# phpMyAdmin Advisory rules file

#

# Use only UNIX style newlines

#

# This file is being parsed by Advisor.class.php, which should handle syntax

# errors correctly. However, PHP Warnings and the like are being consumed by

# the phpMyAdmin error handler, so those won't show up E.g.: Justification line

# is empty because you used an unescape percent sign, sprintf() returns an

# empty string and no warning/error is shown

#

# Rule Syntax:

# 'rule' identifier[the name of the rule] eexpr [an optional precondition]

# expr [variable or value calculation used for the test]

# expr [test, if evaluted to 'true' it fires the rule. Use 'value' to insert the calculated value (without quotes)]

# string [the issue (what is the problem?)]

# string [the recommendation (how do i fix it?)]

# formatted-string '|' comma-seperated-expr [the justification (result of the calculated value / why did this rule fire?)]

# comma-seperated-expr: expr(,expr)\*

# eexpr: [expr] - expr enclosed in []

# expr: a php code literal with extras:

# - variable names are replaced with their respective values

# - fired('name of rule') is replaced with true/false when given rule has

# been fired. Note however that this is a very simple rules engine.

# Rules are only checked in sequential order as they are written down

# here. If given rule has not been checked yet, fired() will always

# evaluate to false

# - 'value' is replaced with the calculated value. If it is a string, it

# will be put within single quotes

# - other than that you may use any php function, initialized variable or

# constant

#

# identifier: A string enclosed in single quotes

# string: A quoteless string, may contain HTML. Variable names enclosed in

# curly braces are replaced with links to directly edit this variable.

# e.g. {tmp\_table\_size}

# formatted-string: You may use classic php sprintf() string formatting here,

# the arguments must be appended after a trailing pipe (|) as

# mentioned in above syntax percent signs (%) are

# automatically escaped (%%) in the following cases: When

# followed by a space, dot or comma and at the end of the

# line)

#

# Comments start with #

#

# Queries

rule 'Uptime below one day'

Uptime

value < 86400

Uptime is less than 1 day, performance tuning may not be accurate.

To have more accurate averages it is recommended to let the server run for longer than a day before running this analyzer

The uptime is only %s | ADVISOR\_timespanFormat(Uptime)

rule 'Questions below 1,000'

Questions

value < 1000

Fewer than 1,000 questions have been run against this server. The recommendations may not be accurate.

Let the server run for a longer time until it has executed a greater amount of queries.

Current amount of Questions: %s | Questions

rule 'Percentage of slow queries' [Questions > 0 && !PMA\_DRIZZLE]

Slow\_queries / Questions \* 100

value >= 5

There is a lot of slow queries compared to the overall amount of Queries.

You might want to increase {long\_query\_time} or optimize the queries listed in the slow query log

The slow query rate should be below 5%, your value is %s%. | round(value,2)

rule 'Slow query rate' [Questions > 0]

(Slow\_queries / Questions \* 100) / Uptime

value \* 60 \* 60 > 1

There is a high percentage of slow queries compared to the server uptime.

You might want to increase {long\_query\_time} or optimize the queries listed in the slow query log

You have a slow query rate of %s per hour, you should have less than 1% per hour. | ADVISOR\_bytime(value,2)

rule 'Long query time' [!PMA\_DRIZZLE]

long\_query\_time

value >= 10

{long\_query\_time} is set to 10 seconds or more, thus only slow queries that take above 10 seconds are logged.

It is suggested to set {long\_query\_time} to a lower value, depending on your environment. Usually a value of 1-5 seconds is suggested.

long\_query\_time is currently set to %ds. | value

rule 'Slow query logging' [!PMA\_DRIZZLE && PMA\_MYSQL\_INT\_VERSION < 50600]

log\_slow\_queries

value == 'OFF'

The slow query log is disabled.

Enable slow query logging by setting {log\_slow\_queries} to 'ON'. This will help troubleshooting badly performing queries.

log\_slow\_queries is set to 'OFF'

rule 'Slow query logging' [!PMA\_DRIZZLE && PMA\_MYSQL\_INT\_VERSION >= 50600]

slow\_query\_log

value == 'OFF'

The slow query log is disabled.

Enable slow query logging by setting {slow\_query\_log} to 'ON'. This will help troubleshooting badly performing queries.

slow\_query\_log is set to 'OFF'

#

# versions

rule 'Release Series' [!PMA\_DRIZZLE]

version

substr(value,0,2) <= '5.' && substr(value,2,1) < 1

The MySQL server version less than 5.1.

You should upgrade, as MySQL 5.1 has improved performance, and MySQL 5.5 even more so.

Current version: %s | value

rule 'Minor Version' [! fired('Release Series')]

version

substr(value,0,2) <= '5.' && substr(value,2,1) <= 1 && substr(value,4,2) < 30

Version less than 5.1.30 (the first GA release of 5.1).

You should upgrade, as recent versions of MySQL 5.1 have improved performance and MySQL 5.5 even more so.

Current version: %s | value

rule 'Minor Version' [! fired('Release Series')]

version

substr(value,0,1) == 5 && substr(value,2,1) == 5 && substr(value,4,2) < 8

Version less than 5.5.8 (the first GA release of 5.5).

You should upgrade, to a stable version of MySQL 5.5

Current version: %s | value

rule 'Distribution'

version\_comment

preg\_match('/source/i',value)

Version is compiled from source, not a MySQL official binary.

If you did not compile from source, you may be using a package modified by a distribution. The MySQL manual only is accurate for official MySQL binaries, not any package distributions (such as RedHat, Debian/Ubuntu etc).

'source' found in version\_comment

rule 'Distribution'

version\_comment

preg\_match('/percona/i',value)

The MySQL manual only is accurate for official MySQL binaries.

Percona documentation is at <a href="http://www.percona.com/software/documentation/">http://www.percona.com/software/documentation/</a>

'percona' found in version\_comment

rule 'Distribution'

version

PMA\_DRIZZLE

The MySQL manual only is accurate for official MySQL binaries.

Drizzle documentation is at <a href="http://www.drizzle.org/content/documentation/">http://www.drizzle.org/content/documentation/</a>

Version string (%s) matches Drizzle versioning scheme | value

rule 'MySQL Architecture'

system\_memory

value > 3072\*1024 && !preg\_match('/64/',version\_compile\_machine) && !preg\_match('/64/',version\_compile\_os)

MySQL is not compiled as a 64-bit package.

Your memory capacity is above 3 GiB (assuming the Server is on localhost), so MySQL might not be able to access all of your memory. You might want to consider installing the 64-bit version of MySQL.

Available memory on this host: %s | ADVISOR\_formatByteDown(value\*1024, 2, 2)

#

# Query cache

# Lame: 'ON' == 0 is true, so you need to compare 'ON' == '0'

rule 'Query cache disabled' [!PMA\_DRIZZLE]

query\_cache\_size

value == 0 || query\_cache\_type == 'OFF' || query\_cache\_type == '0'

The query cache is not enabled.

The query cache is known to greatly improve performance if configured correctly. Enable it by setting {query\_cache\_size} to a 2 digit MiB value and setting {query\_cache\_type} to 'ON'. <b>Note:</b> If you are using memcached, ignore this recommendation.

query\_cache\_size is set to 0 or query\_cache\_type is set to 'OFF'

rule 'Query caching method' [!fired('Query cache disabled')]

Questions / Uptime

value > 100

Suboptimal caching method.

You are using the MySQL Query cache with a fairly high traffic database. It might be worth considering to use <a href="http://dev.mysql.com/doc/refman/5.5/en/ha-memcached.html">memcached</a> instead of the MySQL Query cache, especially if you have multiple slaves.

The query cache is enabled and the server receives %d queries per second. This rule fires if there is more than 100 queries per second. | round(value,1)

rule 'Query cache efficiency (%)' [!PMA\_DRIZZLE && Com\_select + Qcache\_hits > 0 && !fired('Query cache disabled')]

Qcache\_hits / (Com\_select + Qcache\_hits) \* 100

value < 20

Query cache not running efficiently, it has a low hit rate.

Consider increasing {query\_cache\_limit}.

The current query cache hit rate of %s% is below 20% | round(value,1)

rule 'Query Cache usage' [!fired('Query cache disabled') && !PMA\_DRIZZLE]

100 - Qcache\_free\_memory / query\_cache\_size \* 100

value < 80

Less than 80% of the query cache is being utilized.

This might be caused by {query\_cache\_limit} being too low. Flushing the query cache might help as well.

The current ratio of free query cache memory to total query cache size is %s%. It should be above 80% | round(value,1)

rule 'Query cache fragmentation' [!fired('Query cache disabled') && !PMA\_DRIZZLE]

Qcache\_free\_blocks / (Qcache\_total\_blocks / 2) \* 100

value > 20

The query cache is considerably fragmented.

Severe fragmentation is likely to (further) increase Qcache\_lowmem\_prunes. This might be caused by many Query cache low memory prunes due to {query\_cache\_size} being too small. For a immediate but short lived fix you can flush the query cache (might lock the query cache for a long time). Carefully adjusting {query\_cache\_min\_res\_unit} to a lower value might help too, e.g. you can set it to the average size of your queries in the cache using this formula: (query\_cache\_size - qcache\_free\_memory) / qcache\_queries\_in\_cache

The cache is currently fragmented by %s% , with 100% fragmentation meaning that the query cache is an alternating pattern of free and used blocks. This value should be below 20%. | round(value,1)

rule 'Query cache low memory prunes' [!PMA\_DRIZZLE && Qcache\_inserts > 0 && !fired('Query cache disabled')]

Qcache\_lowmem\_prunes / Qcache\_inserts \* 100

value > 0.1

Cached queries are removed due to low query cache memory from the query cache.

You might want to increase {query\_cache\_size}, however keep in mind that the overhead of maintaining the cache is likely to increase with its size, so do this in small increments and monitor the results.

The ratio of removed queries to inserted queries is %s%. The lower this value is, the better (This rules firing limit: 0.1%) | round(value,1)

rule 'Query cache max size' [!fired('Query cache disabled')]

query\_cache\_size

value > 1024 \* 1024 \* 128

The query cache size is above 128 MiB. Big query caches may cause significant overhead that is required to maintain the cache.

Depending on your environment, it might be performance increasing to reduce this value.

Current query cache size: %s | ADVISOR\_formatByteDown(value, 2, 2)

rule 'Query cache min result size' [!fired('Query cache disabled')]

query\_cache\_limit

value == 1024\*1024

The max size of the result set in the query cache is the default of 1 MiB.

Changing {query\_cache\_limit} (usually by increasing) may increase efficiency. This variable determines the maximum size a query result may have to be inserted into the query cache. If there are many query results above 1 MiB that are well cacheable (many reads, little writes) then increasing {query\_cache\_limit} will increase efficiency. Whereas in the case of many query results being above 1 MiB that are not very well cacheable (often invalidated due to table updates) increasing {query\_cache\_limit} might reduce efficiency.

query\_cache\_limit is set to 1 MiB

#

# Sorts

rule 'Percentage of sorts that cause temporary tables' [Sort\_scan + Sort\_range > 0]

Sort\_merge\_passes / (Sort\_scan + Sort\_range) \* 100

value > 10

Too many sorts are causing temporary tables.

Consider increasing {sort\_buffer\_size} and/or {read\_rnd\_buffer\_size}, depending on your system memory limits

%s% of all sorts cause temporary tables, this value should be lower than 10%. | round(value,1)

rule 'Rate of sorts that cause temporary tables'

Sort\_merge\_passes / Uptime

value \* 60 \* 60 > 1

Too many sorts are causing temporary tables.

Consider increasing {sort\_buffer\_size} and/or {read\_rnd\_buffer\_size}, depending on your system memory limits

Temporary tables average: %s, this value should be less than 1 per hour. | ADVISOR\_bytime(value,2)

rule 'Sort rows'

Sort\_rows / Uptime

value \* 60 >= 1

There are lots of rows being sorted.

While there is nothing wrong with a high amount of row sorting, you might want to make sure that the queries which require a lot of sorting use indexed columns in the ORDER BY clause, as this will result in much faster sorting

Sorted rows average: %s | ADVISOR\_bytime(value,2)

# Joins, scans

rule 'Rate of joins without indexes'

(Select\_range\_check + Select\_scan + Select\_full\_join) / Uptime

value \* 60 \* 60 > 1

There are too many joins without indexes.

This means that joins are doing full table scans. Adding indexes for the columns being used in the join conditions will greatly speed up table joins

Table joins average: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

rule 'Rate of reading first index entry'

Handler\_read\_first / Uptime

value \* 60 \* 60 > 1

The rate of reading the first index entry is high.

This usually indicates frequent full index scans. Full index scans are faster than table scans but require lots of CPU cycles in big tables, if those tables that have or had high volumes of UPDATEs and DELETEs, running 'OPTIMIZE TABLE' might reduce the amount of and/or speed up full index scans. Other than that full index scans can only be reduced by rewriting queries.

Index scans average: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

rule 'Rate of reading fixed position'

Handler\_read\_rnd / Uptime

value \* 60 \* 60 > 1

The rate of reading data from a fixed position is high.

This indicates that many queries need to sort results and/or do a full table scan, including join queries that do not use indexes. Add indexes where applicable.

Rate of reading fixed position average: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

rule 'Rate of reading next table row'

Handler\_read\_rnd\_next / Uptime

value \* 60 \* 60 > 1

The rate of reading the next table row is high.

This indicates that many queries are doing full table scans. Add indexes where applicable.

Rate of reading next table row: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

# temp tables

rule 'Different tmp\_table\_size and max\_heap\_table\_size'

tmp\_table\_size - max\_heap\_table\_size

value !=0

{tmp\_table\_size} and {max\_heap\_table\_size} are not the same.

If you have deliberately changed one of either: The server uses the lower value of either to determine the maximum size of in-memory tables. So if you wish to increase the in-memory table limit you will have to increase the other value as well.

Current values are tmp\_table\_size: %s, max\_heap\_table\_size: %s | ADVISOR\_formatByteDown(tmp\_table\_size, 2, 2), ADVISOR\_formatByteDown(max\_heap\_table\_size, 2, 2)

rule 'Percentage of temp tables on disk' [Created\_tmp\_tables + Created\_tmp\_disk\_tables > 0]

Created\_tmp\_disk\_tables / (Created\_tmp\_tables + Created\_tmp\_disk\_tables) \* 100

value > 25

Many temporary tables are being written to disk instead of being kept in memory.

Increasing {max\_heap\_table\_size} and {tmp\_table\_size} might help. However some temporary tables are always being written to disk, independent of the value of these variables. To eliminate these you will have to rewrite your queries to avoid those conditions (Within a temporary table: Presence of a BLOB or TEXT column or presence of a column bigger than 512 bytes) as mentioned in the beginning of an <a href="http://www.facebook.com/note.php?note\_id=10150111255065841&comments">Article by the Pythian Group</a>

%s% of all temporary tables are being written to disk, this value should be below 25% | round(value,1)

rule 'Temp disk rate' [!fired('Percentage of temp tables on disk')]

Created\_tmp\_disk\_tables / Uptime

value \* 60 \* 60 > 1

Many temporary tables are being written to disk instead of being kept in memory.

Increasing {max\_heap\_table\_size} and {tmp\_table\_size} might help. However some temporary tables are always being written to disk, independent of the value of these variables. To eliminate these you will have to rewrite your queries to avoid those conditions (Within a temporary table: Presence of a BLOB or TEXT column or presence of a column bigger than 512 bytes) as mentioned in the <a href="http://dev.mysql.com/doc/refman/5.5/en/internal-temporary-tables.html">MySQL Documentation</a>

Rate of temporary tables being written to disk: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

# I couldn't find any source on the internet that suggests a direct relation between high counts of temporary tables and any of these variables.

# Several independent Blog entries suggest (http://ronaldbradford.com/blog/more-on-understanding-sort\_buffer\_size-2010-05-10/ and http://www.xaprb.com/blog/2010/05/09/how-to-tune-mysqls-sort\_buffer\_size/)

# that sort\_buffer\_size should be left as it is. And increasing read\_buffer\_size is only suggested when there are a lot of

# table scans (http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html#sysvar\_read\_buffer\_size and other sources) though

# setting it too high is bad too (http://www.percona.com/blog/2007/09/17/mysql-what-read\_buffer\_size-value-is-optimal/).

#rule 'Temp table rate'

# Created\_tmp\_tables / Uptime

# value \* 60 \* 60 > 1

# Many intermediate temporary tables are being created.

# This may be caused by queries under certain conditions as mentioned in the <a href="http://dev.mysql.com/doc/refman/5.5/en/internal-temporary-tables.html">MySQL Documentation</a>. Consider increasing {sort\_buffer\_size} (sorting), {read\_rnd\_buffer\_size} (random read buffer, ie, post-sort), {read\_buffer\_size} (sequential scan).

#

# MyISAM index cache

rule 'MyISAM key buffer size' [!PMA\_DRIZZLE]

key\_buffer\_size

value == 0

Key buffer is not initialized. No MyISAM indexes will be cached.

Set {key\_buffer\_size} depending on the size of your MyISAM indexes. 64M is a good start.

key\_buffer\_size is 0

rule 'Max % MyISAM key buffer ever used' [!PMA\_DRIZZLE && key\_buffer\_size > 0]

Key\_blocks\_used \* key\_cache\_block\_size / key\_buffer\_size \* 100

value < 95

MyISAM key buffer (index cache) % used is low.

You may need to decrease the size of {key\_buffer\_size}, re-examine your tables to see if indexes have been removed, or examine queries and expectations about what indexes are being used.

max % MyISAM key buffer ever used: %s%, this value should be above 95% | round(value,1)

# Don't fire if above rule fired - we don't need the same advice twice

rule 'Percentage of MyISAM key buffer used' [!PMA\_DRIZZLE && key\_buffer\_size > 0 && !fired('Max % MyISAM key buffer ever used')]

( 1 - Key\_blocks\_unused \* key\_cache\_block\_size / key\_buffer\_size) \* 100

value < 95

MyISAM key buffer (index cache) % used is low.

You may need to decrease the size of {key\_buffer\_size}, re-examine your tables to see if indexes have been removed, or examine queries and expectations about what indexes are being used.

% MyISAM key buffer used: %s%, this value should be above 95% | round(value,1)

rule 'Percentage of index reads from memory' [!PMA\_DRIZZLE && Key\_read\_requests > 0]

100 - (Key\_reads / Key\_read\_requests \* 100)

value < 95

The % of indexes that use the MyISAM key buffer is low.

You may need to increase {key\_buffer\_size}.

Index reads from memory: %s%, this value should be above 95% | round(value,1)

#

# other caches

rule 'Rate of table open' [!PMA\_DRIZZLE]

Opened\_tables / Uptime

value\*60\*60 > 10

The rate of opening tables is high.

Opening tables requires disk I/O which is costly. Increasing {table\_open\_cache} might avoid this.

Opened table rate: %s, this value should be less than 10 per hour | ADVISOR\_bytime(value,2)

rule 'Percentage of used open files limit' [!PMA\_DRIZZLE]

Open\_files / open\_files\_limit \* 100

value > 85

The number of open files is approaching the max number of open files. You may get a "Too many open files" error.

Consider increasing {open\_files\_limit}, and check the error log when restarting after changing {open\_files\_limit}.

The number of opened files is at %s% of the limit. It should be below 85% | round(value,1)

rule 'Rate of open files' [!PMA\_DRIZZLE]

Open\_files / Uptime

value \* 60 \* 60 > 5

The rate of opening files is high.

Consider increasing {open\_files\_limit}, and check the error log when restarting after changing {open\_files\_limit}.

Opened files rate: %s, this value should be less than 5 per hour | ADVISOR\_bytime(value,2)

rule 'Immediate table locks %' [Table\_locks\_waited + Table\_locks\_immediate > 0]

Table\_locks\_immediate / (Table\_locks\_waited + Table\_locks\_immediate) \* 100

value < 95

Too many table locks were not granted immediately.

Optimize queries and/or use InnoDB to reduce lock wait.

Immediate table locks: %s%, this value should be above 95% | round(value,1)

rule 'Table lock wait rate'

Table\_locks\_waited / Uptime

value \* 60 \* 60 > 1

Too many table locks were not granted immediately.

Optimize queries and/or use InnoDB to reduce lock wait.

Table lock wait rate: %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

rule 'Thread cache' [!PMA\_DRIZZLE]

thread\_cache\_size

value < 1

Thread cache is disabled, resulting in more overhead from new connections to MySQL.

Enable the thread cache by setting {thread\_cache\_size} > 0.

The thread cache is set to 0

rule 'Thread cache hit rate %' [!PMA\_DRIZZLE && thread\_cache\_size > 0]

100 - Threads\_created / Connections

value < 80

Thread cache is not efficient.

Increase {thread\_cache\_size}.

Thread cache hitrate: %s%, this value should be above 80% | round(value,1)

rule 'Threads that are slow to launch' [!PMA\_DRIZZLE && slow\_launch\_time > 0]

Slow\_launch\_threads

value > 0

There are too many threads that are slow to launch.

This generally happens in case of general system overload as it is pretty simple operations. You might want to monitor your system load carefully.

%s thread(s) took longer than %s seconds to start, it should be 0 | value, slow\_launch\_time

rule 'Slow launch time' [!PMA\_DRIZZLE]

slow\_launch\_time

value > 2

Slow\_launch\_threads is above 2s

Set {slow\_launch\_time} to 1s or 2s to correctly count threads that are slow to launch

slow\_launch\_time is set to %s | value

#

#Connections

rule 'Percentage of used connections' [!PMA\_DRIZZLE]

Max\_used\_connections / max\_connections \* 100

value > 80

The maximum amount of used connections is getting close to the value of {max\_connections}.

Increase {max\_connections}, or decrease {wait\_timeout} so that connections that do not close database handlers properly get killed sooner. Make sure the code closes database handlers properly.

Max\_used\_connections is at %s% of max\_connections, it should be below 80% | round(value,1)

rule 'Percentage of aborted connections'

Aborted\_connects / Connections \* 100

value > 1

Too many connections are aborted.

Connections are usually aborted when they cannot be authorized. <a href="http://www.percona.com/blog/2008/08/23/how-to-track-down-the-source-of-aborted\_connects/">This article</a> might help you track down the source.

%s% of all connections are aborted. This value should be below 1% | round(value,1)

rule 'Rate of aborted connections'

Aborted\_connects / Uptime

value \* 60 \* 60 > 1

Too many connections are aborted.

Connections are usually aborted when they cannot be authorized. <a href="http://www.percona.com/blog/2008/08/23/how-to-track-down-the-source-of-aborted\_connects/">This article</a> might help you track down the source.

Aborted connections rate is at %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

rule 'Percentage of aborted clients'

Aborted\_clients / Connections \* 100

value > 2

Too many clients are aborted.

Clients are usually aborted when they did not close their connection to MySQL properly. This can be due to network issues or code not closing a database handler properly. Check your network and code.

%s% of all clients are aborted. This value should be below 2% | round(value,1)

rule 'Rate of aborted clients'

Aborted\_clients / Uptime

value \* 60 \* 60 > 1

Too many clients are aborted.

Clients are usually aborted when they did not close their connection to MySQL properly. This can be due to network issues or code not closing a database handler properly. Check your network and code.

Aborted client rate is at %s, this value should be less than 1 per hour | ADVISOR\_bytime(value,2)

#

# InnoDB

rule 'Is InnoDB disabled?' [!PMA\_DRIZZLE && PMA\_MYSQL\_INT\_VERSION < 50600]

have\_innodb

value != "YES"

You do not have InnoDB enabled.

InnoDB is usually the better choice for table engines.

have\_innodb is set to 'value'

rule 'InnoDB log size' [innodb\_buffer\_pool\_size > 0]

innodb\_log\_file\_size / innodb\_buffer\_pool\_size \* 100

value < 20 && innodb\_log\_file\_size / (1024 \* 1024) < 256

The InnoDB log file size is not an appropriate size, in relation to the InnoDB buffer pool.

Especially on a system with a lot of writes to InnoDB tables you should set {innodb\_log\_file\_size} to 25% of {innodb\_buffer\_pool\_size}. However the bigger this value, the longer the recovery time will be when database crashes, so this value should not be set much higher than 256 MiB. Please note however that you cannot simply change the value of this variable. You need to shutdown the server, remove the InnoDB log files, set the new value in my.cnf, start the server, then check the error logs if everything went fine. See also <a href="http://mysqldatabaseadministration.blogspot.com/2007/01/increase-innodblogfilesize-proper-way.html">this blog entry</a>

Your InnoDB log size is at %s% in relation to the InnoDB buffer pool size, it should not be below 20% | round(value,1)

rule 'Max InnoDB log size' [innodb\_buffer\_pool\_size > 0 && innodb\_log\_file\_size / innodb\_buffer\_pool\_size \* 100 < 30]

innodb\_log\_file\_size / (1024 \* 1024)

value > 256

The InnoDB log file size is inadequately large.

It is usually sufficient to set {innodb\_log\_file\_size} to 25% of the size of {innodb\_buffer\_pool\_size}. A very big {innodb\_log\_file\_size} slows down the recovery time after a database crash considerably. See also <a href="http://www.percona.com/blog/2006/07/03/choosing-proper-innodb\_log\_file\_size/">this Article</a>. You need to shutdown the server, remove the InnoDB log files, set the new value in my.cnf, start the server, then check the error logs if everything went fine. See also <a href="http://mysqldatabaseadministration.blogspot.com/2007/01/increase-innodblogfilesize-proper-way.html">this blog entry</a>

Your absolute InnoDB log size is %s MiB | round(value,1)

rule 'InnoDB buffer pool size' [system\_memory > 0]

innodb\_buffer\_pool\_size / system\_memory \* 100

value < 60

Your InnoDB buffer pool is fairly small.

The InnoDB buffer pool has a profound impact on performance for InnoDB tables. Assign all your remaining memory to this buffer. For database servers that use solely InnoDB as storage engine and have no other services (e.g. a web server) running, you may set this as high as 80% of your available memory. If that is not the case, you need to carefully assess the memory consumption of your other services and non-InnoDB-Tables and set this variable accordingly. If it is set too high, your system will start swapping, which decreases performance significantly. See also <a href="http://www.percona.com/blog/2007/11/03/choosing-innodb\_buffer\_pool\_size/">this article</a>

You are currently using %s% of your memory for the InnoDB buffer pool. This rule fires if you are assigning less than 60%, however this might be perfectly adequate for your system if you don't have much InnoDB tables or other services running on the same machine. | value

#

# other

rule 'MyISAM concurrent inserts' [!PMA\_DRIZZLE]

concurrent\_insert

value === 0 || value === 'NEVER'

Enable {concurrent\_insert} by setting it to 1

Setting {concurrent\_insert} to 1 reduces contention between readers and writers for a given table. See also <a href="http://dev.mysql.com/doc/refman/5.5/en/concurrent-inserts.html">MySQL Documentation</a>

concurrent\_insert is set to 0

# INSERT DELAYED USAGE

#Delayed\_errors 0

#Delayed\_insert\_threads 0

#Delayed\_writes 0

#Not\_flushed\_delayed\_rows