# Optimizing Performance Using Query Results Cache - QRC

When the Query Result Cache (QRC) is enabled, then incoming queries are compared to those in the query cache before parsing. If the incoming query exists in the QRC, then the QRC results are sent. Prepared statements are also supported. The hit rate is generally less than 80% for the QRC as compared to other caches because:

- The QRC requires an identical query to render results
- If a table is modified, all result sets derived from the table in the QRC are invalidated, even if the modification did not changed the result set at all.

Queries must be exactly the same to enable usage of the QRC results.

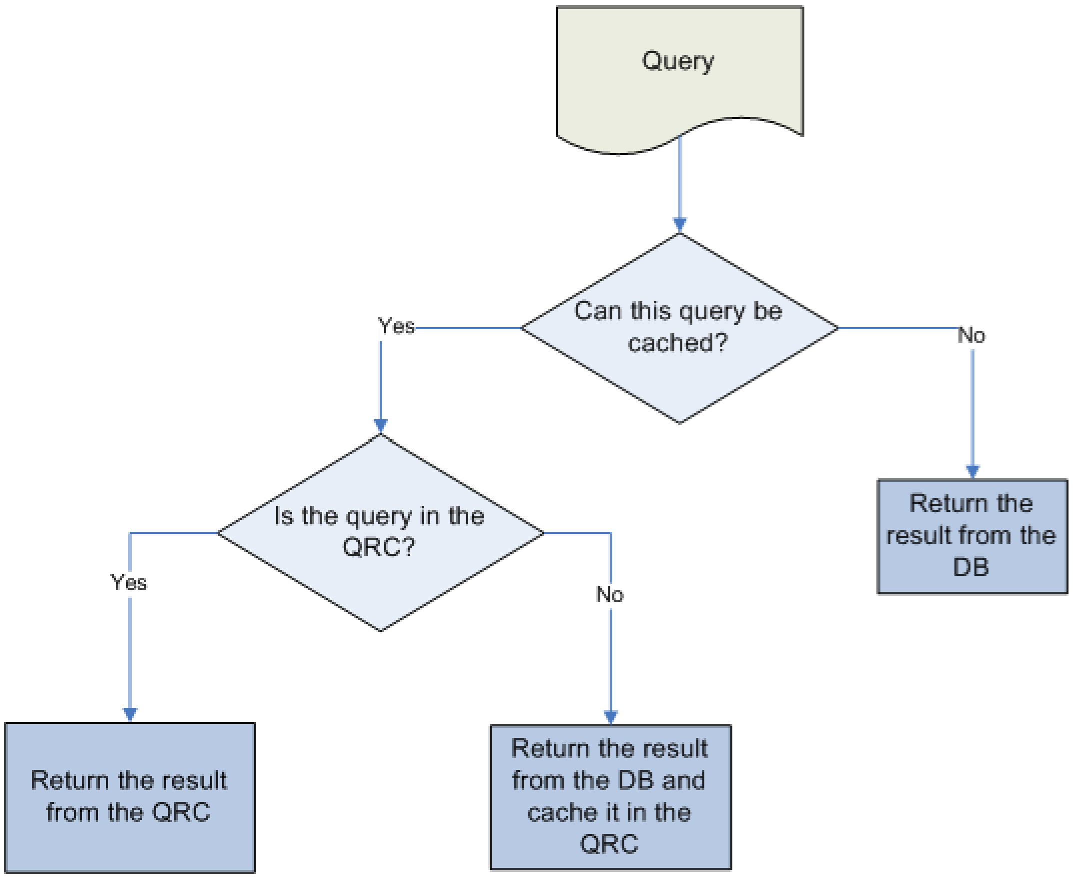
Consider the following two queries:

```
sql> select * from table foo;
sql> SELECT * from table foo;
```

Both of these queries are considered different. Only the QRC results matching the exact same iteration of that SELECT could be used.

## QRC Decision Flow

The following diagram illustrates the QRC decision flow:



Query
Can this query be cached?
Yes
No
Is the query in the QRC?
Yes
No
Return the result from the DB
Return the result from the QRC
Return the result from the DB and cache it in the QRC

## Enabling and Monitoring the QRC

To enable the QRC, issue the following command from a command prompt on the target machine:

```
sql> set global qrc_enabled = true;
```

| Variable | Description | Default |
|---|---|---|
| qrc_enabled | Enables or disables the Query Results Cache. | false |

## Understanding the QRC Statuses

The following QRC-related statuses are available:

- qrc_count - How many queries are cached?
- qrc_hits - How many cached queries have evaluated as matched?
- qrc_misses - How many cached queries have evaluated as unmatched?
- qrc_size - How large in the QRC overall?
- qrc_uncacheable - Is the QRC cacheable?
- uptime - How long has the cache existed?

## Considerations for Enabling the QRC

Choosing to use the QRC depends on your data. If all the queries you are performing are simple (such as selecting a row from a table with one row), but still differ so that the queries cannot be cached, the overhead for having the query cache active is about 10%. This could be regarded as the worst case scenario. In real life, queries tend to be much more complicated, so the overhead normally is significantly lower.

Searches for a single row in a single-row table are two times faster with the query cache enabled than without it. This is typical for a query that is cached.

## What is NOT Cached in QRC?

The following items are not cached:

- Dynamically-generated comments(e.g. ORM), extra spaces, and different cases would result in a QRC miss.
- System tables
- Pure SELECT statements are cached. All other statements, such as SHOW, INSERT, ALTER, etc. are not cached.
- Queries referring to user-defined functions or stored functions
- Queries referring to user variable or local stored program variable
- Queries referring to temporary table
- Queries not using any tables, for example, SELECT 1+1...
- Queries generating warnings

Queries will be cached if they provide exactly the same results from repeat executions and **if the underlying data remains unaltered.** Queries that include any of the following keywords will also not be cached:

- BENCHMARK()
- CONNECTION_ID()
- CONVERT_TZ()
- CURDATE()
- CURRENT_DATE()
- CURRENT_TIME()
- CURRENT_TIMESTAMP()
- CURTIME()
- DATABASE()
- FOUND_ROWS()
- GET_LOCK()
- ENCRYPT() *with one parameter*
- LAST_INSERT_ID()
- LOAD_FILE()
- MASTER_POS_WAIT()
- NOW()
- RAND()
- RELEASE_LOCK()
- SLEEP()
- SYSDATE()
- UNIX_TIMESTAMP() *with no parameters*
- USER()
- UUID()
- UUID_SHORT()