HyperQuery

Quick Start Guide

November 2024

Introduction

HyperQuery is an AI-powered query performance optimization software solution for MySQL databases that radically improves SQL select statement runtimes without monitoring, without adding new or changing existing data structure objects, and without requiring extensive understanding of SQL or Optimizer execution plans. You simply turn HyperQuery on and your queries just run faster. Just as important, HyperQuery returns rewritten fully optimized queries in the same order, therefore, no database changes are needed. While results will vary, we've seen 100s to 1,000s X SQL runtime reductions per query. Since not all queries are problematic, we expect users to experience 10X or better overall. In data warehouse, data lake, and data analytics environments, where queries can be 10s to 100s of billions of rows complex, the sky is truly the limit for potential performance gains.

Architecture

HyperQuery deploys non-intrusively. Specifically, it is not directly involved with the database's SQL optimizer, SQL execution engine, data fetching and returning mechanisms, and in no way sees, touches, or ever copies or moves your actual data. HyperQuery sits alongside the database and performs its optimizations transparently with near zero overhead.

Let us explain.

Normally, the execution of a SQL query is processed as shown below in Figure 1. The query enters the MySQL database and the database optimizer formulates an execution plan. The better the execution plan, the faster the results can be found, fetched, and returned. The database optimizer usually takes only a few milliseconds to choose the execution plan. It does not try to correct poorly written SQL or its effects on the execution plan. Moreover, even when a particular query is run again, the optimizer might generate a *different* execution plan due to updated database statistics, data sizes crossing some threshold, data skew, and numerous other unforeseeable factors. Basically, it works well until it doesn't, and that's the trouble. Plus, reading – and interpreting/understanding execution plans and tuning SQL to result in better execution plans is hard.

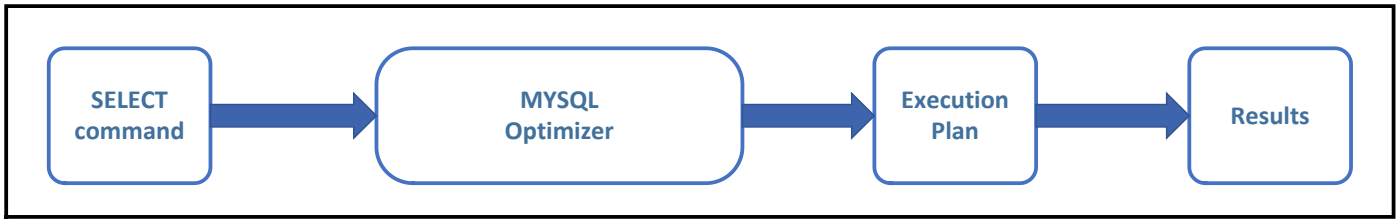


Figure 1: SQL processing without HyperQuery

QIKR solved this basic shortcoming by adding novel artificial intelligence (AI) and machine learning (ML) techniques into HyperQuery's design to help the database's own optimizer run only superior execution plans. It accomplishes this by examining the database's *"slow query"* log, identifying problematic queries guaranteed resulting in poor performance, creating an alternate query which will return the same data in the same order, and placing that rewrite in the database's ***"query rewrite"*** cache.

So now, when a query enters the database, only one additional step occurs – the optimizer looks to see if HyperQuery has already found a better rewrite and then the optimizer simply gets the execution plan for that rewrite.

This whole process still occurs in just a few milliseconds. HyperQuery merely runs at user defined time intervals to examine the slow query log and post known better rewrites to the query rewrite cache. This process takes just a few seconds and can be scheduled to run say once every 4-6 hours. You should not be able to measure any real overhead.

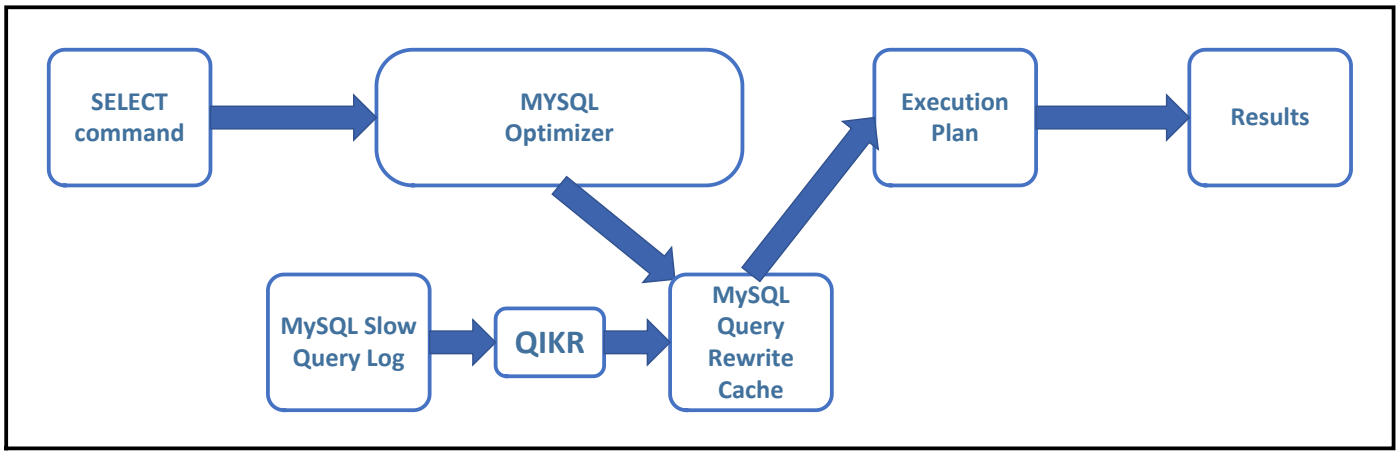


Figure 2: SQL processing with HyperQuery

Deployment

By your choice, and simply requiring intermittent network connectivity to the database whose performance it's to improve HyperQuery can be deployed in numerous ways:

- whether databases are deployed on-premises or cloud
- whether running on Windows or Linux
- on any physical server, virtual machine, container, or cloud image which can connect to the database to be improved.

That's it. There is no requirement for HyperQuery to be installed on the actual database server itself, nor to be installed on the same operating system (OS) as the database, thus, your MySQL database(s) could be on Linux with HyperQuery on either Windows or Linux.

Installation

HyperQuery can be installed on any version of Windows (desktop or server) and on Linux with Mono installed. There are five (5) basic steps which must be performed:

1. configure the database
2. install the HyperQuery software
3. define the database connection credentials
4. schedule how often HyperQuery should run to optimize queries (i.e. 2x daily, 4x daily, always on)
5. start the HyperQuery daemon/service

1. Configure the MySQL database:

A. Enable the MySQL "Slow Queries Log"

Edit the MySQL "my.ini" database configuration file to record all the slow queries into a table (i.e. where slow equals those queries whose run time exceeds some time limit that you define for your database):

- log-output=TABLE
- slow-query-log=1
- long_query_time=30 (choose this value based upon your observations and concerns)

B. Enable the MySQL "Query Rewrite Cache"

Install the MySQL query rewrite cache plug-in (you may need to get the script from the MySQL zip file):

- mysql -u root -p < install_rewriter.sql
- mysql> SHOW GLOBAL VARIABLES LIKE 'rewriter_enabled';

Edit the MySQL "my.ini" database configuration file to enable the MySQL rewriter plug-in:

- rewriter_enabled=ON

C. Reload the MySQL Configuration Settings

Linux:

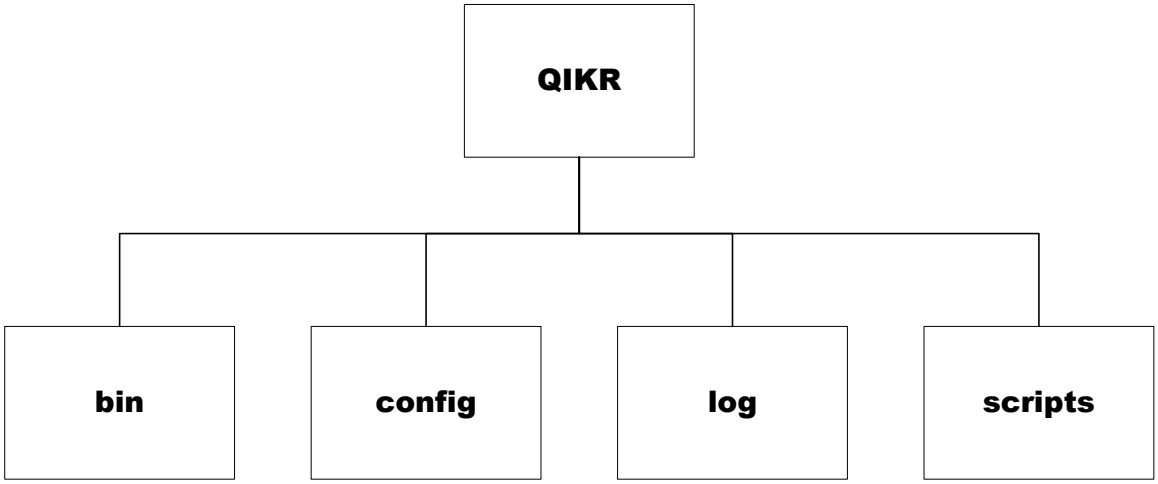
- sudo /etc/init.d/mysql stop
- sudo /etc/init.d/mysql start

Windows:

- net stop <MySQL instance Name>
- net start <MySQL instance Name>

2. Install HyperQuery

HyperQuery is delivered as a .zip file that simply needs to be unzipped into the directory you want HyperQuery to reside. It creates the following directory structure:



- **Linux:**

  Verify that MONO version 5.18 or greater is installed using the following Linux command:

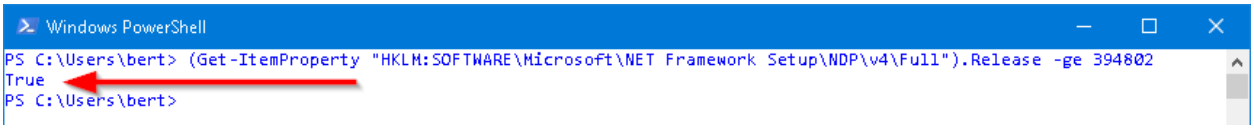
  - mono –version

Download the HyperQuery.zip file and unzip it into a directory that you define (e.g. /opt/HyperQuery).

- open Linux command line
- cd /opt
- mkdir QIKR
- download QIKR.zip
- unzip QIKR.zip into /opt/QIKR
- mono /opt/QIKR/bin/QIKR-HyperQuery.configurator.exe

- **Windows:**

  Verify that .NET Framework 4.6.2 or greater is installed using the following PowerShell command: (Get-ItemProperty "HKLM:SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full").Release -ge 394802



Download the HyperQuery.zip file and unzip it into a directory that you define (e.g. C:\Program Files\HyperQuery).

- open DOS command line
- cd C:\Program Files
- mkdir QIKR
- download QIKR.zip
- unzip QIKR.zip into C:\Program Files
- C:\Program Files\QIKR \bin\ QIKR-HyperQuery.configurator.exe

3. Define database connection credentials

HyperQuery requires the ability to connect to the target database and access to just two (2) system tables:

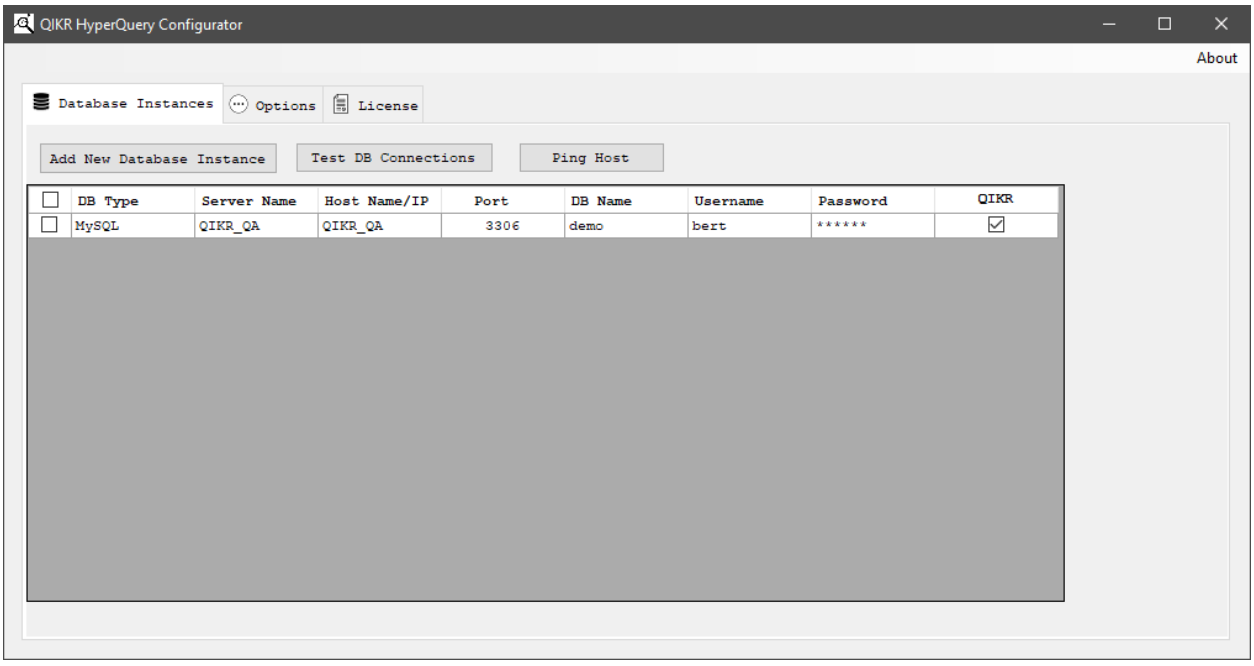
- mysql.slow_log [read access]
- query_rewrite.rewrite_rules [read / write access and the ability to flush]

You may either create a special HyperQuery user account with just these limited privileges, or you may connect with your existing DBA user account [e.g. root]. HyperQuery also requires the MySQL system privilege to flush the query rewrite table containing the rewrite rules. Here are the required grants:

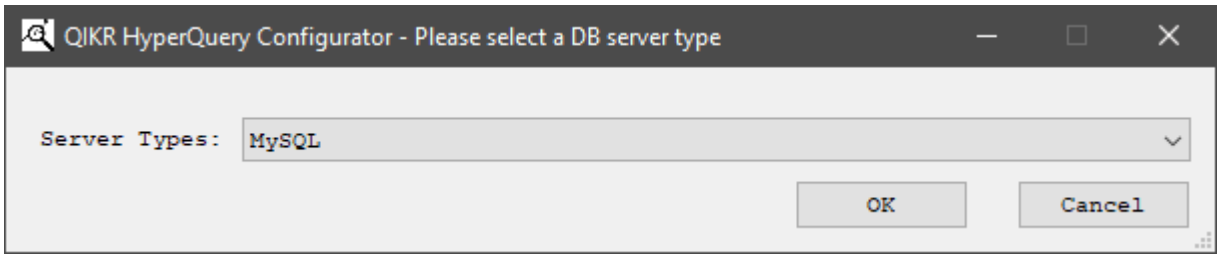
- grant select on mysql.slow_log to 'user'@'%';
- grant select on performance_schema.global_variables to 'user'@'%';
- grant select, insert, delete on query_rewrite.rewrite_rules to 'user'@'%';
- grant execute on procedure query_rewrite.flush_rewrite_rules to 'user'@'%';

HyperQuery has a simple configuration GUI tool for both defining the databases to be optimized and how often. The process is simple and detailed below.

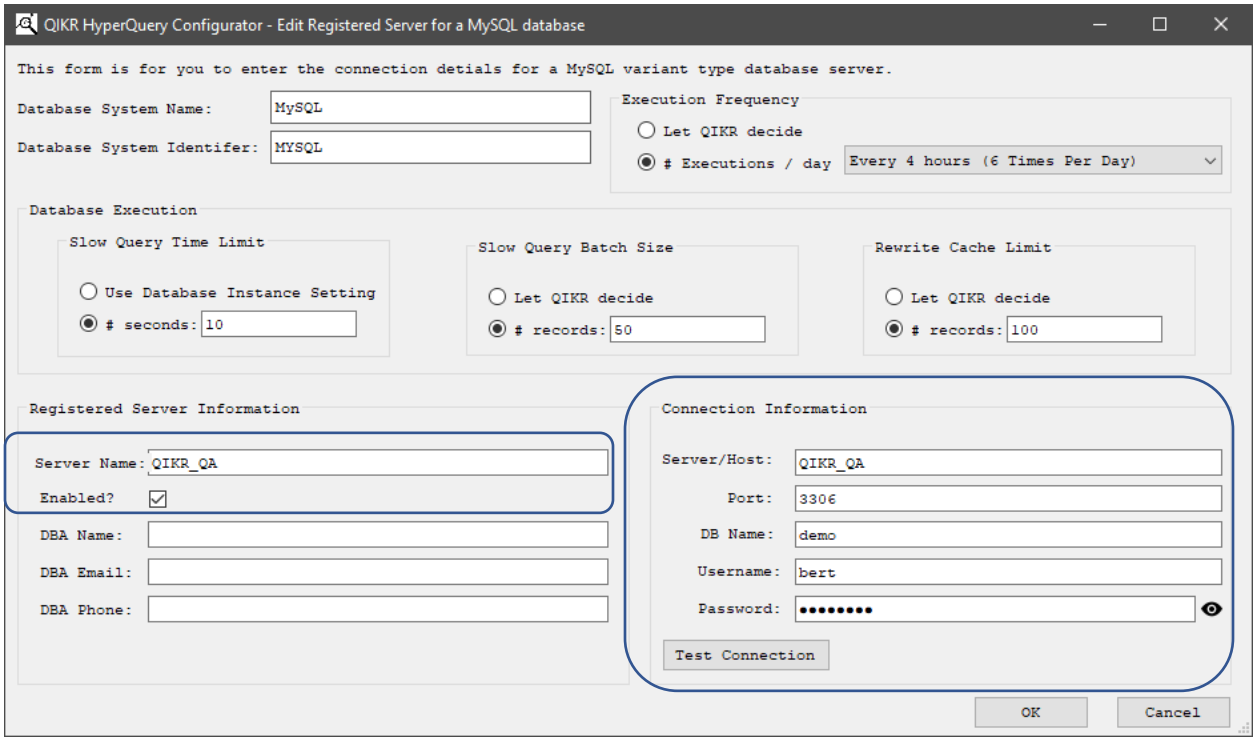
When you run QIKR-HyperQuery.configurator.exe, it launches the utility shown here. You can create a new database connection definition by simply clicking the "***Add New Database Instance***" button.



Doing so opens the following screen, where you define what type of database you are targeting. For now, the only viable option is to choose "MySQL" or one of its variants. In future releases, QIKR will offer additional database platforms, but for now, MySQL is the only choice.



Once you selected, the following screen is displayed where you can then define the database connection credentials and how often it should run. For now, let's focus on just the database connection aspects.



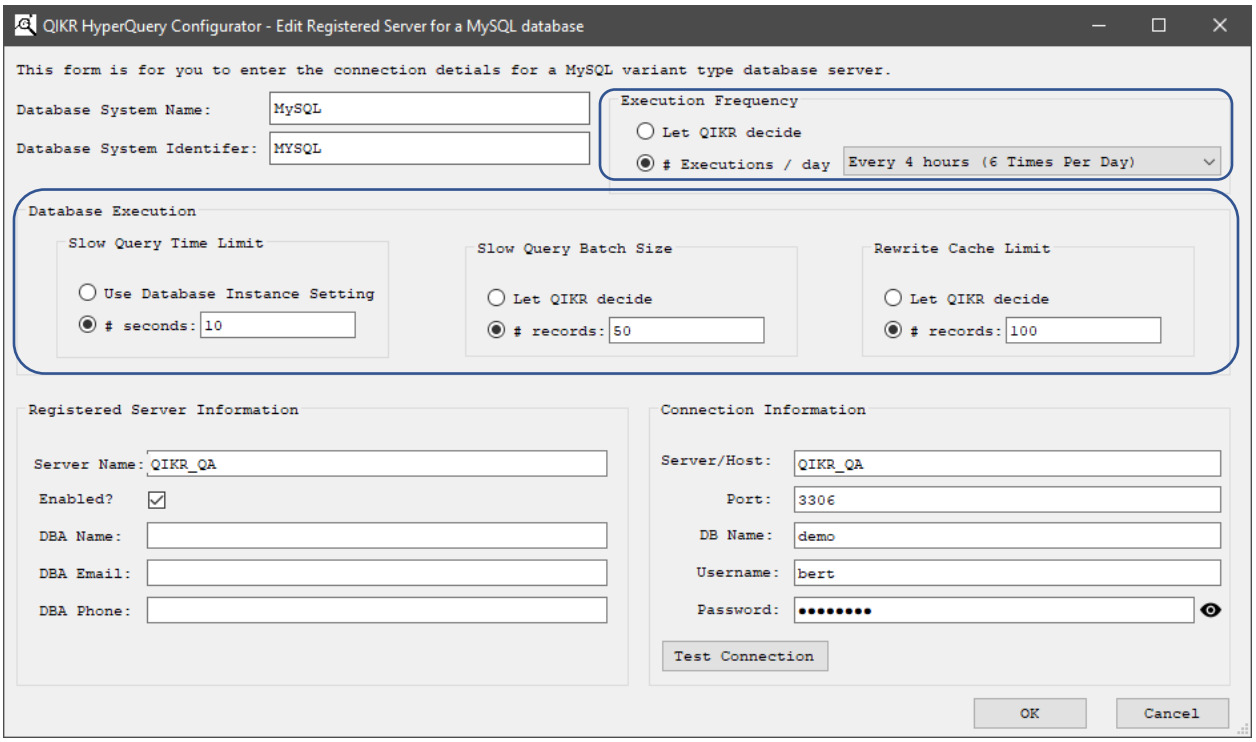
The two highlighted areas allow you to name this database connection, whether it's HyperQuery enabled, and what the connection information for host, port, database, user, and password are. You should then click the "***Test Connection***" button to verify that the information is correct.

4. Schedule QIKR to run at desired intervals

HyperQuery only needs to run occasionally, not continuously, and when it runs, it only runs for a few seconds. The key decision you need to make is how often do you want HyperQuery to check the Slow SQL Log for previously unseen, inefficient SQL that could be improved. A reasonable starting point would be to run HyperQuery every 4 to 6 hours.

NOTE: once HyperQuery has seen an inefficient SQL query and records the rewrite in the query rewrite cache, that query is now automatically handled by the database optimizer which will always use the recorded query rewrite.

Once you have verified that HyperQuery can successfully connect to your database (the prior step), you now need to define when and how it runs. The "*Execution Frequency*" specifies how often per day you want HyperQuery to run against the current target database. It defaults to once every 4 hours or 6 times per day, which is a good value to start with. If you like, you can set this to as often as every 15 minutes or as nominal as just once per day.



The "***Database Execution***" section allows you to customize HyperQuery's runtime behavior. There are three (3) items which you can allow to default or override with your own custom settings. These values are:

1. ***How long must a query execute before being considered slow by HyperQuery and thus a candidate for being rewritten?*** In most case you should consider setting the value the same as in your MySQL database configuration. But there are times where you may desire the database to log all slow queries at one value, and QIKR to only fix those above a certain point higher than that value.

2. ***How many slow queries should be considered as candidates to be rewritten each time HyperQuery is executed?*** In many cases, that should probably be all the candidates available, however, you can specify HyperQuery only look at the top *N* most inefficient queries.

3. ***How many rewrites is HyperQuery allowed to register with the MySQL rewrite cache?*** In many cases, you will want all rewrites to be posted, however, you may also limit that amount such that the MySQL query optimizer has a relatively small subset to review in order to limit any overhead for such examinations.

5. Launch the QIKR daemon/service

- **Linux:**
  - mono /opt/QIKR/bin/ QIKR-HyperQuery.executable.exe
  - ps -al to verify that the process is up and running
- **Windows:**
  - **Run as an executable**
    - C:\Program Files\QIKR \bin\ QIKR-HyperQuery.executable.exe
  - **Run as a .Net Service**
    - installutil C:\Program Files\QIKR \bin\ QIKR-HyperQuery.service.exe
    - Open windows service manager and you will see the QIKR service

6. Check the HyperQuery log file

As HyperQuery runs, it writes out error, warning, and information messages to its log file contained in the HyperQuery/log directory. The file is dated for when the execution started. Below is an example of what the log file contains:

26 Apr 2020 15:54:21,074 INFO QIKR-HyperQuery.executable.exe - QIKR executable starting...

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - QIKR posted 8 rewrites for database: 'MySQL - CentOS - 5.7'

==========

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Original SQL: select * from demo.t1 where concat(year,'-',month,'-',day) = '2020-03-15'

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - ...

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Rewritten SQL: select * from demo.t1 where year = '2020' and month = '03' and day = '15'

==========

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Original SQL: select * from demo.t1 where strcmp(lname,'Doe')=0

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - ...

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Rewritten SQL: select * from demo.t1 where lname = 'Doe'

==========

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Original SQL: select * from demo.t1 where substr(c7,1,3)='xxx'

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - ...

26 Apr 2020 15:54:22,624 INFO QIKR-HyperQuery.executable.exe - Rewritten SQL: select * from demo.t1 where c7 like 'xxx%'

==========

Frequently Asked Questions

Q: Enabling the slow query log has some overhead and is sometimes considered best practice not to enable it. Why should I enable this feature?

A: HyperQuery needs some way to know which queries are running to slow. You can mitigate any performance concerns by setting the MySQL database to only log those queries above some threshold that you are comfortable with. For instance, logging slow queries that take longer than 10 seconds to run may be far too aggressive for your situation. Thus, you might consider a much higher value such as only when the query takes longer than two minutes to run. As such, you can define and limit the overhead to a value that represents a fair tradeoff for overhead vs. automatic performance optimization.

Q: Enabling the query cache has a well-known, unacceptable overhead, and in fact has been deprecated in MySQL version 8.x. Why does HyperQuery rely on this proven risky technology?

A: HyperQuery ***does not*** rely upon nor use that feature, which has a bad performance impact reputation and is no longer available. HyperQuery instead leverages the optional MySQL "***query rewrite cache***", which differs from the deprecated query cache. The query rewrite cache does not add much overhead unless abused.

Q: Does not having the MySQL optimizer for every query first look to see if a rewrite has been posted add an unacceptable overhead?

A: It could, that's why HyperQuery allows you to specify the maximum number of rewritten SQL queries to be posted. That way you can manage that overhead to whatever you deem acceptable. We would not want to suggest that adding an unlimited or even a very large number of rewrites as a universally good idea. You should set this with caution much like you do with the slow query log threshold.

Q: But doesn't just comparing the SQL text of potentially very long queries by itself simply add too much overhead?

A: Not really. MySQL stores queries with a sort of "hash" value based upon the first 1,000 characters. Thus, by examining the length and this hash value, the lookups can be done quickly and efficiently.