mysqlsla v2 Log Filters

mysqlsla v2 supports *complete* MySQL log filtering. There are two classes of filters: meta-property and SQL statement. Since mysqlsla parses slow, general and binary MySQL logs, the number of native log filters is large.

Furthermore, anticipating the rising importance of **MySQL Proxy** and other proxies, mysqlsla v2 also supports parsing and filtering UDL: user-defined logs of varying formats providing various SQL statement meta-properties.

Therefore, the number of log filters is almost limitless. With such an abundance of information it is necessary to filter out what is extraneous to find easily and quickly what is desired. mysqlsla v2 has been redesigned to accomplish this task.

This document provides first an overview of how mysqlsla v2 handles log filtering: **Meta-Property Filter**, **SQL Statement Filter**, **Setting the Filters**. Then, the list of all intrinsically supported **Meta-Property Names** provided by slow, general and binary MySQL logs is given. Finally, a brief overview is made concerning how mysqlsla v2 handles **User-Defined Log Filtering**.

The majority of the information presented here applies equally to any script that implements **MySQL::Log::ParseFilter**. Features specific to mysqlsla are noted.

mysqlsla v2 Log Filters
Table of Contents

Meta-Property Filter

« Top

Each SQL statement has a multitude of basic meta-properties: execution time, connection ID, number of rows sent in result set, etc. From these basic meta-properties, additional meta-properties can be calculated such as average execution time and maximum number of rows sent.

Each type of MySQL log provides different meta-properties. Therefore, which meta-properties are available to mysqlsla is determined by which type of log is being parsed. The four different types of MySQL logs that mysqlsla can parse (slow, general, binary, udl) consitute the four groups of available meta-properties, with some overlap. The group of meta-properties available in slow logs is sub-divided into three groups: regular slow logs, **microslow (msl) patched slow logs**, and msl patched slow logs with InnoDB values.

The meta-property filter is set with the **meta-filter (-mf)** command line option.

(MySQL::Log::ParseFilter **set_meta_filter()**)

SQL Statement Filter

« Top

The SQL statement filter is a positive/negative filter allowing only (postive) or excluding only (negative) certain types of **SQL statements**: SELECT, UPDATE, SHOW, ALTER, etc. By default, mysqlsla accepts every type of SQL statement, even potentially harmful ones such as DROP, unless otherwise exluded or allowed by the SQL statement filter.

The SQL statement filter is set with the **statement-filter (-sf)** command line option.

(MySQL::Log::ParseFilter **set_statement_filter()**)

Setting the Filters

« Top

The **meta-property filter** takes a comma-separated lists of filter condtions in the form: [meta][op][value]. [meta] is a meta-property name from the long list given below. [op] is either =, > or <. And [value] is the value gainst which the value for [meta] from the log must be true according to [op]. [value] is numeric or text depending on [meta]. For text values [op] can only be =.

All filter condtions must pass for the SQL statement to be saved. Multiple condtions for the same [meta-property name] can be given, even if they are redundant or contraditory. This allows for "range conditions" in some cases. For example, with a slow log and filter condtions "t>10,t<100" only SQL statements with a time value between 10 and 100 (exclusive) will be saved. However, in the example with a general log and filter condtions "cid<2000,cid>5000,cid<6000" *no* SQL statements will be saved because there is no number which passes all three condtions. (At least not in bivalent logic which is currently the only type of logic supported by mysqlsla.)

The **statement filter** takes a comma-separated list of SQL statements types in the form: [+-][TYPE],[TYPE]. The [+-] is only included once in front of the first [TYPE]. It indicate if the filter is positive or negative: a positive filter means save only SQL statements of the given [TYPES]s; a negative filter means remove the given [TYPE]s and save the rest. If no [+-] is given, the default is negative.

For example, to save only USE and SELECT statements: "+USE,SELECT". Or, to remove SET statements: "SET".

Meta-Property Names

« Top

These are all the meta-property names that mysqlsla v2 and **MySQL::Log::ParseFilter** recognize from slow, general, binary and msl log types. The list is divided according to each log type and then grouped according to basic meta-properties. For example, the first basic meta-property for all log types is count which has two meta-property names: `c_sum` and `c_sum_p`.

Filter condtions for non-existent meta-properties are simply ignored. Therefore, accidently using a slow log meta-property with a general log will have no effect.

There are a number of exceptions noted below certain meta-property names.

All Logs

« Top

Count

c_sum: Total number of times SQL statement appears in log
c_sum_p: Percentage that c_sum constitutes of grand total c_sum for all SQL statements in log

Database

db: Database used by SQL statement
» Only for **meta-property filter**
» General logs always provide the database for every SQL statement; binary usually do; slow logs sometimes do; raw logs should but are not required to. See the **databases (-db) (-D)** command line option

Real Execution Time

exec: Real execution time of SQL statement when executed on the MySQL server
» Only for mysqlsla
» Only for **sort**
» Only available when using the **time-each-query (-te)** command line option

exec_sum: Total real execution time of SQL statement (c_sum * exec)
» Only for mysqlsla
» Only for **sort**
» Only available when using the **time-each-query (-te)** command line option

WARNING: A safety **SQL statement filter** of "+SELECT,USE" is automatically set when using **time-each-query** or the **time-all report**. Overriding the safety **SQL statement filter** by explicitly setting another with **statement-filter** can permit *real changes* to databases.

Slow Logs

« Top

Host Name

host: Host name of MySQL connection
» Only for **meta-property filter**
» This value is not always provided for every SQL statement; sometimes it is blank in the slow log

IP Address

ip: IP address of MySQL connection
» Only for **meta-property filter**
» This value is not always provided for every SQL statement; sometimes it is blank in the slow log

Lock Time

l: Time spent acquiring lock
» Only for **meta-property filter**

l_min: Minimum l
l_max: Maximum l
l_avg: Average l
l_sum: Total l
l_sum_p: Percentage that l_sum constitutes of grand total l_sum for all SQL statements in log
l_sum_nthp: Nth percent of all l values
» Only for mysqlsla
» Only for **sort**
» Only available when using the **nth-percent (-nthp)** command line option

Rows Examined

re: Number of rows examined by SQL statement
» Only for **meta-property filter**

re_min: Minimum re
re_max: Maximum re
re_avg: Average re
re_sum: Total re
re_sum_p: Percentage that re_sum constitutes of grand total re_sum for all SQL statements in log

Rows Sent

rs: Number of result set rows sent (returned) to client by SQL statement
» Only for **meta-property filter**

rs_min: Minimum rs
rs_max: Maximum rs
rs_avg: Average rs
rs_sum: Total rs
rs_sum_p: Percentage that rs_sum constitutes of grand total rs_sum for all SQL statements in log

Time

t: Execution time of SQL statement
» Only for **meta-property filter**

t_min: Minimum t
t_max: Maximum t
t_avg: Average t
t_sum: Total t

t_sum_p: Percentage that t_sum constitutes of grand total t for all SQL statements in log
t_sum_nthp: Nth percent of all t values
» Only for mysqlsla
» Only for **sort**
» Only available when using the **nth-percent (-nthp)** command line option

User

user: User of MySQL connection
» Only for **meta-property filter**

Microslow (msl)

« Top

Connection ID

cid: Connection ID of MySQL connection

Yes/No Meta-Properties

These meta-properties are a little different from the others because their **meta-property filter** condtion [value] must be either Yes or No. For example: disktmptable=Yes,filesort=No. Only the Yes occurrences are counted. The _t ending means TRUE (Yes).

diskfilesort_t: SQL statement required a disk-based filesort
diskfilesort_t_p: Percentage that diskfilesort_t

disktmptable_t: SQL statement required a disk-based temporary table
disktmptable_t_p: Percentage that disktmptable_t

filesort_t: SQL statement required a regular filesort
filesort_t_p: Percentage that filesort_t

fulljoin_t: SQL statement required a full JOIN
fulljoin_t_p: Percentage that fulljoin_t

fullscan_t: SQL statement required a full table scan
fullscan_t_p: Percentage that fullscan_t

tmptable_t: SQL statement required a regular (memory-based) temporary table
tmptable_t_p: Percentage that tmptable_t

qchit_t: SQL statement was served from the query cache
qchit_t_p: Percentage that qchit_t

Merge Passes

merge: Number of **merge passes** required to sort result set
» Only for **meta-property filter**

merge_min: Minimum merge
merge_max: Maximum merge
merge_avg: Maximum merge
merge_sum: Total merge
merge_sum_p: Percentage that merge_sum constitutes of grand total merge_sum for all SQL statements in log

Microslow (msl) with InnoDB Values

« Top

InnoDB IO Read Bytes

iorbytes: Number of bytes read by InnoDB for SQL statement
» Only for **meta-property filter**

iorbytes_min: Minimum iorbytes
iorbytes_max: Maximum iorbytes
iorbytes_avg: Average iorbytes
iorbytes_sum: Total iorbytes
iorbytes_sum_p: Percentage that iorbytes_sum constitutes of grand total iorbytes_sum for all SQL statements in log
iorbytes_sum_nthp: Nth percent of all iorbytes values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

InnoDB IO Read Operations

iorops: Number of InnoDB read operations made for SQL statement
» Only for **meta-property filter**

iorops_min: Minimum iorops
iorops_max: Maximum iorops
iorops_avg: Average iorops
iorops_sum: Total iorops
iorops_sum_p: Percentage that iorops_sum constitutes of grand total iorops_sum for all SQL statements in log
iorops_sum_nthp: Nth percent of all iorops values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

InnoDB IO Read Wait

iorwait: Time spent reading IO
» Only for **meta-property filter**

iorwait_min: Minimum iorwait
iorwait_max: Maximum iorwait
iorwait_avg: Average iorwait
iorwait_sum: Total iorwait
iorwait_sum_p: Percentage that iorwait_sum constitutes of grand total iorwait_sum for all SQL statements in log
iorwait_sum_nthp: Nth percent of all iorwait values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

InnoDB Pages Distinct

pages: Number of distinct pages accessed by InnoDB for SQL statement
» Only for **meta-property filter**

pages_min: Minimum pages
pages_max: Maximum pages
pages_avg: Average pages
pages_sum: Total pages
pages_sum_p: Percentage that pages_sum constitutes of grand total pages_sum for all SQL statements in log
pages_sum_nthp: Nth percent of all pages values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

InnoDB Record Lock Wait

reclwait: Time spent waiting for record lock
» Only for **meta-property filter**

reclwait_min: Minimum reclwait
reclwait_max: Maximum reclwait
reclwait_avg: Average reclwait
reclwait_sum: Total reclwait
reclwait_sum_p: Percentage that reclwait_sum constitutes of grand total reclwait_sum for all SQL statements in log
reclwait_sum_nthp: Nth percent of all reclwait values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

InnoDB Queue Wait

qwait: Time thread spent waiting in queue
» Only for **meta-property filter**

qwait_min: Minimum qwait
qwait_max: Maximum qwait
qwait_avg: Average qwait
qwait_sum: Total qwait
qwait_sum_p: Percentage that qwait_sum constitutes of grand total qwait_sum for all SQL statements in log
qwait_sum_nthp: Nth percent of all qwait values
» Only for mysqlsla
» Only for **sort**
» Only available when using the command line options **nth-percent (-nthp)** *and* **save-all-values**

General Logs

« Top

Connection ID

cid: Connection ID of MySQL connection

Host Name

host: Host name of MySQL connection
» Only for **meta-property filter**

User

user: User of MySQL connection
» Only for **meta-property filter**

Binary Logs

« Top

Connection ID (Thread ID)

cid: Connection ID of MySQL connection

Execution Time

ext: Execution time of SQL statement
» Only for **meta-property filter**
» Not to be confused with exec which is the "real" execution time of SQL statement; ext is the "reported" execution time in seconds

ext_min: Minimum ext
ext_max: Maximum ext
ext_avg: Average ext
ext_sum: Total ext
ext_sum_p: Percentage that ext_sum constitutes of grand total ext_sum for all SQL statements in log
ext_sum_nthp: Nth percent of all ext values
» Only for mysqlsla
» Only for **sort**
» Only available when using the **nth-percent (-nthp)** command line option

Error Code

err: Error code (if any) caused by SQL statement

Server ID

sid: Server ID of MySQL server

User-Defined Log Filtering

« Top

User-defined log filtering works nearly identically to basic log filtering. The obvious difference of course is the random nature and availability of meta-properties. But since the user-defined logs are *user*-defined, you should know already what meta-properties are available: those which you defined in the **udl format file**.

Let's imagine a simple UDL that provides a meta-property called t_sending_data: time sending data, the same value available through the **MySQL Query Profiler**. Furthermore, the udl has this meta-property defined as a full aggregate value (nf) and we are running mysqlsla with the the **nth-percent** command line option.

After creating an appropriate **udl format** to define the UDL log format and specifying this UDL format using the **udl-format** option, mysqlsla will parse the UDL and make available the following meta-property names:

t_sending_data: Time sending data to client
» Only for **meta-property filter**

t_sending_data_min: Minimum t_sending_data
t_sending_data_max: Maximum t_sending_data
t_sending_data_avg: Average t_sending_data
t_sending_data_sum: Total t_sending_data
t_sending_data_sum_p: Percentage that t_sending_data_sum constitutes of grand total t_sending_data for all SQL statements in log
t_sending_data_sum_nthp: Nth percent of all t_sending_data values
» Only for mysqlsla
» Only for **sort**
» Only available when using the **nth-percent (-nthp)** command line option

Those meta-properties act like the meta-properties from the basic log types: unless otherwise noted they can be used as filter conditions with the **meta-property filter**; they can be the the value given to **sort**; and they are accessible and formattable in the **standard report**.

Concerning the standard report (which applies only to mysqlsla), you will surely need to either update the default standard report for udl, which is extremely basic, or create your own and set it with the **report-format** command line option. For more information on standard report formats, read **mysqlsla v2 Reports**.

Overall, there are no notable limitations on user-defined logs. In fact, mysqlsla could be rewritten to treat slow and binary logs as user-defined logs.

mysqlsla v2 Log Filters was last updated July 9, 2008 for mysqlsla v2.00.