MySQL Overview and New Performance-Related Features

With an ever-growing market share and a continual stream of new software and functionality, MySQL AB's products have made a dramatic impact on the technology industry. New versions, features, and functions are arriving at an increasing pace, but database designers, developers, and administrators might feel overwhelmed by the sheer number of products, along with how frequently they’re updated.

Although the primary purpose of this book is to help you coax the most performance out of your MySQL installation, you should first get the lay of the land of MySQL's entire product suite. To help make things clear, each of the major products can be classified into a small group of categories. In addition, this chapter describes how they are covered within this book, and also mentions some helpful performance-related tools that now ship with the products.

After reviewing the product line, this chapter briefly calls out the versions that are covered in this book, along with some of the platforms that we tested when making our recommendations.

Finally, this chapter lists all of MySQL's major performance enhancements beginning with version 4.0. This list might help you determine when it's time to upgrade (if you're an existing user) or which version to choose (if you’re new to MySQL).

MySQL Products

To help make the most sense of MySQL AB's broad, rapidly growing product suite, these products are classified into the following categories:

- MySQL Core Technologies
- Database Storage Engines and Table Types
- Distributed Computing Technologies
CHAPTER 3  MySQL Overview and New Performance-Related Features

- Graphical Tools and Assistants
- Connectors
- APIs
- Utilities

Each of these categories are explained in the following sections. In addition to these products, MySQL's website features hundreds of partner solutions (commercial, shareware, and freeware) that add value throughout the database design, development, deployment, and management cycles; there are also many applications built using MySQL technology.

MySQL Core Technologies
As the foundation of the entire product line, these technologies span a wide range of functionality, from MySQL's implementation of SQL to its query optimizer to memory management and communication. This book continually points out ways to improve these components' performance. Specifically, chapters are dedicated to making the most of your SQL statements, MySQL's query optimizer, general database server engine settings, and other core technology-related features.

Database Storage Engines and Table Types
Responsible for accumulating and retrieving information, the database storage engine lies at the heart of your MySQL installation. When it comes to picking a specialized storage engine or table type, MySQL offers database designers and administrators a surfeit of choices. This book spends considerable time discussing the following:

- **MyISAM**—Fast, compressible, and FULLTEXT-searchable, this is the default MySQL engine.
- **InnoDB**—Robust, transaction-ready, with strong referential integrity, this storage engine is often used to support complex, high-volume applications, in which transactional guarantees are essential.
- **MERGE**—By creating a single view of multiple identical MyISAM tables, this storage engine is essential to feed reporting or Decision Support System (DSS)/Online Analytical Processing (OLAP) tools.
- **MEMORY**—Previously known as HEAP, its tables are memory-based, extremely fast and easy to configure, letting developers leverage the benefits of in-memory processing via a standard SQL interface.
- **ARCHIVE**—As its name indicates, this storage engine is aimed at applications with very large volumes of infrequently-or-never updated information. Its tables are parsimonious in their consumption of disk resources.
- **CSV**—By creating comma-separated files (.csv), this storage engine makes it very easy for developers to feed other applications that consume these kinds of files with MySQL-based data.
- **FEDERATED**—Define and access remote tables as if they were hosted locally.

- **NDB Cluster**—As the underlying storage engine technology of MySQL Cluster, NDB Cluster makes it possible for multiple computers to keep their in-memory data in sync, leading to dramatic scalability and performance improvements.

Of the preceding list, the MyISAM and InnoDB storage engines see the most usage, which is one reason why this book has chapters dedicated to each of them, along with a chapter exploring MySQL Cluster (Chapter 17, “Clustering and Performance”).

MySQL offers several additional storage engines that are not covered in this book. These include the following:

- **ISAM**—Although this is the original MySQL storage engine, the MyISAM engine has superseded this product; in fact, it will no longer be distributed from version 5.0. Nevertheless, many of the suggestions for improving MyISAM response might also apply for legacy ISAM tables.

- **Berkeley Database (BDB)**—This was the first MySQL storage engine to offer transactional support, among many other advanced features. However, the InnoDB storage engine has garnered, by far, the higher market share for this kind of storage engine, so this book primarily focuses on InnoDB.

- **EXAMPLE**—This is not a storage engine per se; instead, it can best be thought of as a template that shows MySQL’s worldwide development community how to write a storage engine.

- **MaxDB**—This is not a storage engine, but a separate product, originally developed by Adabas, and then overseen by SAP. It’s used by thousands of SAP customers today. Given the different lineages of the main MySQL product line and MaxDB, it is not covered in this book. However, many of the general-purpose recommendations (for example, designing for speed, indexing, and overhead reduction) made in this book are also applicable to MaxDB.

### Distributed Computing Technologies

Replication and MySQL Cluster are the two foremost MySQL distributed computing technologies. Replication refers to the act of keeping multiple “slave” computers in sync with a “master” server. Because this is such a simple yet powerful way to increase throughput, Chapter 16, “Optimal Replication,” is dedicated to replication best practices.

MySQL Cluster leverages multiple computers into a single team; this yields impressive performance and reliability gains, and is only limited by the amount of hardware you have at your disposal. This topic also merits its own chapter. Chapter 17 explores scenarios in which clustering makes good performance sense.
CHAPTER 3 MySQL Overview and New Performance-Related Features

Graphical Tools and Assistants
From the beginning, MySQL products have typically been configured, monitored, and managed from the command line. However, several MySQL offerings now provide an easy-to-use, graphical interface:

- **MySQL Administrator**—Makes it possible for administrators to set up, evaluate, and tune their MySQL database server. This is intended as a replacement for `mysqladmin`.
- **MySQL Query Browser**—Provides database developers and others with a graphical database operation interface. It is especially useful for seeing multiple query plans and result sets in a single user interface.
- **Configuration Wizard**—Makes it easy for administrators to pick and choose from a predefined list of optimal settings, or create their own.
- **MySQL System Tray**—Provides Windows-based administrators a single view of their MySQL instance, including the ability to start and stop their database servers. It is similar to tools offered by other database vendors.

These important capabilities are referred to throughout the book. The Configuration Wizard is examined later in this chapter.

Connectors
Connectors provide database application developers and third-party tools with packaged libraries of standards-based functions to access MySQL. These libraries range from Open Database Connectivity (ODBC) technology through Java and .NET-aware components.

By using the ODBC connector to MySQL, any ODBC-aware client application (for example, Microsoft Office, report writers, Visual Basic) can connect to MySQL without knowing the vagaries of any MySQL-specific keyword restrictions, access syntax, and so on; it's the connector's job to abstract this complexity into an easily used, standardized interface.

Chapter 9, “Developing High Speed Applications,” coverage of optimizing application logic discusses how to streamline ODBC access to MySQL.

APIs
MySQL AB and several third parties provide application programming interface (API) libraries to let developers write client applications in a wide variety of programming languages, including the following:

- C (provided automatically with MySQL)
- C++
- Eiffel
- .NET
MySQL Products

Currently, C, PHP, and Perl represent the most widely used APIs from the preceding list, with ODBC connector-using client application development tools also seeing extensive usage. Although this book is not meant to be a detailed programming guide for any particular language, it does discuss the interplay between your chosen API and MySQL performance in Chapter 9.

Utilities
MySQL's primarily character-based utilities cover a broad range of database management tasks, including the following:

- Exporting information (`mysqldump`)
- Importing information (`mysqlimport`)
- Entering SQL statements, either interactively or via script (`mysql`)
- Checking MyISAM table integrity (`myisamchk`)
- Working with the binary log (`mysqlbinlog`)
- Compressing MyISAM tables (`myisampack`)

Where applicable, this book points out how to use these tools to boost performance. For example, the `mysqldump` utility is covered in great detail in Chapter 15, “Improving Import and Export Operations.”

Performance-Related Tools
MySQL ships a number of tools that can help database administrators configure, test, and tune their MySQL installations. Some of these tools are aimed at people interested in source code, whereas others are aimed at a broader audience. Each of these tools are briefly examined in the following sections.

Benchmark Suite
MySQL's benchmark suite, available for download from their website, is a useful set of automated tests to help determine overall system performance for a broad collection of common database-oriented tasks. For example, the following is a snippet of Perl code that tests inserting new rows into a table:
for ($i=0 ; $i < $opt_row_count ; $i++)
{
    $query="insert into bench values ( " . ("$i," x ($opt_start_field_count-1)) . "$i)";
    $dbh->do($query) or die $DBI::errstr;
}

if ($opt_fast && $server->{transactions})
{
    $dbh->commit;
    $dbh->{AutoCommit} = 1;
}

$end_time=new Benchmark;

print "Time for insert ($opt_row_count)",
    timestr(timediff($end_time, $loop_time),"all") . "\n\n";

Although these tests don't help you determine the optimal database schema design, query
construction, or application logic practices, they are useful for testing the before-and-after
impact of changes to your MySQL server configuration settings. Just be certain that you
take overall system load into consideration when evaluating the results.

**BENCHMARK() Function**

The built-in BENCHMARK() function is useful for running raw timing tests on various computa-
tional functions within MySQL. The results of these tests can help you:

- Compare MySQL's processing capabilities for disparate operations.
- Compare the same operations on different hardware/OS platforms.

For example, you can compare how long it takes MySQL to calculate the MD5 128 bit
checksum for a randomly generated number on a modern, multiprocessor Linux machine
versus a five-year-old, single-CPU desktop computer. This actually tests two MySQL func-
tions: MD5() and RAND().

You could perform this test by hand, time the results, and write them down on paper:

```
mysql> SELECT MD5(RAND());
+----------------------------------+
| MD5(RAND())                      |
+----------------------------------+
```

...
This might get a little tedious after a while, so it's best to use the `BENCHMARK()` function. To make the numbers significant, you can have MySQL perform the operation 500,000 times:

New, expensive Linux server:

```
mysql> SELECT BENCHMARK(500000, MD5(rand()));
```

```
+-------------------------------+
| BENCHMARK(500000, MD5(rand())) |
| +-------------------------------+ |
| 0                             | |
+-------------------------------+
```

```
1 row in set (2.18 sec)
```

History museum-ready desktop:

```
mysql> SELECT BENCHMARK(500000, MD5(rand()));
```

```
+-------------------------------+
| BENCHMARK(500000, MD5(rand())) |
| +-------------------------------+ |
| 0                             | |
+-------------------------------+
```

```
1 row in set (33.27 sec)
```

Notice the difference in how long it took to return the results: This is the number you should watch.

You can use this function to test the amount of time necessary to complete any expression. Note that `BENCHMARK()`, although valuable, does not tell you whether a particular query is efficient. For that kind of task, use the `EXPLAIN` statement, which is reviewed in great detail during Chapter 6, “Understanding the MySQL Optimizer,” study of the MySQL query optimizer.

**Configuration Wizard**

Recent versions of MySQL now offer an optional Configuration Wizard, typically launched upon installation. This section takes a look at the sequence of steps followed by this wizard, along with how these topics are addressed throughout the book.
Note that this wizard is quite dynamic, so your experience might be different from the one presented here (see Figure 3.1).

![MySQL Configuration Wizard](image1)

**FIGURE 3.1** The launch screen for the MySQL Configuration Wizard.

Your first decision is to choose either a boilerplate (“standard”) or customized (“detailed”) installation process. Don’t underestimate the value of the boilerplate configuration; it has been well thought out, and represents a good catch-all setup (see Figure 3.2).

![MySQL Server Instance Configuration Wizard](image2)

**FIGURE 3.2** Choose between a customized or general-purpose configuration.
If you choose the customized path, the first decision you must make is to select the type of database server that you are configuring as shown in Figure 3.3.

**FIGURE 3.3** Choose one of three possible server configurations.

There are marked differences in memory caching and other key server settings depending on the server's role. These distinctions are continually cited throughout the book.

After you’ve chosen a server type, you must then categorize your typical processing profile (see Figure 3.4).

**FIGURE 3.4** Pick the dominant processing profile for this server.
This is an important decision because the workloads experienced by transactional and decision support database servers are quite different, meaning that their respective configurations need to reflect this diversity.

This book keeps this diversity in mind throughout, and makes recommendations accordingly.

The wizard next provides a choice on how to configure the initial InnoDB tablespace (see Figure 3.5).

Enhancing InnoDB performance is explored in Chapter 12, “InnoDB Parameters and Tuning”; disk-specific considerations are covered as part of Chapter 13, “Improving Disk Speed,” general-purpose data storage review.

Configuring the correct number of concurrent sessions, network protocols, and character sets are your next assessments, as shown in Figures 3.6, 3.7, and 3.8.

The impact of connectivity and network settings on performance are examined as part of several chapters, including those on general engine tuning, optimal application development, and network configuration. However, character set issues are not part of the subject matter in this book.

The wizard then gives us a choice on how the database server will be started, as well as security alternatives (see Figures 3.9 and 3.10).
because a Windows server is running for this example, MySQL provides Windows-specific options. The interplay between MySQL and its host operating system is explored in Chapter 14, “Operating System, Web Server and Connectivity Tuning”; aside from the performance degradation inherent in overly complex permission schemes, security is largely a peripheral topic for this book.
After answering the final questions, the wizard automatically generates the configuration file, and starts the server (see Figures 3.11 and 3.12).
FIGURE 3.10  Implementing security preferences.

FIGURE 3.11  Preparing to write the site-specific configuration.
CHAPTER 3  MySQL Overview and New Performance-Related Features

![MySQL Server Instance Configuration Wizard](image)

FIGURE 3.12  Configuration written, MySQL service started.

About the Versions Used for This Book

Open source software has many advantages. One of the most compelling benefits is the speed at which new features and bug corrections arrive. Of course, this makes writing a book about an open source product line like MySQL a bit of a challenge: Things change so rapidly that what is true and cutting-edge today might be the software equivalent of a leisure suit or pet rock tomorrow. With that said, we’ve tried to use the most recent MySQL versions on a variety of platforms, as described in the following sections.

MySQL Versions

In the 4.1 series, we’ve tested our recommendations with versions ranging from 4.1.6 through 4.1.11. For the upcoming 5.0 series, we’ve used 5.0.0 through 5.0.4.

Operating Systems

Unlike MySQL products, operating systems move at a slower release pace. For this book, we’ve installed MySQL products on Windows XP, Windows Server 2003, Red Hat Fedora Linux, and Debian Linux.

Performance-Boosting Features from Version 4.0 Onward

MySQL AB excels at rapidly fixing issues and introducing new features to the entire product line. Many of these new features have a significant impact on performance. Unfortunately,
for the average overworked, underpaid database developer or administrator, it can be
difficult to keep up with all of these new capabilities. In fact, in some cases it's likely that
beneficial upgrades are put off because the administrator is unaware of the advantages of
upgrading.

Because this book focuses on advancing MySQL performance, some of the major database
speed augmentations provided in MySQL versions beginning with 4.0 are listed. These
product enrichments show a clear pattern of continual performance-related improvements
over time.

For brevity's sake, other enhancements that don't really impact system response are omitted.
Internal engine improvements and bug fixes are also skipped, unless they provide direct,
controllable access to developers. Finally, note that each of the topics listed in Table 3.1 are
covered in the appropriate chapter.

<table>
<thead>
<tr>
<th>Version</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>HANDLER interface</td>
<td>This feature provides a fast interface to MyISAM tables, letting developers directly position and move anywhere within the table, and then operate on rows accordingly.</td>
</tr>
<tr>
<td>4.0</td>
<td>FULLTEXT variables</td>
<td>Several server variables give administrators more control over how FULLTEXT indexes are built and managed.</td>
</tr>
<tr>
<td>4.0</td>
<td>UNION support</td>
<td>Improved SQL capability also yields potentially better index utilization.</td>
</tr>
<tr>
<td>4.0</td>
<td>Row estimation</td>
<td>Two new features (SQL_CALC_FOUND_ROWS, FOUND_ROWS()) give developers better visibility into costs and expected results from a query.</td>
</tr>
<tr>
<td>4.0.1</td>
<td>Query cache</td>
<td>You can cache queries and their results, which adds value to your applications regardless of your chosen storage engine.</td>
</tr>
<tr>
<td>4.0.1</td>
<td>Thread control</td>
<td>This feature introduces the innodb_thread_concurrency server setting, improving the speed of parallel operations for the InnoDB storage engine.</td>
</tr>
<tr>
<td>4.0.2</td>
<td>Track long queries</td>
<td>This feature adds the long_query_time configuration setting to provide better log tracking of problem queries. Queries that exceed this threshold are logged.</td>
</tr>
<tr>
<td>4.0.2</td>
<td>Improved ORDER BY</td>
<td>Indexes are now more efficiently used for additional sorting scenarios.</td>
</tr>
<tr>
<td>4.0.3</td>
<td>Faster reads</td>
<td>The added read_buffer_size setting gives administrators more control over sequential read performance.</td>
</tr>
<tr>
<td>4.0.4</td>
<td>Smarter loading</td>
<td>This feature reduces the amount of index creation work performed by LOAD DATA INFILE if data is already present in table.</td>
</tr>
<tr>
<td>4.0.5</td>
<td>Better concurrency</td>
<td>Four transaction isolation levels are now available for developers to control concurrency.</td>
</tr>
<tr>
<td>Version</td>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.0.5</td>
<td>Deadlock detection</td>
<td>InnoDB now has improved deadlock avoidance and detection algorithms.</td>
</tr>
<tr>
<td>4.0.6</td>
<td>Query cache details</td>
<td>The Qcache_lowmem_prunes indicator tells administrators how often contents of the query cache had to be removed because of insufficient memory.</td>
</tr>
<tr>
<td>4.0.9</td>
<td>Avoid table scans</td>
<td>Developers can now use the FORCE INDEX syntax to override the optimizer's query plan, thus going even further to avoid a costly table scan.</td>
</tr>
<tr>
<td>4.0.10</td>
<td>FULLTEXT control</td>
<td>Administrators can now define their own lists of words to be ignored in FULLTEXT searches via the --ft-stopword-file option.</td>
</tr>
<tr>
<td>4.0.13</td>
<td>MyISAM and threading</td>
<td>New parallel table repair and index creation features (configured via new myisam_repair_threads setting) introduce potential for significant index creation speed improvements.</td>
</tr>
<tr>
<td>4.0.13</td>
<td>InnoDB buffer pool</td>
<td>You can now specify how many pages in the InnoDB buffer pool are allowed to be dirty (that is, have altered data or index information) by setting innodb_max_dirty_pages_pct.</td>
</tr>
<tr>
<td>4.0.13</td>
<td>Limit thread delay</td>
<td>The new max_delayed_threads variable controls how many threads are allowed to queue to perform their inserts.</td>
</tr>
<tr>
<td>4.0.14</td>
<td>Guide optimizer</td>
<td>The max_seeks_for_key setting helps drive the optimizer toward choosing an index-based query plan, even if the index holds very duplicate information.</td>
</tr>
<tr>
<td>4.0.14</td>
<td>Slave control</td>
<td>The --read-only parameter for mysqld prevents inadvertent writes to a slave server.</td>
</tr>
<tr>
<td>4.0.16</td>
<td>Buffer control</td>
<td>Five new server variables let administrators more accurately allocate buffer memory. Variables include transaction_alloc_block_size, transaction_prealloc_size, range_alloc_block_size, query_alloc_block_size, and query_prealloc_size.</td>
</tr>
<tr>
<td>4.0.22</td>
<td>InnoDB deadlocks</td>
<td>Administrators can now tune the new innodb_table_locks session variable to reduce the likelihood of deadlocks.</td>
</tr>
<tr>
<td>4.1</td>
<td>MEMORY B-tree index</td>
<td>Database designers can now elect to use a B-tree index on a MEMORY table, instead of the default hash index.</td>
</tr>
<tr>
<td>4.1</td>
<td>Windows memory</td>
<td>MySQL now supports the extended (up to 64GB) memory capabilities (AWE) on Windows servers.</td>
</tr>
<tr>
<td>4.1</td>
<td>Detailed EXPLAIN</td>
<td>The EXPLAIN query report now provides additional data useful in helping determine if a query is as efficient as possible.</td>
</tr>
</tbody>
</table>
Performance-Boosting Features from Version 4.0 Onward

4.1 Better `mysqldump` Running `mysqldump` now disables foreign key checks automatically when generating a load file, helping to speed the reloading process.

4.1 Improved MyISAM Several new important features for the MyISAM engine are now available. First, administrators can now use symbolic links for MyISAM tables, which lets tables be spread among multiple disk drives if desired. Next, key cache performance has been boosted by allowing for midpoint insertions, as well as permitting multiple threads to simultaneously access the cache.

4.1 Temp round robin Administrators can now configure several directories to serve as temporary storage for MySQL by setting the `tmpdir` parameter. This can help to balance the disk load among multiple drives.

4.1.1 Key caching/indexes You can now create multiple, specialized instances of the MyISAM performance-enhancing key cache. The new `preload_buffer_size` setting lets administrators configure memory when preloading indexes.

4.1.1 Filesort behavior The new `max_length_for_sort_data` setting helps MySQL determine what kind of file sort algorithm to use when processing an `ORDER BY`.

4.1.2 Index enhancements You can now specify up to 1,000 bytes for a MyISAM table's index key; you can create up to 64 indexes per table for InnoDB and MyISAM.

4.1.2 Large table support You can now set MyISAM's row pointer size (`myisam_data_pointer_size`), which lets you address very large tables.

4.1.5 InnoDB expansion The new `innodb_autoextend_increment` setting lets you control much additional disk space InnoDB requests when growing a tablespace.

4.1.6 Purge control The `innodb_max_purge_lag` setting lets you control what happens when there is a significant amount of information to purge from internal InnoDB logs.

4.1.8 Better `mysqldump` New parameters now let you use MySQL to generate a point-in-time InnoDB backup.

5.0.0 Index merge The MySQL optimizer is now able to create query plans that use multiple indexes to satisfy an `OR` clause.

5.0.0 Stored procedures You can now create server-side stored procedures, helping to remove workload from clients as well as centralize software development.

---

**TABLE 3.1 Continued**

<table>
<thead>
<tr>
<th>Version</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Better <code>mysqldump</code></td>
<td>Running <code>mysqldump</code> now disables foreign key checks automatically when generating a load file, helping to speed the reloading process.</td>
</tr>
<tr>
<td>4.1</td>
<td>Improved MyISAM</td>
<td>Several new important features for the MyISAM engine are now available. First, administrators can now use symbolic links for MyISAM tables, which lets tables be spread among multiple disk drives if desired. Next, key cache performance has been boosted by allowing for midpoint insertions, as well as permitting multiple threads to simultaneously access the cache.</td>
</tr>
<tr>
<td>4.1</td>
<td>Temp round robin</td>
<td>Administrators can now configure several directories to serve as temporary storage for MySQL by setting the <code>tmpdir</code> parameter. This can help to balance the disk load among multiple drives.</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Key caching/indexes</td>
<td>You can now create multiple, specialized instances of the MyISAM performance-enhancing key cache. The new <code>preload_buffer_size</code> setting lets administrators configure memory when preloading indexes.</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Filesort behavior</td>
<td>The new <code>max_length_for_sort_data</code> setting helps MySQL determine what kind of file sort algorithm to use when processing an <code>ORDER BY</code>.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Index enhancements</td>
<td>You can now specify up to 1,000 bytes for a MyISAM table's index key; you can create up to 64 indexes per table for InnoDB and MyISAM.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Large table support</td>
<td>You can now set MyISAM's row pointer size (<code>myisam_data_pointer_size</code>), which lets you address very large tables.</td>
</tr>
<tr>
<td>4.1.5</td>
<td>InnoDB expansion</td>
<td>The new <code>innodb_autoextend_increment</code> setting lets you control much additional disk space InnoDB requests when growing a tablespace.</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Purge control</td>
<td>The <code>innodb_max_purge_lag</code> setting lets you control what happens when there is a significant amount of information to purge from internal InnoDB logs.</td>
</tr>
<tr>
<td>4.1.8</td>
<td>Better <code>mysqldump</code></td>
<td>New parameters now let you use MySQL to generate a point-in-time InnoDB backup.</td>
</tr>
<tr>
<td>5.0.0</td>
<td>Index merge</td>
<td>The MySQL optimizer is now able to create query plans that use multiple indexes to satisfy an <code>OR</code> clause.</td>
</tr>
<tr>
<td>5.0.0</td>
<td>Stored procedures</td>
<td>You can now create server-side stored procedures, helping to remove workload from clients as well as centralize software development.</td>
</tr>
</tbody>
</table>
### CHAPTER 3  MySQL Overview and New Performance-Related Features

<table>
<thead>
<tr>
<th>Version</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0.1</td>
<td>Views</td>
<td>Views provide numerous benefits for administrators, including letting them define relationships among multiple tables, specify filter criteria, and present a simpler data interface to developers.</td>
</tr>
<tr>
<td>5.0.1</td>
<td>Optimizer tuning</td>
<td>The <code>optimizer_prune_level</code> and <code>optimizer_search_depth</code> settings let you dictate how you want the MySQL optimizer to examine potential query plans. Also, you can use the new <code>Last_query_cost</code> indicator to get an idea of the price of your most recent query.</td>
</tr>
<tr>
<td>5.0.2</td>
<td>InnoDB indicators</td>
<td>Many new indicators for InnoDB report on its current status and workload.</td>
</tr>
<tr>
<td>5.0.2</td>
<td>View control</td>
<td>The <code>updateable_views_with_limit</code> setting helps prevent a runaway update of a view when <code>LIMIT</code> is specified.</td>
</tr>
<tr>
<td>5.0.2</td>
<td>Server-side cursors</td>
<td>This feature lets you define structures to hold retrieved information. Typically used in conjunction with stored procedures, they add significant processing flexibility to your applications.</td>
</tr>
<tr>
<td>5.0.2</td>
<td>Triggers</td>
<td>Triggers let you define data-driven events that will kick off activities on the server.</td>
</tr>
<tr>
<td>5.0.3</td>
<td>FEDERATED engine</td>
<td>This feature defines table structures and operates upon data in remote locations as if they were local.</td>
</tr>
</tbody>
</table>