



# **MySQL Enterprise Monitor with Query Analyzer**

## ***Monitoring Applications for Problem Queries***

**A MySQL<sup>®</sup> White Paper**

April, 2010



## Table of Contents

<b>Introduction .....</b>	<b>3</b>
<b>Application Strategy – Performance and Scalability!.....</b>	<b>3</b>
<b>Common Challenges to Performance and Scalability.....</b>	<b>4</b>
<b>Common Ways to Monitor and Tune Problem Queries.....</b>	<b>5</b>
<b>Meeting Challenges with MySQL Enterprise Monitor and Query Analyzer .....</b>	<b>7</b>
<b>Monitoring for Performance and Scalability .....</b>	<b>8</b>
<b>Finding and Fixing Problem Queries with the Query Analyzer .....</b>	<b>9</b>
<b>Conclusion .....</b>	<b>11</b>
<b>Learn More.....</b>	<b>12</b>
<b>Appendix A – MySQL Enterprise Monitor and Query Analyzer Deployments .....</b>	<b>13</b>



## Introduction

Most DBAs and Developers understand how poorly tuned SQL code can negatively impact application performance and throughput. Even code that was tuned in development and performed well in system and QA testing may bog down under production loads. Further, despite your best efforts, tuning and optimizing your backend databases will not help fix performance problems caused by poorly tuned code. This paper explores common use cases and challenges faced by those developing and supporting MySQL. This paper also provides a review of MySQL Enterprise, which is comprised of database and monitoring software and production support services that are designed to alleviate these challenges. It also introduces the new Query Analyzer capabilities of the MySQL Enterprise Monitor and describes how the Query Analyzer can be used to tune and optimize SQL code during development and then to monitor applications after they are promoted to production.

## Application Strategy – Performance and Scalability!

How an application is designed to scale is typically a function of how it will ultimately be used. MySQL is flexible in that it supports scalable configurations for medium to high volume OLTP applications and the most demanding, highly trafficked sites on the web. The most basic configuration involves an application that connects directly to a backend MySQL database which services all user level activity and requests. This is commonly configured as shown in Figure 1:

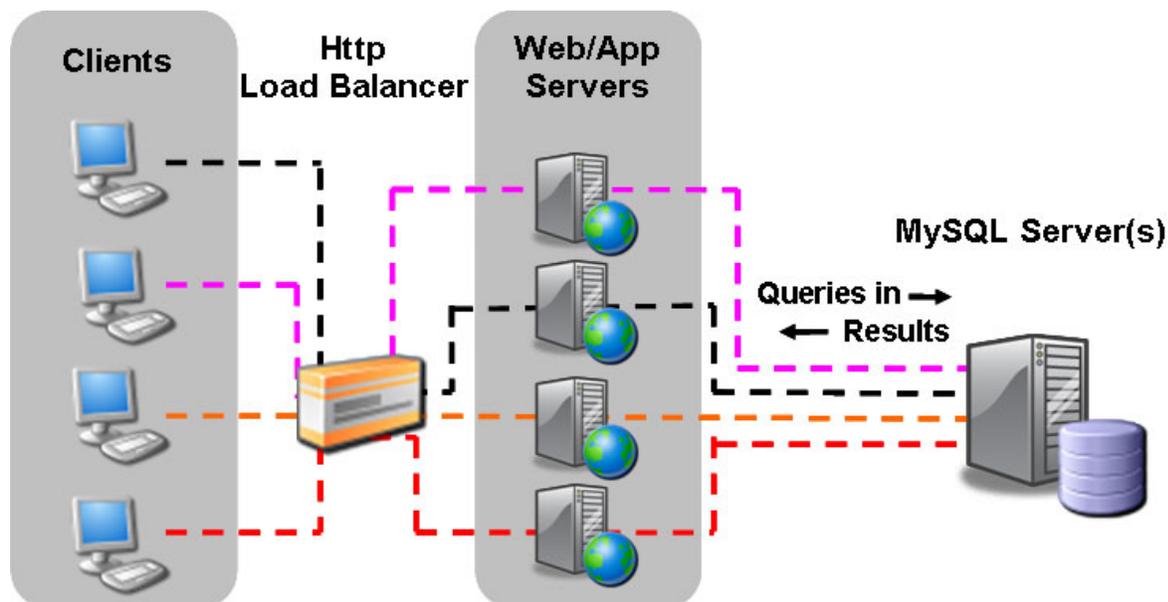


Figure 1: Application configured for direct database connectivity

In this example the application is configured to connect and send all query code directly to a MySQL server, where the request is processed and the results are sent back to the requesting client. In this configuration, scalability is somewhat limited, but can be attained by upgrading the processing power of the server that is running MySQL.

### MySQL Replication and Scale-out

One of the most popular features of MySQL is replication, which ships with every version of the MySQL server. It is characterized by being extremely easy to use and setup, very reliable, and offering high rates of transfer speed. MySQL replication is used by a great number of online and web businesses to “scale-out” an application, by using different physical servers to make up one logical database – the end result being faster performance than putting the entire load of an application on one machine. To this end MySQL replication is typically used to clone some or all parts of any application horizontally across many different servers. While some applications are designed to connect and send all queries to a master, which then propagates writes across the underlying slaves, many are configured to off load read activity to dedicated slaves, freeing up the master to service writes. This is commonly configured as shown in Figure 2:

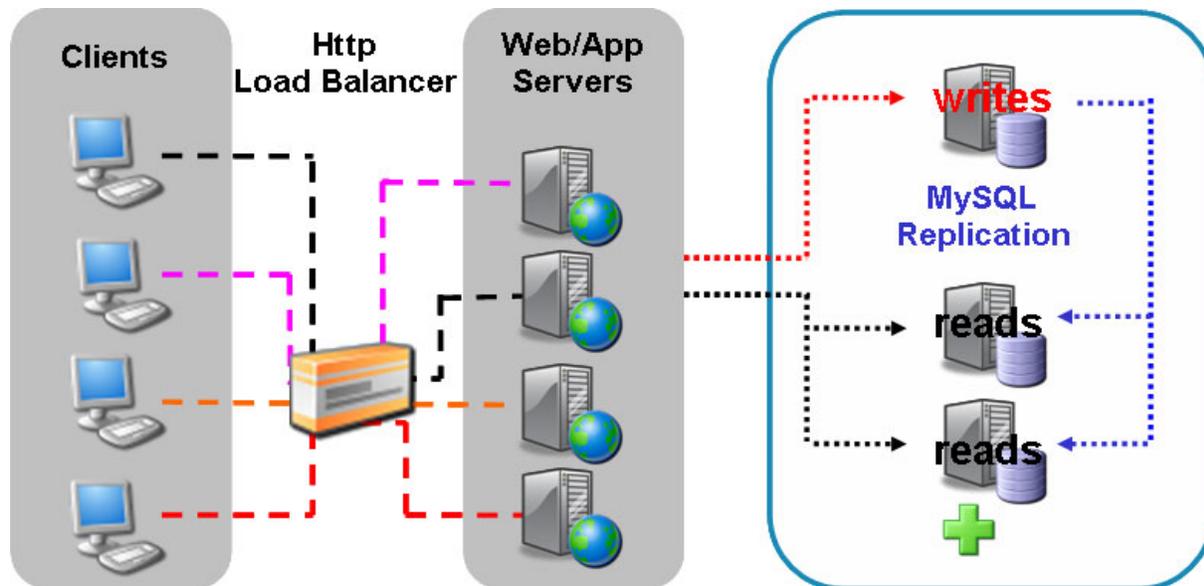


Figure 2: Application configured for application level read/write load balancing

In this configuration the online application is configured to be aware of the masters and slaves that make up a MySQL replication topology. Scalability and performance are achieved by:

- Application level load balancing of writes and reads, to master and slaves, respectively.
- Only writes are sent to a master (reads are off loaded to slaves). Slaves are asynchronously updated.
- Reads are typically serviced in a “round robin” fashion across read slaves unless more advanced load balancing is in place.
- Additional read capacity can be added by adding more read slaves to the topology.

## Common Challenges to Performance and Scalability

The two simple examples above represent foundational configurations commonly used in conjunction with the deployment of scalable MySQL applications. It is important to note that each has the same basic flow:

1. Client requests new page load from application, waits for response.
2. Application forwards request to MySQL waits for response.
3. MySQL returns page data to application.
4. Application loads new page to client.



The performance of an online application is often measured by user response times or more specifically in how fast requests for new page loads are filled. In this context, the performance of a particular web site, even down to the page level, can impact the host company's balance sheet; basically if a company's web site is slow, consumers will tend not to frequent it and consequently will not consume information about the goods and services offered there. Further, even the most productivity enhancing internal application is useless if clients have to wait each time a new page load is requested. This could mean lost productivity or worse fewer customer orders or calls are serviced during peak times. For e-commerce companies this may mean customers will shop, book, socialize, gamble or get their software from a competitor. Both mean lost opportunity and lost customers.

Complicating matters is that many times performance degradations are not telegraphed in advance and often slow response times are less about system loads and more about application level SQL code. While MySQL is known for its performance, reliability and ease of use, it comes with limitations around tracking application and user level activity, specifically around poorly performing queries:

- The MySQL server is not well instrumented for collecting performance related data.
- Query logs help but are low-level and not centralized. Prior to 5.1, the slow query log had to be manually enabled.
- "Evil" queries or "good-queries-gone-bad" are hard to pinpoint for tuning.

Outside of monitoring for basic MySQL and Operating System performance, nothing is more challenging than diagnosing performance issues that are rooted in application level SQL code. This is especially true as it relates to web applications that are constantly being updated or that are heavy users of dynamic SQL. Pinpointing when a slowdown started is easy, finding and fixing the cause, especially if the problem is rooted in code that has been victimized due to an application upgrade, dropped index, or by a combination of user specific variable inputs, is the challenge to be met.

## Common Ways to Monitor and Tune Problem Queries

Even with the limitations noted above, MySQL provides several options to help DBAs and Developers find and fix problem queries. These options include:

### Enabling the Slow Query Log

The Slow Query Log is enabled by starting the monitored MySQL server with the `--log-slow-queries[=file_name]` option. When enabled the Slow Query Log lists queries that took longer than X seconds (user defined value typically between 1 and 10). Beginning in MySQL 5.0 it can be configured to also log any queries that do not use indexes. The entries in the Slow Query Log are formatted to include client IP address, date/time the query started, execution time, rows sent to client, rows read and finally the query content. A typical entry looks like this:

```
# Time: 08073101 16:25:24
# User@Host: root[root] @ localhost [127.0.0.1]
# Query_time: 8 Lock_time: 0 Rows_sent: 20 Rows_examined: 243661
SELECT part_num FROM `inventory`.`parts` WHERE
('vendor' = "foo") ORDER BY `delivery_datetime` DESC LIMIT 100;
```

While the Slow Query Log provides good insight into expensive queries it comes with some known limitations:

- Prior to 5.1, the MySQL server had to be stopped and restarted for it to be enabled.
- When enabled, the log grows very large, very quickly. For this reason, it is only routinely enabled when a slowdown occurs or when managed via scripts using FLUSH LOGS.



- The log entries must be parsed and manually sorted for relevance to determine which queries were running during a reported slowdown in performance. Logged queries can be summarized using the **mysqldumpslow** command but this is still somewhat tedious and time consuming.

### SHOW PROCESSLIST;

The MySQL process list allows you to see what processes are currently running on your MySQL servers. The results of the command are formatted to include the process id, user, originating host, database, command type, status and info (which includes the executing query when applicable). The output looks like this:

```
mysql> SHOW FULL PROCESSLIST\G

***** 1. row *****
Id: 1
User: MyUser
Host: localhost
db: merlin
Command: Query
Time: 1030455
State: Sending Data
Info: SELECT * from advisors;
***** 2. row *****
Id: 2
User: MyUser
Host: localhost
db: inventory
Command: Query
Time: 1004
State: Sending Data
Info: SELECT part_num FROM `inventory`;
2 rows in set (0.00 sec)
```

Each of these options can be automated via custom scripts. In fact, many DBAs use the Slow Query Log in conjunction with SHOW PROCESSLIST to monitor their MySQL servers for poorly performing queries. A typical scenario involves custom scripts that:

- are deployed to each MySQL server
- parse the Slow Query Log for queries that exceed a specific execution time threshold
- execute SHOW PROCESSLIST, capture and aggregate the results for analysis
- notify the DBA staff there is problem

The DBA then manually parses through the aggregated data to determine which MySQL servers have queries that may be causing a bottleneck

### EXPLAIN

After a problem query is identified, the EXPLAIN statement can be used to analyze if indexes are being used effectively, whether a filesort is required, and what tables and columns are being queried. The EXPLAIN statement supports many options, but the typical output looks like this:

```
Query in:
EXPLAIN SELECT part_num FROM `inventory`.`parts` WHERE
('vendor' = "foo") ORDER BY `delivery_datetime` DESC LIMIT 100;
```

EXPLAIN out:

```
ID select_type table type possible_keys key key_len ref rows Extra
```



```
1 SIMPLE      parts ref  vendor, part_num  vendor 3      NULL 3872 Using WHERE
1 row in set (0.00 sec)
```

The EXPLAIN output is really the final stop in fixing bad query code. The DBA can tune the code by adding indexes, altering table columns for length or datatype or changing the query structure itself to improve performance. This is typically done iteratively (tune, EXPLAIN query, tune some more) until the query is tuned or performs within SLA levels.

While these options have served DBAs and Developers well in solving SQL related problems, they are labor intensive and custom scripts or third party solutions are required to integrate the end to end process, which only grows in complexity as the number of MySQL servers to be monitored grows in number.

The remainder of this paper focuses on the MySQL Enterprise Monitor with Query Analyzer and how it can be used to alleviate common pain points around building and scaling applications on MySQL.

## Meeting Challenges with MySQL Enterprise Monitor and Query Analyzer

The MySQL Enterprise Monitor and Query Analyzer are included as part of MySQL Enterprise. MySQL Enterprise is a commercial offering comprised of database and monitoring software plus production support for organizations delivering highly available, business critical applications and services.

### MySQL Enterprise includes three primary components:

**MySQL Enterprise Server** – the most reliable, secure and up-to-date version of MySQL that provides a cost-effective solution for:

- Scalable Web 2.0 and Enterprise 2.0 on-line applications
- Corporate OLTP Applications
- Multi-terabyte Business Intelligence and Data Warehousing deployments
- Complex communications networks

The MySQL Enterprise Server is configured for high performance and supports both “safe” and ACID (atomic, consistent, isolated, durable) compliant transactions with full commit, rollback, crash recovery and roll level locking capabilities.

**MySQL Enterprise Monitor and Advisors with Query Analyzer** – a distributed web application that you deploy within the safety of your corporate firewall. The Monitor continually monitors all of your MySQL servers and applications and proactively alerts you to potential problems and tuning opportunities before they become costly outages. It also provides you with MySQL expert advice on the issues it has found so you know where to spend your time tuning your systems.

**MySQL Production Support Services** – MySQL Enterprise includes 24 X 7 X 365 production support for your MySQL servers and the Enterprise Monitor to help ensure your business critical applications are continuously available and running at their peak. MySQL Production Support Services include:

- Online Self-Help Support – The knowledge base is a self-help tool that provides you with access to 2,000+ technical articles on MySQL specific topics that help quickly answer questions and solve problems.
- Problem Resolution Support – Allows you to work directly with the MySQL Production Support team via phone, email or an online for quick resolution of technical problems.



- Advanced Support for MySQL High Availability and Scalability Solutions – MySQL Enterprise includes full production support for additional advanced MySQL features and third-party solutions to scale the availability and performance of your online applications.

## Monitoring for Performance and Scalability

One of the biggest challenges for those tasked with developing and supporting applications built on MySQL is that over time MySQL servers tend to propagate rapidly, especially when MySQL is deployed as part of a scale-out strategy. A recent survey of MySQL users revealed that the average MySQL DBA is supporting 5-10 physical MySQL servers. The same survey revealed that, on average, those using MySQL Replication to scale high traffic web applications add a new read slave to their topology every 2 months, each with its own administration and monitoring needs. The Enterprise Monitor is designed to help you manage more MySQL servers with less time and effort. Running completely within your datacenter, the Enterprise Monitor is a “Virtual DBA” assistant that helps your DBA staff keep your MySQL servers running at the highest levels of security, performance and availability. The Monitor proactively monitors your MySQL systems and provides expert advice on how even those new to MySQL can tighten security, identify and tune bad SQL code and find and fix problems *before* they can become serious problems or costly outages.

### ***Enterprise Dashboard***

Regardless of the size of your MySQL environment, each server requires specific attention when it comes to basic administration, security, performance monitoring and availability. Dynamic, replicated environments must be documented to reflect current Master/Slave relationships as new capacity is added for scale-out to ensure you can monitor for scalability related latency issues. To give you a proactive advantage in all of these areas, the MySQL Enterprise Monitor provides the Enterprise Dashboard. Using the Enterprise Dashboard, you can monitor MySQL and OS specific metrics for single servers or complete application groupings, and can quickly determine where you have real and potential performance issues. The Enterprise Dashboard is designed so you can easily understand the complete security, availability, and performance picture of all your MySQL servers in one place, all from a thin, browser-based console. The Enterprise Dashboard is shown in Figure 3:



Figure 3: MySQL Enterprise Dashboard

## Finding and Fixing Problem Queries with the Query Analyzer

Integrated into the Enterprise Monitor the Query Analyzer is designed to help Developers and DBAs accurately pinpoint SQL code that is causing a slowdown, quickly diagnose the inefficiencies and get the affected application back to peak performance. To this end, the Query Analyzer provides options to use the MySQL Agent/Proxy technology or the MySQL Connectors to collect application SQL code and performance metrics, and then report the results back to the Service Manager for monitoring and presentation.

See Appendix A for specifics on the Enterprise Monitor components and the available Query Analyzer deployment and enablement options.

### How the Query Analyzer Helps the Developer and DBA

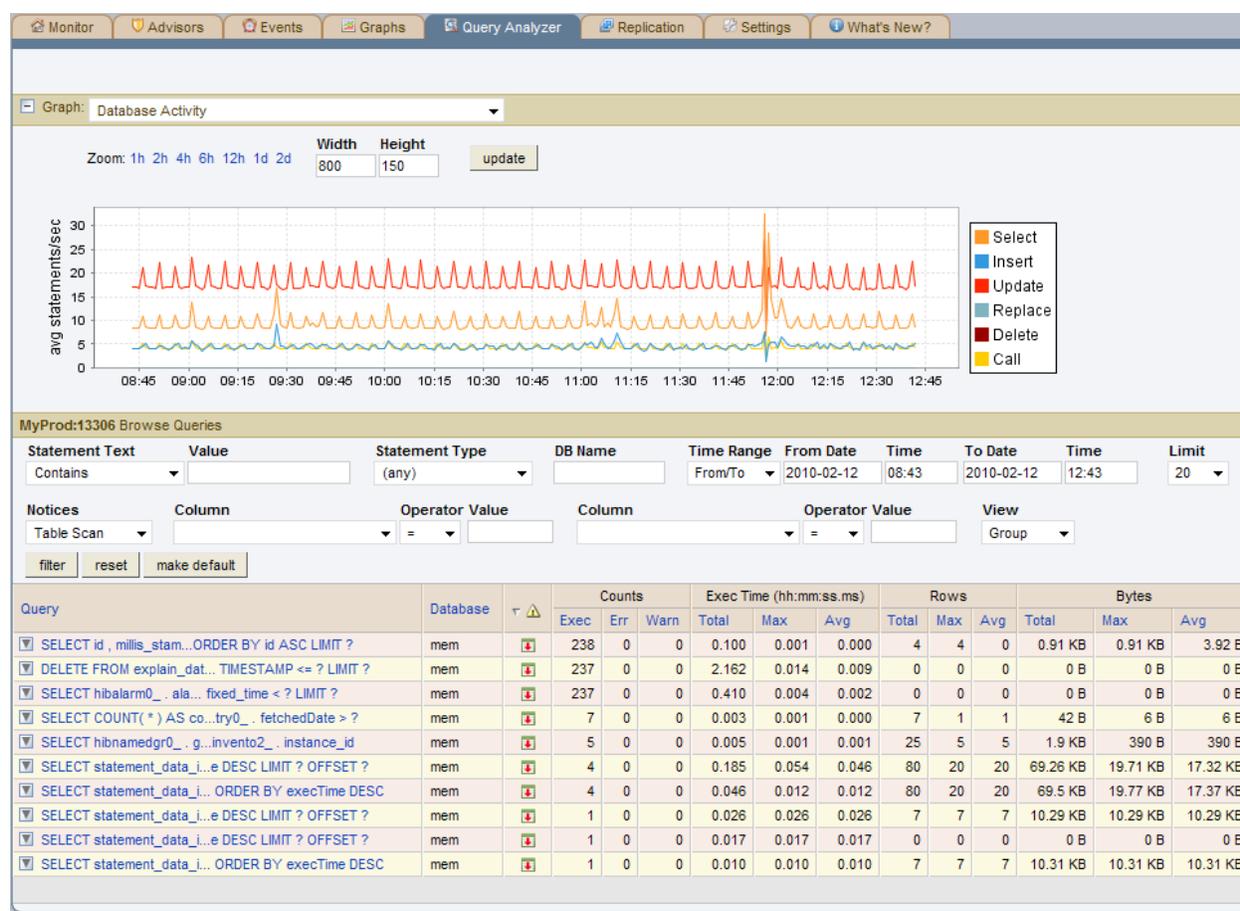
The Query Analyzer saves you time and effort in monitoring your MySQL servers for problem queries by providing:

- An integrated monitoring solution for all supported versions of MySQL (4.1 and higher).
- Aggregated query content and performance stats in real time with no reliance on MySQL logs or SHOW PROCESSLIST.
- Visual correlation of query activity with Monitor graphs.
- A consolidated view into query activity across all MySQL servers, no user parsing required.
- Historical browsing/analysis of queries across all MySQL servers.



- Aggregated, searchable roll ups of all queries in canonical form (no variables) with total number of executions, total execution time, total data size and date/time of when query was “first seen”:
  - **Total Executions** helps you see if you have queries that are running too often or in error. Even properly tuned queries cause performance problems when they run excessively.
  - **SQL Warning and Error counts** help you see queries that did not finish or that returned incorrect result sets. These executions may never be found using other query monitoring options.
  - **Total Execution Time** helps you see your most expensive queries across all of your servers. This value helps you see where your systems are spending the most time and resources and where you should focus your tuning efforts.
  - **Total Data Size** (Rows and Bytes) helps you analyze if you have queries that are returning more data than your application is using. Sorting on this value, examining the underlying queries and comparing the returned rows and columns with your application requirements will help you tune your applications and schema for better performance.
  - **“First Seen”** allows you to easily monitor when queries attributed to new application deployments are affecting the performance of your production systems.
- Drill downs into query details, number of executions, execution stats, visual EXPLAIN plan, and query specific execution graphs.

The Query Analyzer summary of all queries and aggregated execution statistics is shown here:

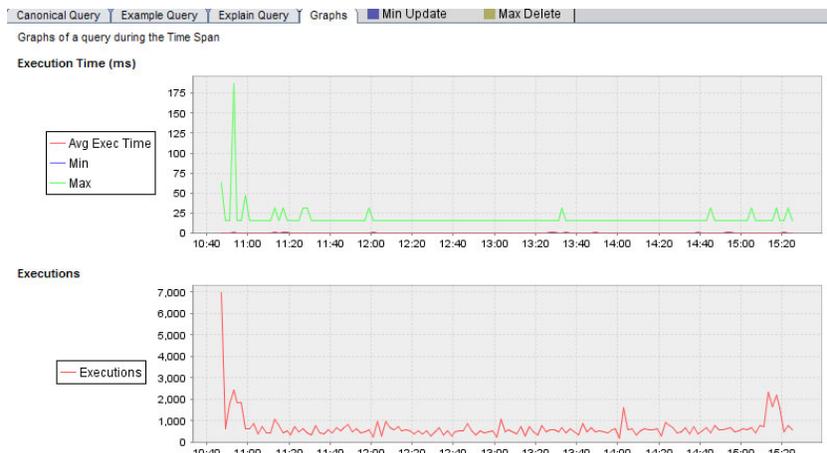


The worst performing queries are presented will full drill down details so you can analyze what variable combinations are leading to performance degradations.



Details include:

- Fully qualified (variable substitution) views of worst performing statements.
- Execution time for specific occurrence
- Date/time of execution
- Originating session/user/thread id
- Source/destination host and MySQL server
- Integrated EXPLAIN of worst performing query for analysis of execution path.



Canonical Query | Example Query | Explain Query

Explain of a query that occurred during the Time Span

**Explain**

id	select_type	table	type	possible_keys	key	key_len	ref	rows	extra
1	PRIMARY	p	ALL	null	null	null	null	16334	Using where
2	DEPENDENT SUBQUERY	payment	ref	idx_fk_customer_id,idx_customer_paydate	ix_customer_paydate	2	sakila.p.customer_id	14	Using index

hide

Using the Enterprise Monitor Query Analyzer you can:

- Tune your SQL code during active development before it is promoted to production.
- Quickly identify queries that are negatively affecting the throughput of your production environments
  - by query type, content, server or application user.
- Once problematic code is surfaced you can:
  - View an EXPLAIN to determine the current access path and index usage and tune the offending code for better performance.

In addition to the code monitoring and diagnostic benefits noted above, the Query Analyzer:

- Shares common, core components with other MySQL Enterprise scalability and performance solutions.
- Adds little pass-thru overhead to your monitored environments (benchmarked at < 400 micro seconds per packet).
- Is fully supported by MySQL production support services.

## Conclusion

MySQL is the most popular open source database in use today for two main reasons: its lower total cost of ownership compared to other proprietary database products and because of its performance, reliability and ease of use. These things allow companies of any size to build low cost, highly available, on-line applications that scale as their business grows. As these applications, and the MySQL servers that power them, grow in number and become more business-critical outages and slow response times become more costly. From regular updates to the Enterprise Server to micro-level SQL analytics, MySQL Enterprise and its suite of database and monitoring software and production support services is designed to help ensure each of your applications and supporting MySQL servers is maintained at the highest levels of security, performance and availability.



## Learn More

The MySQL product portfolio is the most popular open source database software in the world. Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems and packaged software -- including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube and Zappos.com. For more information on how MySQL provides corporate users with commercial subscriptions and services, and actively supports the large MySQL open source developer community visit <http://www.mysql.com/>.

### ***More customer success stories and use cases:***

<http://www.mysql.com/why-mysql/case-studies/>

### ***MySQL Enterprise, the Enterprise Monitor, Query Analyzer, Production Support***

<http://mysql.com/products/enterprise/>

### ***MySQL Replication and Scale-Out***

[http://mysql.com/products/enterprise/high\\_availability.html](http://mysql.com/products/enterprise/high_availability.html)

### ***MySQL High Availability***

[http://www.mysql.com/products/enterprise/high\\_availability.html](http://www.mysql.com/products/enterprise/high_availability.html)

### ***MySQL Professional Services and Consulting***

<http://mysql.com/consulting/>

## Appendix A – MySQL Enterprise Monitor and Query Analyzer Deployments

This section explains the available options for deploying the MySQL Enterprise Monitor and enabling the Query Analyzer. Each example deployment references some combination of the following components that are needed to collect MySQL, OS and Query performance data:

- MySQL Enterprise Service Manager – Central server for MySQL Enterprise Monitor deployments. Comprised of Java services that store, monitor and present MySQL and OS performance metrics on the Enterprise Dashboard web user interface.
- MySQL Agent – Deployed to each monitored MySQL server, the agent collects inventory and performance data about MySQL and the OS and sends the data back to the Service Manager for monitoring and presentation.
- MySQL Proxy – Plugin to Agent that collects query content and performance data that is aggregated, monitored, and presented on the Monitor’s Query Analyzer tab.
- MySQL Connectors – Used by native applications to connect to MySQL databases. Connectors are available for the following:
  - Connector/J – used to connect Java applications to MySQL
  - Connector/NET – used to connect Windows/.NET applications to MySQL
- MySQL Enterprise Plugin for Connectors – Used by Enterprise Monitor to collect query performance data from MySQL Connectors. Plugins are currently available for the following:
  - MySQL Enterprise Plugin for Connector/J (requires Connector/J v5.1 or newer)
  - MySQL Enterprise Plugin for Connector/NET (requires Connector/NET v6.2 or newer)

### Monitoring Queries using the MySQL Proxy Plugin to MySQL Agent

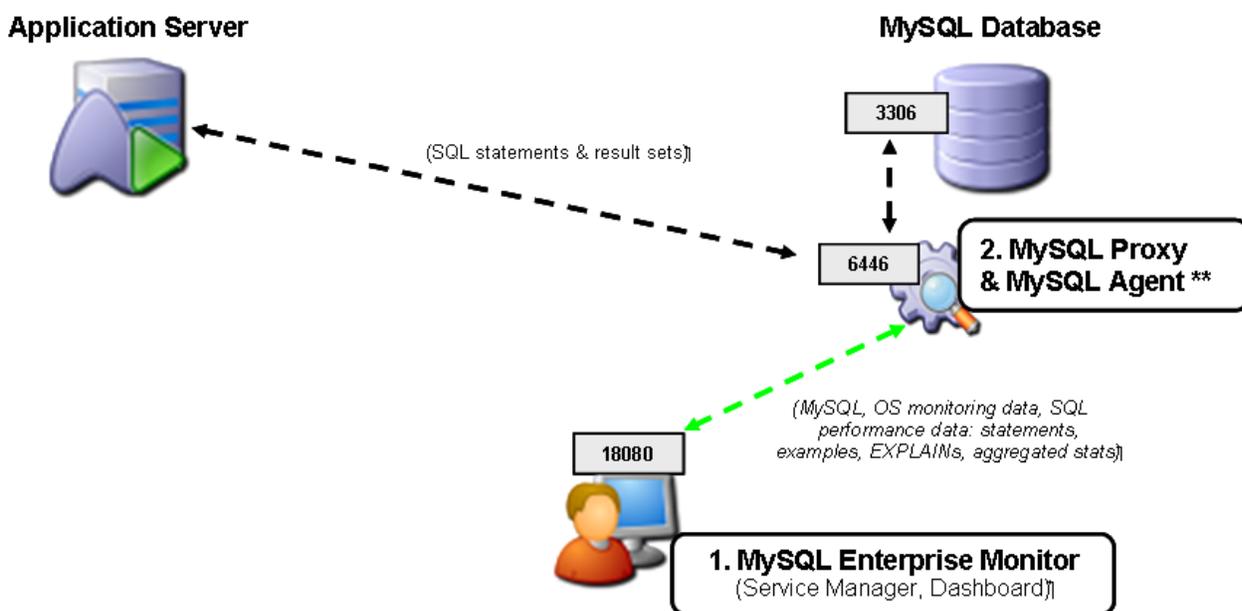


Figure 1 – Query Analyzer enabled using the MySQL Proxy

In this deployment:

- The MySQL Proxy Plugin to the MySQL Agent has been deployed and is configured to listen for application queries on default port 6446.

- The application is connected to default port 6446 which acts as a proxy to the backend MySQL database that is running on port 3306.
- Inbound queries are captured by the proxy before being forwarded to the backend database.
- All outbound result sets are returned to the client or application and query execution statistics, along with other MySQL and OS stats are returned to the Service Manager via default port 18080 for storage, monitoring and analysis by the Query Analyzer and other Enterprise Monitor features.

### Monitoring Queries using Connector/J

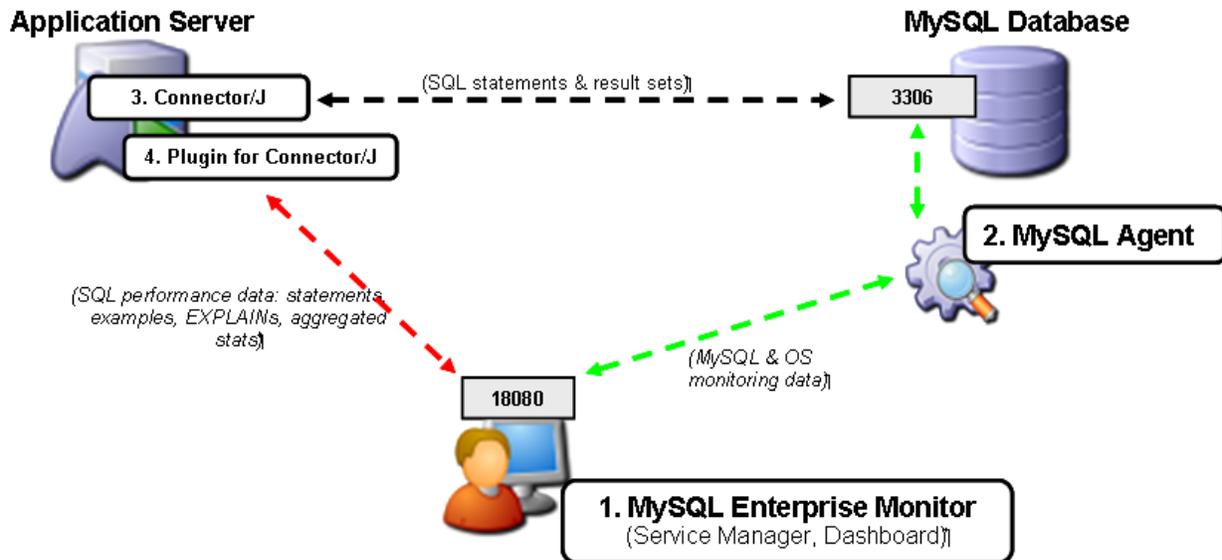


Figure 2 – Query Analyzer enabled using Connector/J

In this deployment:

- The MySQL Enterprise Plugin for Connector/J has been deployed to the application server tier and is configured to work with Connector/J to collect application queries and related performance data.
- The MySQL Agent has been deployed to the monitored MySQL server and is connected to port 3306 to collect MySQL and OS specific performance metrics.
- The Plugin for Connector/J collects and returns all query content, performance statistics, examples, EXPLAIN and aggregated statistics to the Service Manager via default port 18080 for storage, monitoring and analysis using the Query Analyzer. All query result sets are returned to the client or application.
- The MySQL Agent returns all MySQL and OS specific performance metrics to the Service Manager via default port 18080 for storage and analysis by other Enterprise Monitor features.

All query result sets are returned to the client or application

### Monitoring Queries using Connector/NET

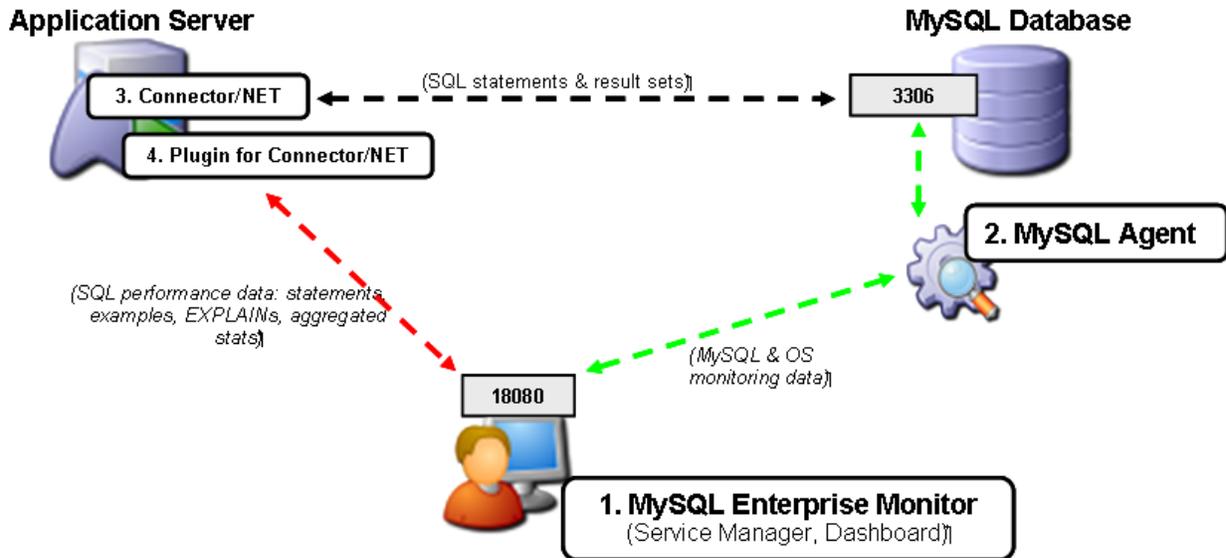


Figure 3 – Query Analyzer enabled using Connector/NET

In this deployment:

- The MySQL Enterprise Plugin for Connector/NET has been deployed to the application server tier and is configured to work with Connector/NET to collect application queries and related performance data.
- The MySQL Agent has been deployed to the monitored MySQL server running on port 3306 and is configured to collect MySQL and OS specific performance metrics.
- The Plugin for Connector/NET collects and returns all query content, performance statistics, examples, EXPLAIN and aggregated statistics to the Service Manager via default port 18080 for storage, monitoring and analysis using the Query Analyzer. All query result sets are returned to the client or application.
- The MySQL Agent returns all MySQL and OS specific performance metrics to the Service Manager via default port 18080 for storage and analysis by other Enterprise Monitor features.