports CGI, which includes Apache, Roxen, IIS, iPlanet, WebSphere, and a multitude of others.

Although CGI will do the job of integration for most Web servers—thus making it easy to move from one server to another—as you would expect there is a price to pay, just as in the case of ODBC: reduced performance. If the Web server is a busy one, this can be a serious concern, to the point of outweighing the issues of portability. An alternative solution using MySQL's internal server API (such as PHP for Apache and ASP for IIS) might be advisable.

Web applications making the use of Java (JSP, servlets, or applets) can use MySQL using the MySQL Connector/J JDBC driver. The portability of JDBC facilitates porting to MySQL from other databases, and allows you to write code supporting multiple databases.

When making a decision between portability and performance, first perform a set of benchmarks simulating the application to get an idea of what kind of performance difference solution paths can provide and compare it with the performance requirements of the application. Realistic analysis of capabilities versus requirements will help save a lot of development time and other resources.

Getting Help with MySQL

One of the stated goals of MySQL AB is to make MySQL easy to use. To a great extent, it is reaching this goal. Many questions can be resolved easily by a quick look at the documentation. Regardless of how intuitive and well documented a product might be, some users will run into issues that they need extra help to resolve.

So what should you do when you're stuck? One solution (which I mention with a grin on my face) is to read this book. Many other resources are available—some of the most useful ones are listed as follows.

Online Documentation

A searchable online manual is available at www.mysql.com/doc/. This resource is ideal if you have the intention of browsing through the manual section hierarchy. If you would rather search the documentation, a quicker way is to go to www.mysql.com/doc/home.html.

I recommend that you first spend some time looking for the answer in the manual before turning to other sources for help. Even if you are not able to find the solution itself, reading the manual will help you become more familiar with the issues associated with your problem. This increased level of understanding will help you get a better idea of what questions you need to answer to solve the problem, and thus make your use of other resources more efficient.

Mailing List

MySQL AB maintains a public mailing list dedicated to the discussion of MySQL. The posts are open to anyone. Unlike a number of other mailing lists, it is not necessary to subscribe in order to post. To post to the list, simply send a mail to mysql@lists.mysql.com.

The posts are read by about 4000 subscribers, and some of them are very knowledgeable MySQL users. Most questions are answered, although there is no guarantee. The questions that are not answered typically fall into three categories: very frequently asked questions that are answered already in the manual, vague questions, and very difficult questions that need a MySQL developer to answer.

The following rules of etiquette will help you maximize your chances of getting an answer and enhance the quality of your experience with the MySQL community:

- Be sure to spend some time reading the manual so you are at least somewhat familiar with the issue before you ask. In a couple of sentences in your post, state that you have done so and demonstrate that you have some knowledge of the issue.
- Search through list archives first to see if your question has already been answered. You can find a list of searchable archives at www.mysql.com/documentation/searchlists.html.
- On average, you can expect an answer within 24 hours. However, you must remember that you are, figuratively speaking, fishing in a lake. So if the fish are not hungry or simply do not like your bait, you may not catch any. If that happens, try a different bait; in other words, post again by restating your question in a different way so that somebody knowledgeable might find it appealing enough to answer.
- Be as specific as possible, stating all the relevant information, such as the MySQL version you are using, your operating system, the commands you have typed, and the output produced exactly as it appears on the screen. At the same time, try not to overwhelm the potential helper with unnecessary details. Attempt to find the fine balance between too little and too much detail.
- Be polite. Even if you do have an urgent problem, avoid talking about its urgency in the message. Concentrate on asking the question nicely—do not demand an answer. Avoid angry tones. Remember that you are asking for help from the people who are not in any way obligated to give it to you and will be doing you a favor.

- Study the posts you find in the mail archives that have received a good answer in the past and learn from them.
- Never send a personal e-mail to somebody active on the mailing list asking for free assistance with your problem. Just imagine what would happen to that person's mailbox if everyone did that.
- Be creative in your post. Proper use of humor; a short, interesting story about yourself or your company; a brief description of what you are doing with MySQL; or something else that will make your post stand out could make a difference between getting an answer and not getting one, or between having some MySQL guru spend only three minutes thinking about your problem and then giving you his first semi-educated guess and having him work on your problem for as long as it takes to solve it.

In addition to the general list, MySQL AB maintains similar lists for more specialized discussions. Those lists do not have as many subscribers, and the traffic on them is not so heavy. Similar rules of etiquette apply for the posts. In addition, the posts have to stay within the topic of the list. For example, you should not ask for help with installation on the MySQL Internals list, which is dedicated to the discussion of the internals of the MySQL server. More information about those lists can be obtained from www.mysql.com/documentation/lists.html.

Local Linux User Groups

In addition to the general mailing list, it is possible to obtain help with MySQL from a local Linux user group. Many Linux professionals have a good understanding of MySQL. The two skills tend to go together because MySQL is a popular database for Linux.

Such users groups usually have a mailing list of their own. Many require that you subscribe before you can post, which means that you will receive all the posts to that list for the duration of your subscription. It is, of course, possible to unsubscribe at any time.

To find your local Linux user group, visit www.linux.org/groups/.

Commercial Support from MySQL AB

While many issues can be successfully resolved with the help of the user community, there are times when you may need to talk to a developer at MySQL AB. Some of the advantages of this are as follows:

- If you have purchased support, response time is guaranteed.

- The quality standard of the response is high because it comes from somebody working closely with the MySQL development team. If you need to, you will be able to communicate with the developer who wrote the code that you are having issues with or questions about.
- MySQL AB is committed to working with you until the problem is solved.

MySQL AB provides commercial support. Various support options are available, from the entry level to the 24x7 support, at reasonable prices. More information about commercial support contracts is available at www.mysql.com/support/.

