

## 一：帮助命令

1: proxyhelp

命令功能

查询 DBProxy 支持的管理命令

命令语法

**proxyhelp**

举例

```
mysql> proxyhelp;

+-----+-----+
| Command          | comment                                |
+-----+-----+
| AddBackend        | add a new backend to backend list, example: Addbackend
--backend=127.0.0.1:3306
|
| SetBackendParam    | set the param of a backend, example: SetBackendParam
--backend=127.0.0.1:3306 --rw-weight= --ro-weight= --rise= --fall= --inter --fastdowninter=
|
| DelBackend         | delete a backend from backend list(same as setbkoffline), example:
DelBackend --backend=127.0.0.1:3306
|
| SetBKOffline       | set the status of a backend to pending, example: SetBKOffline
--backend=127.0.0.1:3306
|
| SetBKOnline        | set the status of a backend to up, example: SetBKOnline
--backend=127.0.0.1:3306
|
| ShowBackends       | list the status of backends, example: ShowBackends
|
| AddUser            | add a user of proxy, example: AddUser --username=root --passwd=XXX
--hostip=%
|
| DelUser            | delete a user of proxy username@ip, example: deluser --username=root
[--hostip=%]
|
```

```

| UpdatePwd          | update the password of user, example: updatepwd --username=root
--passwd=XXXX
|
| ShowUsers          | list the status of users, example: ShowUsers
|
| SetConnLimit       | set the connection limit of a user@hostip, example: SetConnLimit
--username=root --port-type=rw/ro [--hostip=X.X.%]
|
| SetPoolConfig      | set the pool configuration of a user, example: SetPoolConfig
--username=root --port-type=rw/ro --max-conn= --min-conn= --max-interval=
|
| ShowPoolConfig     | list the pool configuration of users, example: ShowPoolConfig
|
| ShowPoolStatus     | list the pool status of users, example: ShowPoolStatus
|
| AddSQLFilter       | add a sql filter to proxy, example: AddSQLFilter --username=XXXX
--database=mysql --filter-sql='XXX' --filter-type=single|template
--filter-action=safe|log|warning|block [--filter-disabled=true|false] |
| DelSQLFilter       | delete a sql filter from proxy, example: DelSQLFilter --username=XXXX
--database=mysql --filter-sql='XXXX' --filter-type=single|template
|
| SetFilterSwitch    | [en|dis]able a sql filter in proxy, example: SetFilterSwitch
--username=XXXX --database=mysql --filter-sql='XXXX' --filter-type=single|template
--filter-disabled=true|false
|
| SetFilterAction    | set actionof sql filter in proxy, example: SetFilterAction
--username=XXXX --database=mysql --filter-sql='XXXX' --filter-type=single|template
--filter-action=safe|log|warning|block
|
| ShowSQLFilter      | list sql filter in proxy, ShowSQLFilter
|
| SetMultiplex       | set multiplex function on/off, example: SetMultiplex --flag=on/off
|
| ShowMultiplex      | list multiplex flag, example: ShowMultiplex
|
| ShowProxyProcesslist | list current connections status, example: ShowProxyProcesslist
|
| Addparalimit       | add a sql para, example: AddParaLimit --limit-type= --filter-type=
--username= --database= --filter-sql= --para-limit= --rule-switch=;
|
| ModifyParaLimit    | modify the limitation of sql para execution
|
| ModifyLimitSwitch  | modify the limit switch of sql para execution
|
| DelParaLimit       | delete given para limit
|

```

```

| ShowParaLimit          | display all the para limit
|
| SetParaLimit           | set Para limit function on/off
|
| Showparalimitflag      | list para limit flag
|
| Addduralimit           | add a sql dura limit, example: AddduraLimit --limit-type=
--filter-type= --username= --database= --filter-sql= --posi-limit= --rule-switch=;
|
| ModifyDuraLimit        | modify the limitation of sql para execution
|
| ModifyDuraLimitSwitch  | modify the limit switch of sql para execution
|
| DelDuraLimit           | delete given para limit
|
| ShowDuraLimit          | display all the para limit
|
| SetDuraLimit           | set Para limit function on/off
|
| Showduralimitflag      | list para limit flag
|
| proxyhelp              | list all commands and corresponding comments
|
+-----+
|
+-----+

37 rows in set (0.00 sec)

```

## 二：admin 命令持久化选项

1: **--save-option=all/mem/disk**

选项含义

该选项适用于以下所有修改命令，对于查看命令该选项无意义，但也不会报错

选项参数含义

all 表示 admin 命令将作用于持久化 xml 文件和内存数据结构

mem 表示 admin 命令将仅仅作用于内存数据结构，参数不会持久化

disk 表示 admin 命令将仅仅作用于持久化 xml 文件，当前内存数据结构不会变化，只有重启 DBProxy 进程才会加载参数

持久化文件备份

每次执行修改命令后，都会将上一次的持久化 xml 文件备份，备份文件路径是 xml 同级目录下的 confxmlbak 目录

### 三: Backend 动态管理

#### 1: Backend 动态添加: AddBackend

##### 命令功能

动态添加一个指定的特定类型 backend 至 DBProxy 的 backend 列表

##### 命令语法

AddBackend --backend=ip: port --bktype=rw/ro --save-option=all/mem/disk

##### 参数说明

--backend: 要添加的 backend 的 ip 地址和端口

--bktype: 添加的 backend 的类型, rw:对应写节点; ro:对应读节点

##### 举例

```
mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise       | fall            | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
| 0           | X.X.X.X:3201    | up     | rw   | 1                 | 2         | 2         |
| 3          | 10              | 10     | 4    |                   |           |           |
| 1           | X.X.X.X:3202    | up     | ro   | 2                 | 10        | 3         |
| 4          | 15              | 15     | 1    |                   |           |           |
| 2           | X.X.X.X:3203    | up     | ro   | 4                 | 5         | 4         |
| 5          | 20              | 10     | 8    |                   |           |           |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)
mysql> addbackend --backend=X.X.X.X:3304 --bktype=ro;
Query OK, 1 row affected (0.07 sec)
mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise       | fall            | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
| 0           | X.X.X.X:3201    | up     | rw   | 1                 | 2         | 2         |
| 3          | 10              | 10     | 4    |                   |           |           |
| 1           | X.X.X.X:3202    | up     | ro   | 2                 | 10        | 3         |
| 4          | 15              | 15     | 1    |                   |           |           |
| 2           | X.X.X.X:3203    | up     | ro   | 4                 | 5         | 4         |
| 5          | 20              | 10     | 8    |                   |           |           |
+-----+-----+-----+-----+-----+-----+-----+
```

3		X.X.X.X:3304	unknown	ro	0	2	2	2
3	10	10	2					
-----								
-----								
4 rows in set (0.00 sec)								

## 注意事项

backend 以 ip:port 的形式标示, port>=1025

添加的 backend 不应该与已有的 backend 的列表重复

backend 的类型支持 rw:读写、ro: 只读两种类型

若 backend 列表中已经有活动的 rw 后端时, 不允许在添加新的 rw 类型的 backend

## 2: Backend 参数调整: SetBackendParam

### 命令功能

修改指定 backend 的参数

### 命令语法

```
SetBackendParam --backend=ip:port --rw-weight= --ro-weight= --rise= --fall= --inter=
--fastdowninter=
```

### 参数说明

--backend: 要修改的 backend 的 ip 地址和端口

--rw-weight: 主库权重值

--ro-weight: 只读库权重值

--rise: 连续 rise 次检测成功由 down 状态切换成 up 状态, 如果之前是 up 状态则

不变化

--fall: 连续几次检测成功由 down 状态切换成 up 状态, 如果之前是 up 状态则不

变化

--inter: 脚本的检测间隔时间

--fastdowninter: 如果是 down 状态脚本的检测间隔时间

### 举例

```
mysql> showbackends;
```

-----								
-----								
backend_ndx	address	status	type	connected_clients	rw_weight	ro_weight		
rise	fall	interval	fastdowninter	health				
-----								
-----								
0	X.X.X.X:3401	up	rw	0	1	2	2	
2	10	10	3					
1	X.X.X.X:3402	up	ro	0	2	3	2	
2	10	10	3					

```

| 2          | X.X.X.X:3403 | up    | ro  | 0          | 2          | 2          | 2
| 3          | 10           | 10           | 4          |
+-----+-----+-----+-----+-----+-----+-----+
mysql> setbackendparam --backend=X.X.X.X:3402 --rw-weight=0 --rise=4 --inter=8;
Query OK, 1 row affected (0.11 sec)
mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise | fall | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
| 0          | X.X.X.X:3401 | up    | rw  | 1          | 2          | 2
| 2          | 10           | 10           | 3          |
| 1          | X.X.X.X:3402 | up    | ro  | 0          | 0          | 3          | 4
| 2          | 8           | 10           | 5          |
| 2          | X.X.X.X:3403 | up    | ro  | 2          | 2          | 2
| 3          | 10           | 10           | 4          |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

### 注意事项

--rw-weight 和 --ro-weight 可以设置为 0，其余的值应该为正值，如果为 0 或负值不会报错，但是修改会不生效

### 3: Backend 动态删除: DelBackend（暂时不支持）

#### 命令功能

从DBProxy的backend列表中删除指定的backend(即将backend的状态设置为pending, 达到逻辑删除的目的)

#### 命令语法

**DelBackend --backend=ip: port --save-option=all/mem/disk**

#### 参数说明

--backend: 要被删除的 backend 的 ip+port

#### 举例

```

mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise | fall | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 0          | X.X.X.X:3401 | up      | rw   | 0          | 1          | 2          | 2
| 2  | 10      | 10      | 3      |            |            |            |
| 1          | X.X.X.X:3402 | up      | ro   | 0          | 2          | 3          | 2
| 2  | 10      | 10      | 3      |            |            |            |
| 2          | X.X.X.X:3403 | up      | ro   | 0          | 2          | 2          | 2
| 3  | 10      | 10      | 4      |            |            |            |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
mysql> delbackend --backend=X.X.X.X:3403;
Query OK, 1 row affected (0.02 sec)
mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise  | fall  | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 0          | X.X.X.X:3401 | up      | rw   | 0          | 1          | 2          | 2
| 2  | 10      | 10      | 3      |            |            |            |
| 1          | X.X.X.X:3402 | up      | ro   | 0          | 1          | 2          | 2
| 2  | 10      | 10      | 5      |            |            |            |
| 2          | X.X.X.X:3403 | pending | ro   | 0          | 2          | 2          | 2
| 2  | 10      | 10      | 4      |            |            |            |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

## 注意事项

要删除的 backend 需要在 backend 列表中

## 4: Backend 设置下线: SetBKOffline

### 命令功能

将指定的 backend 设置为不可用即下线

### 命令语法

**SetBKOffline --backend=ip:port**

### 参数说明

--backend: 要设置下线的后端的 ip+port

### 举例

```
mysql> showbackends;
```

```

+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise | fall | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
| 0          | X.X.X.X:3401 | up     | rw   | 0                | 1         | 2         | 2
| 2          | 10           | 10     | 3    |                  |           |           |
| 1          | X.X.X.X:3402 | up     | ro   | 0                | 2         | 3         | 2
| 2          | 10           | 10     | 3    |                  |           |           |
| 2          | X.X.X.X:3403 | up     | ro   | 0                | 2         | 2         | 2
| 3          | 10           | 10     | 4    |                  |           |           |
+-----+-----+-----+-----+-----+-----+-----+
mysql> setbkoffline --backend=X.X.X.X:3403;
Query OK, 1 row affected (0.02 sec)
mysql> showbackends;
+-----+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | status | type | connected_clients | rw_weight | ro_weight |
| rise | fall | interval | fastdowninter | health |
+-----+-----+-----+-----+-----+-----+-----+
| 0          | X.X.X.X:3401 | up     | rw   | 0                | 1         | 2         | 2
| 2          | 10           | 10     | 3    |                  |           |           |
| 1          | X.X.X.X:3402 | up     | ro   | 0                | 1         | 2         | 2
| 2          | 10           | 10     | 5    |                  |           |           |
| 2          | X.X.X.X:3403 | pending| ro   | 0                | 2         | 2         | 2
| 2          | 10           | 10     | 4    |                  |           |           |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

## 注意事项

要设置下线的 backend 需要在 backend 列表中

## 5: Backend 设置上线: SetBKOnline

### 命令功能

设置 backend 的状态为上线，（即将 backend 的状态有 pending 设置为可更新状态）

### 命令语法

**SetBKOnline --backend=ip:port**

### 参数说明



--backend: 需要设置上限的 backend 的 ip+port

### 举例

```
mysql> showbackends;
```

backend_ndx	address	status	type	connected_clients
0	X.X.X.X:3201	up	rw	0
1	X.X.X.X:3202	pending	ro	0
2	X.X.X.X:3203	up	ro	0
3	X.X.X.X:3304	down	ro	0

```
4 rows in set (0.00 sec)
```

  

```
mysql> setbkonline --backend= X.X.X.X:3202;
```

Query OK, 1 row affected (0.00 sec)

  

```
mysql> showbackends;
```

backend_ndx	address	status	type	connected_clients
0	X.X.X.X:3201	up	rw	0
1	X.X.X.X:3202	up	ro	0
2	X.X.X.X:3203	up	ro	0
3	X.X.X.X:3304	down	ro	0

```
4 rows in set (0.00 sec)
```

### 注意事项

要设置为上线的 backend 需要在 backend 列表中

## 6: Backend 状态查询: ShowBackends

### 命令功能

列出 DBProxy 后端代理的所有的 backend 及相关信息包括: backend 的状态, 类型, 上面的 client 连接数

### 命令语法

**ShowBackends**

### 显示各字段含义说明

backend_ndx:	backend 序号
address:	ip+port
status:	backend 的状态 (up/down)
type:	backend 的角色 (rw:主库, ro:备库)

connected\_clients: 该 backend 上面连接的 client 数  
 rw\_weight: 后端对应写请求负载均衡的权重  
 ro\_weight: 后端对应读请求负载均衡的权重  
 rise: 确认后端可用时的连续 up 次数  
 fall: 确认后端不可用时的连续 down 次数  
 interval: 后端检测的时间间隔  
 fastdowninter: 后端状态为 down 时的检测时间间隔  
 health: 后端的健康值 取值在[0 至 rise+fall-1]

#### 举例

```
mysql> showbackends;
```

backend_ndx	address	status	type	connected_clients	rw_weight	ro_weight	rise	fall	interval	fastdowninter	health
0	X.X.X.X:3401	up	ro	0	2	2	2	4	5	5	5
1	X.X.X.X:3402	up	rw	0	4	4	4	2	4	5	5
2	X.X.X.X:3403	up	ro	0	1	1	1	2	4	5	5

3 rows in set (0.00 sec)

## 四：用户管理命令

### 1: proxy 用户动态添加: AddUser

#### 命令功能

动态添加 proxy 用户，包括用户名、密码、允许访问的地址段

#### 命令语法

```
AddUser --username=user --passwd='密码' --hostip=IP 段 --save-option=all/mem/disk
```

#### 参数说明

--username: 添加用户的用户名  
 --passwd: 添加用户的密码  
 --hostip: 用户允许访问的 ip 地址段(注意 IP 段必须是四段，例：10.% 必须写成 10.%.%.%, 又例%必须写成%.%.%.%)

#### 举例

```
mysql> showusers;
```

```

+-----+-----+-----+-----+-----+-----+
| user   | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
| test3  | X.X.X.% | 20(default)   | 0               | 20(default)   | 0               |
| test2  | X.X.X.% | 0              | 0               | 20(default)   | 0               |
| test1  | X.X.X.% | 3              | 0               | 20(default)   | 0               |
| test   | X.X.X.% | 10             | 1               | 50             | 0               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> adduser --username=testuser --passwd='testpwd' --hostip=10.%.%.%;
Query OK, 1 row affected (0.01 sec)

mysql> showusers;
+-----+-----+-----+-----+-----+-----+
--+
| user   | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
--+
| testuser | X.%.%.% | 20(default)   | 0               | 20(default)   | 0               |
| test3    | X.X.X.% | 20(default)   | 0               | 20(default)   | 0               |
| test2    | X.X.X.% | 0              | 0               | 20(default)   | 0               |
| test1    | X.X.X.% | 3              | 0               | 20(default)   | 0               |
| test     | X.X.X.% | 10             | 1               | 50             | 0               |
+-----+-----+-----+-----+-----+-----+
--+
5 rows in set (0.01 sec)

```

## 注意事项

MySQL 中用户名+IP 地址组合成为一个完整的用户，但对 DBProxy 来说一个用户名就是一个用户，IP 地址是用户的一种属性，IP 地址用于控制前端访问

添加用户时需要指定用户名、密码及允许访问的 ip 地址段。支持 X.X.X.%这种格式；

若向一个已存在用户添加 IP 地址段，也需要指定密码，而且必须与原来相同，否则报错；

现在 proxy 中的用户还没有和后端 mysql-server 上面的用户进行同步，即在 proxy 上面添加用户只能保证用户可以通过 proxy 的认证。要想让用户能够利用后端的 backend 的服务，需要在后端也添加相应的用户 username@proxyip

密码要用单引号或双引号括起来，密码中如果有单或双引号，要用反斜杠转移，比如\'和\'。

## 2: proxy 用户动态删除: DelUser

### 命令功能

动态删除 proxy 用户，包括用户名及相应允许访问的地址段

### 命令语法

`DelUser --username=user [--hostip=ip(段)] --save-option=all/mem/disk`

#### 参数说明

`--username:` 删除用户的用户名  
`--hostip:` 删除用户的 ip 段, 若不指定将所有的 ip 段都删除(注意 ip 段必须是四段, 例: 10.% 必须写成 10.%.%, 又例%必须写成%.%.%)

#### 举例

```
mysql> showusers;
+-----+-----+-----+-----+-----+-----+
| user | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
| test | X.X.X.% | 10             | 1              | 50             | 0              |
| test | X.X.%.% | 5              | 0              | 20(default)    | 0              |
| test | X.X.%.% | 20(default)    | 0              | 20(default)    | 0              |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> deluser --username=test --hostip=X.X.X.%;
Query OK, 1 row affected (0.00 sec)

mysql> showusers;
+-----+-----+-----+-----+-----+-----+
| user | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
| test | X.X.%.% | 5              | 0              | 20(default)    | 0              |
| test | X.X.%.% | 20(default)    | 0              | 20(default)    | 0              |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> deluser --username=test;
Query OK, 1 row affected (0.00 sec)

mysql> showusers;
Empty set (0.00 sec)
```

#### 注意事项

删除用户时, `--hostip` 是可选选项, 当指定 `hostip` 时, 只是删除对应于 `@username@hostip` 的用户; 当没有指定 `hostip` 时, 会将 `username` 对应的信息全部删除  
删除未存在用户时, 会报错;  
注意这里也存在用户同步问题, 这里删除用户只是删除用户对 proxy 的认证访问权限, 不会涉及后端 backend 的用户权限。

### 3: proxy 用户密码修改: UpdatePwd

## 命令功能

动态修改某用户登陆的密码

## 命令语法

```
UpdatePwd --username=user --passwd='XXXX'
```

## 参数说明

--username: 更新密码的用户名  
--passwd: 用户新的密码

## 举例

```
mysql> updatepwd --username=testuser --passwd='testppp';  
Query OK, 1 row affected (0.00 sec)
```

## 注意事项

更新密码的账户需要在 proxy 中存在

密码要用单引号或双引号括起来，密码中如果有单或双引号，要用反斜杠转移，比如\'和\'。

## 4: proxy 用户状态查看: ShowUsers

### 命令功能

查询 proxy 允许访问的用户名、相应的地址段及各自的登陆情况

### 命令语法

ShowUsers

### 显示各字段含义说明

user: 用户名  
ip\_range: 用户允许访问的 ip 段  
rw\_login\_limit: 该用户写端口对应的连接限制数  
rw\_login\_count: 该用户写端口实际连接的 client 数目  
ro\_login\_limit: 该用户读端口对应的连接限制数  
ro\_login\_count: 该用户读端口实际连接的 client 数目

## 举例

```
mysql> showusers;  
+-----+-----+-----+-----+-----+-----+  
| user   | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |  
+-----+-----+-----+-----+-----+-----+  
| test3  | X.X.X.% | 20(default)   | 0               | 20(default)   | 0               |  
| test2  | X.X.X.% | 0              | 0               | 20(default)   | 0               |  
| test1  | X.X.X.% | 3              | 0               | 20(default)   | 0               |  
| test   | X.X.X.% | 10             | 1               | 50             | 0               |  
| test   | X.X.%.% | 5              | 0               | 20(default)   | 0               |
```

```
| test | X.X.X.% | 20(default) | 0 | 20(default) | 0 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

## 五：用户限制命令

1：设置用户的连接限制

命令功能

设置用户前端（proxy）的连接数限制

命令语法

```
setconnlimit --username=root --port-type=rw/ro --conn-limit=-1,0,...
[--hostip=ip(段)] --save-option=all/mem/disk
```

参数说明

```
--username:      用户名
--port-type:      读写属性（rw:写端口，ro:读端口）
--conn-limit:     设置实际的连接数（-1：不能连接；0：没有限制；>0:实际连接限制数）
--hostip:         配置连接限制的 ip 段(注意 ip 段必须是四段，例：10.% 必须写成 10.%.%.%，又例%必须写成%.%.%.%)
```

举例

```
mysql> showusers;
+-----+-----+-----+-----+-----+-----+
| user | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
| test3 | X.X.X.% | 20(default) | 0 | 20(default) | 0 |
| test2 | X.X.X.% | 0 | 0 | 20(default) | 0 |
| test1 | X.X.X.% | 3 | 0 | 20(default) | 0 |
| test | X.X.X.% | 10 | 0 | 50 | 0 |
| test | X.X.X.% | 5 | 0 | 20(default) | 0 |
| test | X.X.X.% | 20(default) | 0 | 20(default) | 0 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> setconnlimit --username=test --hostip= X.X.X.% --port-type=ro --conn-limit=400;
Query OK, 1 row affected (0.00 sec)

mysql> showusers;
+-----+-----+-----+-----+-----+-----+
| user | ip_range | rw_login_limit | rw_login_count | ro_login_limit | ro_login_count |
+-----+-----+-----+-----+-----+-----+
| test3 | X.X.X.% | 20(default) | 0 | 20(default) | 0 |
| test2 | X.X.X.% | 0 | 0 | 20(default) | 0 |
```

test1	X.X.X.%	3	0	20(default)	0
test	X.X.X.%	10	0	400	0
test	X.X.X.%	5	0	20(default)	0
test	X.X.X.%	20(default)	0	20(default)	0

5 rows in set (0.00 sec)

## 注意事项

连接限制没有设置时，限制会去系统默认值；

限制数为 0 时，标示没有限制；

限制数为-1 时，标志不允许访问；

设置连接限制时，若 hostip 是可选选项，若 hostip 没有指定会将该用户在所有 ip 段上的连接线指数修改。

## 六：连接池配置命令

### 1: 用户连接池配置

命令功能

设置用户对应于每个后端的连接池的配置信息

命令语法

```
setpoolconfig --username=root --port-type=rw/ro --max-conn= --min-conn=
--max-interval= --save-option=all/mem/disk
```

参数说明

--username: 设置连接池对应的用户名  
--port-type: 设置的连接池的读写属性  
--max-conn: 连接池对应的最大连接数  
--min-conn: 连接池对应的最小连接数  
--max-interval: 连接最大空闲时间

举例

```
mysql> showpoolconfig;
```

user	rw_max_connection	rw_min_connection	rw_idle_interval	ro_max_connection	ro_min_connection	ro_idle_interval
test3	200(default)	50(default)	3600(default)	200(default)	50(default)	3600(default)
test2	200(default)	50(default)	3600(default)	200(default)	50(default)	3600(default)
test1	200(default)	50(default)	3600(default)	200(default)	50(default)	3600(default)

```

| test | 200(default) | 50(default) | 3600(default) | 200(default) |
50(default) | 3600(default) |
+-----+-----+-----+-----+-----+-----+
-----+-----+
4 rows in set (0.00 sec)

mysql> setpoolconfig --username=test1 --port-type=rw --max-conn=100;
Query OK, 1 row affected (0.00 sec)

mysql> setpoolconfig --username=test3 --port-type=ro --max-conn=800;
Query OK, 1 row affected (0.00 sec)

mysql> showpoolconfig;
+-----+-----+-----+-----+-----+-----+
-----+-----+
| user | rw_max_connection | rw_min_connection | rw_idle_interval | ro_max_connection |
ro_min_connection | ro_idle_interval |
+-----+-----+-----+-----+-----+-----+
-----+-----+
| test3 | 200(default) | 50(default) | 3600(default) | 800 | 50 |
| 3600 |
| test2 | 200(default) | 50(default) | 3600(default) | 200(default) |
50(default) | 3600(default) |
| test1 | 100 | 50 | 3600 | 200(default) |
50(default) | 3600(default) |
| test | 200(default) | 50(default) | 3600(default) | 200(default) |
50(default) | 3600(default) |
+-----+-----+-----+-----+-----+-----+
-----+-----+
4 rows in set (0.00 sec)

```

## 注意事项

我们提供的连接池是针对用户区分的,每个用户在每个后端 backend 上面的连接池配置信息是一样的

用户对应的连接池中的连接数目会在 min\_connection 和 max\_connection 之间:

- a. 在用户请求比较频繁,且连接数比较多且不断增加时;后端连接池会不断创建新的连接,但是连接池最大只能到, max\_connection(分 ro 和 rw 两类)
- b. 同时 DBProxy 会检测连接池中的连接是否空闲,会定期将空闲的连接 kill 掉如 c 所述;
- c. 当用户的请求不频繁且使用的连接数比较少时, DBProxy 每隔一个时间周期,检查空闲的连接,每次 kill 掉一定数目的空闲连接数,但是连接池最少会为用户维持 min\_connection 的连接



## 2: 用户连接池配置查询

命令功能

查询对应用户的连接池的配置情况

命令语法

**showpoolconfig**

显示各字段含义说明

<b>user:</b>	用户名
<b>rw_max_connection:</b>	写连接池最大连接数
<b>rw_min_connection:</b>	写连接池最小连接数
<b>rw_idle_interval:</b>	写连接池中连接的最大空闲时间
<b>ro_max_connection:</b>	读连接池最大连接数
<b>ro_min_connection:</b>	读连接池最小连接数
<b>rw_idle_interval:</b>	读连接池中连接的最大空闲时间

举例

```
mysql> showpoolconfig;
+-----+-----+-----+-----+-----+-----+
| user   | rw_max_connection | rw_min_connection | rw_idle_interval | ro_max_connection | ro_min_connection | ro_idle_interval |
+-----+-----+-----+-----+-----+-----+
| test3  | 200(default)     | 50(default)       | 3600(default)   | 800               | 50                | |
| 3600   |                   |                   |                  |                   |                   |
| test2  | 200(default)     | 50(default)       | 3600(default)   | 200(default)      | 50(default)       | 3600(default)   |
| test1  | 100              | 50                | 3600            | 200(default)      | 50(default)       | 3600(default)   |
| test   | 200(default)     | 50(default)       | 3600(default)   | 200(default)      | 50(default)       | 3600(default)   |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

## 3: 用户连接池状态查询

命令功能

查询对应用户的连接池的使用情况

命令语法

**showpoolstatus**

显示各字段含义说明

user:	连接池对应的用户名
backend:	连接池对应的 backend(ip+port)
rw_conn_idle:	写连接池对应的空闲连接数
rw_conn_using:	写连接池对应的在用的连接数
rw_conn_pending:	写连接池对应的在创建的连接数
ro_conn_idle:	读连接池对应的空闲连接数
ro_conn_using:	读连接池对应的在用的连接数
ro_conn_pending:	读连接池对应的在创建的连接数

## 举例

```
mysql> showpoolstatus;
```

user	backend	rw_conn_idle	rw_conn_using	rw_conn_pending	ro_conn_idle	ro_conn_using	ro_conn_pending
test3	X.X.X.X:3201	0	0	0	0	0	0
0							
test3	X.X.X.X:3202	0	0	0	0	0	0
0							
test3	X.X.X.X:3203	0	0	0	0	0	0
0							
test2	X.X.X.X:3201	0	0	0	0	0	0
0							
test2	X.X.X.X:3202	0	0	0	0	0	0
0							
test2	X.X.X.X:3203	0	0	0	0	0	0
0							
test1	X.X.X.X:3201	0	0	0	0	0	0
0							
test1	X.X.X.X:3202	0	0	0	0	0	0
0							
test1	X.X.X.X:3203	0	0	0	0	0	0
0							
test	X.X.X.X:3201	3	0	0	0	0	0
0							
test	X.X.X.X:3202	0	0	0	2	0	0
0							
test	X.X.X.X:3203	0	0	0	2	0	0
0							

```
12 rows in set (0.00 sec)
```

## 七: sql 限制操作命令

### 1: sql 过滤功能说明

过滤规则是基于用户名和数据库的。

sql 过滤规则现在分为两类: 单个语句的 sql 限制和某类的 sql 限制。

有 sql 语句到来时, 先在找是否有单个 sql 的过滤规则与之对应, 有则按该规则对应的动作执行; 没有会找是否有某类 sql 的规律规则与之对应, 有则按照对应的动作执行; 都没有则认为 sql 是安全的, 正常执行。注意单个 sql 的过滤规则优先匹配, 比如你配置 2 条规则, 规则 1: 单个 sql 通过, 规则 2: 这个 sql 对应的模板是阻塞, 最终结果是这条 sql 会通过正常秩序

关于标准化 sql, 里面包括两种: 单条语句过滤规则对应的标准化 sql 和某类规律规则对应的标准化 sql, 具体说明如下:

单条语句的标准化规则:

```
/**
 * @note 此类 sql 的标准化只需要做一下工作:
 * 1. 将关键字变为小写
 * 2. 将 hint, 注释、多余的空格去掉
 * 3. 去掉换行符
 * 4. 将`db`.`table`的``符号去掉;
 * 不需要绑定变量替换
 */
```

某类 sql 语句的标准化规则

```
/**
 * 输入的 SQL 必须进行规范化 标准如下:
 * 1. 替换成绑定变量;
 * 2. 将 in (1,2,3,4,5) 转换成 in (N);
 * 3. 将 hint, 注释、多余的空格去掉
 * 4. 去除回车换行符;
 * 5. 统一为小写;
 * 6. 将`db`.`table`的``符号去掉;
 */
```

### 2: 新增 sql 过滤规则

命令功能

添加一个 sql 过滤规则, 可以添加一个 sql 语句的过滤规则 and 一类 sql 语句的过滤规则。

命令语法

```
addsqlfilter --username=XXXX --filter-sql='sql' --database=db
--filter-type=single --filter-action=block --filter-disabled=true/false
--save-option=all/mem/disk
```

参数说明:

--username: 添加规则对应的用户名  
 --filter-sql: 规则对应的 sql 语句（一个代表语句，proxy 会对 sql 语句进行标准化）  
 --database: 规则对应的数据库。（如果对应的数据库不存在请使用 NULL）  
 --filter-type: 规则的类型，暂时支持 single 和 template 两种类型  
 --filter-action: 规则对应的动作：包括 safe/log/warning/block 四类  
 --filter-disabled: 设置规则是否启用。可取值为 true 或 false。true 表示规则不启用，False 表示启用。  
 注意：该选项是可选项，默认是启用规则。

## 举例

```
mysql> showsqlfilter;
Empty set (0.00 sec)

mysql> addsqlfilter --username=test --filter-sql='select * from t1 limit 1' --database=d1
--filter-type=single --filter-action=block;
Query OK, 1 row affected (0.01 sec)

mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | d1 | select * from t1 limit 1 | block | single | true |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

从另一个客户端登陆看到的情景是：

添加规则之前：

```
mysql> select * from t1 limit 1;
+-----+-----+
| intcol1 | charcol1 |
+-----+-----+
| 1804289383 | mxvttvmC9127qJNm06sGB8R92q2j7vTi iITRDGXM9ZLzkdekbWtmXKwZ2qG111kRw5m9DHOFi lEREk3q7oce803BEJC0w oJsm6uzFAEynLH2xCsw1KQ11T4zg9rdxBL |
+-----+-----+
1 row in set (0.08 sec)
```

添加规则之后：

```
mysql> select * from t1 limit 1;
ERROR 1045 (28000): SQL not allowed: May be it's not safe to run this sentence
```

### 3: 删除 sql 过滤规则

命令功能

删除某条 sql 过滤规则

命令语法

```
delsqlfilter --username=XXXX --filter-sql='sql' --database=db
--filter-type=single|template --save-option=all/mem/disk;
```

参数说明

参考添加 sql 过滤规则的参数说明

--username:	规则对应的用户名
--filter-sql:	规则对应的 sql 语句（一个代表语句，proxy 会对 sql 语句进行标准化）
--database:	规则对应的数据库
--filter-type:	规则的类型，暂时支持 single 和 template 两种类型

举例

```
mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | dl      | select * from t1 limit 1 | block      | single   | true        |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
mysql> delsqlfilter --username=test --filter-sql='select * from t1 limit 1' --database=dl
--filter-type=single;
Query OK, 1 row affected (0.00 sec)
mysql> showsqlfilter;
Empty set (0.00 sec)
```

从另一个客户端登陆看到的情景是:

规则删除之后:

```
mysql> select * from t1 limit 1;
ERROR 1045 (28000): SQL not allowed: May be it's not safe to run this sentence
```

规则删除之前:

```
mysql> select * from t1 limit 1;
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
| intcol1 | charcol1 |
|
+-----+-----+
| 1804289383 |
mxvtvmC9127qJNm06sGB8R92q2j7vTiiITRDGXM9ZLzkdekbWtmXKwZ2qG11lkRw5m9DHOFilEREk3q7oce803BEJC0w
oJsm6uzFAEynLH2xCsw1KQ11T4zg9rdxBL |
+-----+-----+

1 row in set (0.08 sec)
```

#### 4: 修改过滤规则的动作

命令功能

动态修改 sql 过滤规则的动作

命令语法

```
setfilteraction --username=test --database=d1 --filter-sql='select * from t1
limit 1' --filter-type=single --filter-action=warning
--save-option=all/mem/disk;
```

参数说明

--username:	规则对应的用户名
--filter-sql:	规则对应的 sql 语句（一个代表语句，proxy 会对 sql 语句进行标准化）
--database:	规则对应的数据库
--filter-type:	规则的类型，暂时支持 single 和 template 两种类型
--filter-action:	规则对应的动作：包括 safe/log/warning/block 四类

举例

```
mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | d1 | select * from t1 limit 1 | block | single | true |
+-----+-----+-----+-----+-----+-----+

1 row in set (0.00 sec)

mysql> setfilteraction --username=test --database=d1 --filter-sql='select * from t1 limit 1'
--filter-type=single --filter-action=warning;
Query OK, 1 row affected (0.00 sec)

mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | d1 | select * from t1 limit 1 | warning | single | true |
+-----+-----+-----+-----+-----+-----+

1 row in set (0.00 sec)
```

user	database	normalized_sql	rule_action	rule_type	rule_enabled
test	dl	select * from t1 limit 1	warning	single	true

1 row in set (0.00 sec)

## 5: 开停过滤规则

### 命令功能

设置 sql 过滤规则是否启用

### 命令语法

```
setfilterswitch --username=XXXX --database=db --filter-sql='sql'
--filter-type=single|template --filter-disabled=true|false
--save-option=all/mem/disk
```

### 参数说明

--username: 规则对应的用户名  
 --filter-sql: 规则对应的 sql 语句（一个代表语句，proxy 会对 sql 语句进行标准化）  
 --database: 规则对应的数据库  
 --filter-type: 规则的类型，暂时支持 single 和 template 两种类型  
 --filter-disabled: 设置规则是否启用。可取值为 true 或 false。true 表示规则不启用，False 表示启用。该选项是可选项，默认是启用规则。

### 举例

```
mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | dl | select * from t1 limit 1 | block | single | true |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> setfilterswitch --username=test --database=dl --filter-sql='select * from t1 limit 1'
--filter-type=single --filter-disabled=true;
Query OK, 1 row affected (0.00 sec)

mysql> showsqlfilter;
+-----+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+-----+
| test | dl | select * from t1 limit 1 | block | single | false |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

从另一个客户端登陆看到关闭规则前后的情景是：

```
规则关闭之前：

mysql> select * from t1 limit 1;
ERROR 1045 (28000): SQL not allowed: May be it's not safe to run this sentence

规则关闭之后：

mysql> select * from t1 limit 1;
+-----+
| intcoll | charcoll |
+-----+
| 1804289383 | mxvtvmC9127qJNm06sGB8R92q2j7vTiiITRDGXM9ZLzkdekbWtmXKwZ2qG111kRw5m9DHOFi1EREk3q7oce803BEJC0woJsm6uzFAEynLH2xCsw1KQ11T4zg9rdxBL |
+-----+
```

6：查看所有的过滤规则

命令功能

列出所有的 sql 过滤规则

命令语法

**showsqlfilter**

显示各字段含义说明

user;	规则对应的用户名
database:	规则对应的数据库
normalized_sql:	规则对应的标准化 sql
rule_action:	规则对应的动作
rule_type:	规则的类型
rule_enabled:	规则是否启用

举例

```
mysql> showsqlfilter;
+-----+-----+-----+-----+-----+
| user | database | normalized_sql | rule_action | rule_type | rule_enabled |
+-----+-----+-----+-----+-----+
| test | dl | select * from t1 limit 1 | block | single | true |
+-----+-----+-----+-----+-----+
```



```
+-----+
1 row in set (0.00 sec)
```

## 7: SetSqlFilterFlag

### 功能说明

动态设置全局 sql 过滤规则的是否启用

### 命令语法

**SetSqlFilterFlag --flag=on/off**

### 选项说明

**--flag:** 设置并发限制是否启用（on 为启用；off 为关闭不启用）

### 举例说明

```
mysql> showsqlfilterflag;
+-----+
| SqlFilter_flag |
+-----+
| on             |
+-----+

1 row in set (0.00 sec)
mysql> setsqlfilterflag --flag=off;

Query OK, 1 row affected (0.11 sec)
mysql> showsqlfilterflag;
+-----+
| SqlFilter_flag |
+-----+
| off            |
+-----+

1 row in set (0.00 sec)
```

## 8: ShowSqlFilterFlag

### 功能说明

查看 sql 过滤规则的启用标志

### 命令语法

**ShowSqlFilterFlag**

### 举例说明

```
mysql> showsqlfilterflag;
```

```

+-----+
| SqlFilter_flag |
+-----+
| off           |
+-----+

1 row in set (0.00 sec)

```

## 八：连接复用控制命令

### 1：连接复用控制功能说明

对管理员提供连接复用的开关接口，可以在线的启停连接复用，不影响 DBProxy 的正常运行。

### 2：设置连接复用启停

命令功能

设置连接复用的启停状态

命令语法

**SetMultiplexSwitch --flag=on/off**

显示各列含义说明

**flag**:标识是否启用连接复用，可取值为 on 或 off。on 为连接复用开启，off 为连接复用关闭。

举例

```

mysql> SetMultiplexSwitch --flag=off;
Query OK, 1 row affected (0.00 sec)

mysql> ShowMultiplexSwitch;
+-----+
| multiplex switch |
+-----+
| off              |
+-----+

1 row in set (0.01 sec)

mysql> SetMultiplexSwitch --flag=on;
Query OK, 1 row affected (0.00 sec)

mysql> ShowMultiplexSwitch;
+-----+
| multiplex switch|
+-----+

```

```
| on |
+-----+
1 row in set (0.00 sec)
```

### 3: 连接复用状态查看

命令功能

查看 DBProxy 中连接复用是否启用

命令语法

ShowMultiplexSwitch

## 九：查看 sql 当前运行状态命令

### 1: ShowProxyProcesslist

功能说明

ShowProxyProcesslist 语句目的是为了查看当前的 sql 执行的状态，典型的应用场景是查看运行时间较长的 sql 语句

sql 语句的 State 分为两类:Running 和 Sleeping。Running 表示当前的语句正在执行，Sleeping 表示语句已经执行完毕，显示的是上一条执行的语句

命令语法

ShowProxyProcesslist

字段说明

Thread_Id:	对应后端 mysql 的 id, 所以通过这个 id 可以看 mysql 服务的 show processlist 的 state 列和其他列
User:	登录的用户名
Client Host:	前端连接的 ip:port
Backend Host:	后端连接的 mysql 的 ip:port
Database:	数据库名
State:	sql 语句的状态 ,running 表示正在执行。Sleep 表示空闲,如果有 sql, 表示上次执行的 sql
Time:	单位秒, sql 的执行时间 (如果 stat=Sleep, 是上次 sql 实际执行的时间.如果 stat=Runing, 是 sql 运行到现在为止的时间)
Sql:	正在执行的 sql 语句或者上一次执行的 sql

注意事项

连接复用选项是否开启会使 ShowProxyProcesslist 命令结果有所差别

如果连接复用功能关闭, 每一个客户端 query 会长期占用 mysql 的连接, 这样 ShowProxyProcesslist 可以看到上一条 sql 的 Sleep 状态以及执行时间

如果连接复用功能开启, 每个客户端的 query 执行完毕后, 连接会被别的 query 复用, 那么 ShowProxyProcesslist 可能看不到上一条 sql 的 Sleep 状态以及执行时间

无论是否开启连接复用, 都会显示状态为 Running 的 Sql 以及 sql 已经执行的时间

举例

```
mysql> showproxyprocesslist;
```

Thread Id	User	Client Host	Backend Host	Database	State	Time	Sql
1723481	test	X.X.X.X:64498	X.X.X.X:3402	test	Running	24	select sleep(40), 1
1723482	test	X.X.X.X:64499	X.X.X.X:3402	test	Running	24	select sleep(40), 5
1690076	test	X.X.X.X:64501	X.X.X.X:3403	test	Running	24	select sleep(40), 2
1723483	test	X.X.X.X:64502	X.X.X.X:3402	test	Running	24	select sleep(40), 3
1723522	test	X.X.X.X:64504	X.X.X.X:3402	test	Running	24	select sleep(40), 4
1723523	test	X.X.X.X:64510	X.X.X.X:3402	test	Running	24	select sleep(40), 7
1723524	test	X.X.X.X:64512	X.X.X.X:3402	test	Running	24	select sleep(40), 8
1723526	test	X.X.X.X:64513	X.X.X.X:3402	test	Running	24	select sleep(40), 9
1690609	test	X.X.X.X:64514	X.X.X.X:3403	test	Running	24	select sleep(40), 10
1723525	test	X.X.X.X:64516	X.X.X.X:3402	test	Running	24	select sleep(40), 6

```
10 rows in set (0.00 sec)
```

## 十: Sql 直方图统计命令

### 1: 查看 user/db 的标准化 sql 统计直方图

#### 功能说明

ShowQueryResponseTime 语句目的是为了查看相应的 username, database 上的标准化后的 sql 执行时间直方图

可以按照指定的 username 和/或 database 来查看关心的执行时间直方图

#### 命令语法

```
ShowProxyProcesslist [--username=XXXX] [--database=db]
```

#### 参数说明

```
--username:      统计对应的用户名
--database:      统计对应的数据库
```

## 字段说明

Time:	统计对应的时间范围
User:	统计对应的用户名
Db:	统计对应的数据库
Sql:	标准化后的 sql 语句
Count:	sql 语句执行次数
Total:	sql 语句执行总时长

## 注意事项

由于该管理员统计命令会对内存数据结构加锁, 执行时可能会导致 mysql-proxy 的整体性能下降, 建议在执行前先关闭统计功能, 执行后再开启统计功能。当然, 关闭统计功能过程中的 sql 语句统计将不再记录

## 举例

```
mysql> ShowQueryResponseTime;
```

Time	User	Db	Sql	Count	Total
0.001000~0.010000	test1	test	select * from test where a=?	2	0.003513
0.010000~0.100000	test1	test	select * from test where a=?	1	0.013403
0.001000~0.010000	test1	test	select ?	4	0.016922
0.001000~0.010000	test	test2	select @@version_comment limit ?	1	0.005449
0.001000~0.010000	test	test2	select ?	3	0.009984
0.001000~0.010000	test	test1	select @@version_comment limit ?	2	0.015030
0.001000~0.010000	test	test1	select ?	5	0.019605
0.001000~0.010000	test	test	select @@version_comment limit ?	2	0.010099
0.001000~0.010000	test	test	select ?	6	0.027194
0.001000~0.010000	test1	test2	select ?	1	0.003582
0.001000~0.010000	test1	test1	select @@version_comment limit ?	4	0.020319
0.001000~0.010000	test1	test1	select ?	2	0.008974
0.010000~0.100000	test1	test	select @@version_comment limit ?	7	0.351421
0.010000~0.100000	test1	test	select ?	5	0.312646
0.010000~0.100000	test	test2	select @@version_comment limit ?	6	0.169532
0.010000~0.100000	test	test2	select ?	7	0.238122
0.010000~0.100000	test	test1	select @@version_comment limit ?	5	0.189049
0.010000~0.100000	test	test1	select ?	4	0.168559
0.010000~0.100000	test	test	select @@version_comment limit ?	6	0.279458
0.010000~0.100000	test	test	select ?	4	0.144074
0.010000~0.100000	test1	test2	select @@version_comment limit ?	6	0.175733
0.010000~0.100000	test1	test2	select ?	7	0.228964
0.010000~0.100000	test1	test1	select @@version_comment limit ?	2	0.150856
0.010000~0.100000	test1	test1	select ?	4	0.177885
0.100000~1.000000	test	test2	select @@version_comment limit ?	3	0.811905

```

| 0.100000~1.000000 | test1 | test | select @@version_comment limit ? | 2 | 0.494362 |
| 0.100000~1.000000 | test1 | test | select ? | 1 | 0.141776 |
| 0.100000~1.000000 | test | test1 | select @@version_comment limit ? | 3 | 0.613485 |
| 0.100000~1.000000 | test | test1 | select ? | 1 | 0.139505 |
| 0.100000~1.000000 | test | test | select @@version_comment limit ? | 2 | 0.307679 |
| 0.100000~1.000000 | test1 | test2 | select @@version_comment limit ? | 4 | 0.543711 |
| 0.100000~1.000000 | test1 | test1 | select @@version_comment limit ? | 4 | 0.670676 |
+-----+-----+-----+-----+-----+-----+
31 rows in set (0.01 sec)
mysql> ShowQueryResponseTime --username=test1;
+-----+-----+-----+-----+-----+-----+
| Time | User | Db | Sql | Count | Total |
+-----+-----+-----+-----+-----+-----+
| 0.001000~0.010000 | test1 | test | select @@version_comment limit ? | 1 | 0.003513 |
| 0.001000~0.010000 | test1 | test | select ? | 4 | 0.016922 |
| 0.001000~0.010000 | test1 | test2 | select ? | 1 | 0.003582 |
| 0.001000~0.010000 | test1 | test1 | select @@version_comment limit ? | 4 | 0.020319 |
| 0.001000~0.010000 | test1 | test1 | select ? | 2 | 0.008974 |
| 0.010000~0.100000 | test1 | test | select @@version_comment limit ? | 7 | 0.351421 |
| 0.010000~0.100000 | test1 | test | select ? | 5 | 0.312646 |
| 0.010000~0.100000 | test1 | test2 | select @@version_comment limit ? | 6 | 0.175733 |
| 0.010000~0.100000 | test1 | test2 | select ? | 7 | 0.228964 |
| 0.010000~0.100000 | test1 | test1 | select @@version_comment limit ? | 2 | 0.150856 |
| 0.010000~0.100000 | test1 | test1 | select ? | 4 | 0.177885 |
| 0.100000~1.000000 | test1 | test | select @@version_comment limit ? | 2 | 0.494362 |
| 0.100000~1.000000 | test1 | test | select ? | 1 | 0.141776 |
| 0.100000~1.000000 | test1 | test2 | select @@version_comment limit ? | 4 | 0.543711 |
| 0.100000~1.000000 | test1 | test1 | select @@version_comment limit ? | 4 | 0.670676 |
+-----+-----+-----+-----+-----+-----+
15 rows in set (0.01 sec)
mysql>

```

## 2: 查看总体标准化 sql 统计直方图

### 功能说明

ShowTotalResponseTime 语句目的是为了查看整体的标准化后的 sql 执行时间直方图，与 percona 中的 show response time 命令相似

与 ShowQueryResponseTime 不同的是：

a.ShowQueryResponseTime 体现的是不同 user，db 的 sql 执行时间直方图，如果某个时间段如 0.001~0.01 没有相应的记录，那么将不会展示这一时间段

b.ShowQueryResponseTime 在一个时间段如 0.001~0.01 有多于一条的记录，将会全部展示出来

### 命令语法

ShowTotalResponseTime

字段说明

Time:	统计对应的时间范围
Count:	所有该范围内的标准化后的 sql 语句执行次数
Total:	所有该范围内的标准化后的 sql 语句执行总时长

注意事项

由于该管理员统计命令会对内存数据结构加锁, 执行时可能会导致 mysql-proxy 的整体性能下降, 建议在执行前先关闭统计功能, 执行后再开启统计功能。当然, 关闭统计功能过程中的 sql 语句统计将不再记录

举例

```
mysql> ShowTotalResponseTime;
+-----+-----+-----+
| Time          | Count | Total   |
+-----+-----+-----+
| 0.000001      | 0     | 0.000000 |
| 0.000010      | 0     | 0.000000 |
| 0.000100      | 0     | 0.000000 |
| 0.001000      | 0     | 0.000000 |
| 0.010000      | 31    | 0.140671 |
| 0.100000      | 63    | 2.586299 |
| 1.000000      | 20    | 3.723099 |
| 10.000000     | 0     | 0.000000 |
| 100.000000    | 0     | 0.000000 |
| 1000.000000   | 0     | 0.000000 |
| 10000.000000  | 0     | 0.000000 |
| 100000.000000 | 0     | 0.000000 |
| 1000000.000000| 0     | 0.000000 |
| 10000000.000000| 0    | 0.000000 |
| 100000000.000000| 0    | 0.000000 |
+-----+-----+-----+
15 rows in set (0.03 sec)
```

3:清空 sql 统计直方图

功能说明

ClearStatistics 语句目的是为了清空所有当前统计的直方图结构

命令语法

ClearStatistics

注意事项

由于该管理员统计命令会对内存数据结构加锁，执行时可能会导致 mysql-proxy 的整体性能下降，建议在执行前先关闭统计功能，执行后再开启统计功能。当然，关闭统计功能过程中的 sql 语句统计将不再记录

#### 举例

```
mysql> ShowQueryResponseTime --username=test1;
```

Time	User	Db	Sql	Count	Total
0.001000~0.010000	test1	test	select @@version_comment limit ?	1	0.003513
0.001000~0.010000	test1	test	select ?	4	0.016922
0.001000~0.010000	test1	test2	select ?	1	0.003582
0.001000~0.010000	test1	test1	select @@version_comment limit ?	4	0.020319
0.001000~0.010000	test1	test1	select ?	2	0.008974
0.010000~0.100000	test1	test	select @@version_comment limit ?	7	0.351421
0.010000~0.100000	test1	test	select ?	5	0.312646
0.010000~0.100000	test1	test2	select @@version_comment limit ?	6	0.175733
0.010000~0.100000	test1	test2	select ?	7	0.228964
0.010000~0.100000	test1	test1	select @@version_comment limit ?	2	0.150856
0.010000~0.100000	test1	test1	select ?	4	0.177885
0.100000~1.000000	test1	test	select @@version_comment limit ?	2	0.494362
0.100000~1.000000	test1	test	select ?	1	0.141776
0.100000~1.000000	test1	test2	select @@version_comment limit ?	4	0.543711
0.100000~1.000000	test1	test1	select @@version_comment limit ?	4	0.670676

15 rows in set (0.00 sec)

```
mysql> ShowTotalResponseTime;
```

Time	Count	Total
0.000001	0	0.000000
0.000010	0	0.000000
0.000100	0	0.000000
0.001000	0	0.000000
0.010000	31	0.140671
0.100000	63	2.586299
1.000000	20	3.723099
10.000000	0	0.000000
100.000000	0	0.000000
1000.000000	0	0.000000
10000.000000	0	0.000000
100000.000000	0	0.000000
1000000.000000	0	0.000000
10000000.000000	0	0.000000



```
| 100000000.000000 | 0 | 0.000000 |
+-----+-----+
15 rows in set (0.00 sec)
mysql> ClearStatistics;
Query OK, 1 row affected (0.01 sec)
mysql> ShowTotalResponseTime;
+-----+-----+
| Time          | Count | Total    |
+-----+-----+
| 0.000001      | 0     | 0.000000 |
| 0.000010      | 0     | 0.000000 |
| 0.000100      | 0     | 0.000000 |
| 0.001000      | 0     | 0.000000 |
| 0.010000      | 0     | 0.000000 |
| 0.100000      | 0     | 0.000000 |
| 1.000000      | 0     | 0.000000 |
| 10.000000     | 0     | 0.000000 |
| 100.000000    | 0     | 0.000000 |
| 1000.000000   | 0     | 0.000000 |
| 10000.000000  | 0     | 0.000000 |
| 100000.000000 | 0     | 0.000000 |
| 1000000.000000| 0     | 0.000000 |
| 10000000.000000| 0     | 0.000000 |
| 100000000.000000| 0     | 0.000000 |
+-----+-----+
15 rows in set (0.00 sec)
```

#### 4: 设置统计直方图的底数

##### 命令功能

设置统计直方图的底数，目前支持以 2/10 为底

##### 命令语法

**SetStatisticsBase --base=2|10**

##### 参数说明

<b>--base:</b>	统计直方图对应的底数(只支持 2 和 10)
----------------	------------------------

##### 注意事项

由于该管理员统计命令会对内存数据结构加锁，执行时可能会导致 mysql-proxy 的整体性能下降，建议在执行前先关闭统计功能，执行后再开启统计功能。当然，关闭统计功能过程中的 sql 语句统计将不再记录

换底数后所有之前的统计值均清空

##### 举例

```
mysql> ShowTotalResponseTime;
```

Time	Count	Total
0.000001	0	0.000000
0.000010	0	0.000000
0.000100	0	0.000000
0.001000	0	0.000000
0.010000	20	0.100185
0.100000	29	1.410377
1.000000	11	1.546909
10.000000	0	0.000000
100.000000	0	0.000000
1000.000000	0	0.000000
10000.000000	0	0.000000
100000.000000	0	0.000000
1000000.000000	0	0.000000
10000000.000000	0	0.000000
100000000.000000	0	0.000000

15 rows in set (0.00 sec)

```
mysql> SetStatisticsBase --base=2;
```

Query OK, 1 row affected (0.00 sec)

```
mysql> ShowTotalResponseTime;
```

Time	Count	Total
0.000002	0	0.000000
0.000004	0	0.000000
0.000008	0	0.000000
0.000015	0	0.000000
0.000031	0	0.000000
0.000061	0	0.000000
0.000122	0	0.000000
0.000244	0	0.000000
0.000488	0	0.000000
0.000977	0	0.000000
0.001953	0	0.000000
0.003906	0	0.000000
0.007812	0	0.000000
0.015625	0	0.000000
0.031250	0	0.000000

0.062500	0	0.000000
0.125000	0	0.000000
0.250000	0	0.000000
0.500000	0	0.000000
1.000000	0	0.000000
2.000000	0	0.000000
4.000000	0	0.000000
8.000000	0	0.000000
16.000000	0	0.000000
32.000000	0	0.000000
64.000000	0	0.000000
128.000000	0	0.000000
256.000000	0	0.000000
512.000000	0	0.000000
1024.000000	0	0.000000
2048.000000	0	0.000000
4096.000000	0	0.000000
8192.000000	0	0.000000
16384.000000	0	0.000000
32768.000000	0	0.000000
65536.000000	0	0.000000
131072.000000	0	0.000000
262144.000000	0	0.000000
524288.000000	0	0.000000
1048576.000000	0	0.000000
2097152.000000	0	0.000000
4194304.000000	0	0.000000
8388608.000000	0	0.000000
16777216.000000	0	0.000000
33554432.000000	0	0.000000
67108864.000000	0	0.000000
+-----+-----+		
46 rows in set (0.01 sec)		

## 5: 设置 sql 统计直方图开关

### 命令功能

设置 sql 统计直方图启停

### 命令语法

**SetStatisticsFlag --flag=on/off**

### 参数说明

**--flag:** 标识是否启用 sql 统计直方图，可取值为 on 或 off。on 为 sql 统计直方图开启，off 为 sql 统计直方图关闭。

## 举例

```
mysql> SetStatisticsFlag --flag=on;
Query OK, 1 row affected (0.00 sec)

mysql> ShowStatisticsFlag;
+-----+
| statistics flag |
+-----+
| on              |
+-----+
1 row in set (0.01 sec)

mysql> SetStatisticsFlag --flag=off;
Query OK, 1 row affected (0.01 sec)

mysql> ShowStatisticsFlag;
+-----+
| statistics flag |
+-----+
| off             |
+-----+
1 row in set (0.00 sec)
```

## 6: 查看 sql 统计直方图启停状态

### 命令功能

查看 sql 统计直方图功能是否启用

### 命令语法

**ShowStatisticsFlag;**

## 十一：设置 sql 并行限制

### sql 并行限制说明

sql 并行限制的引入是为了避免过度消耗资源的 sql 批量执行，通过限制这种 sql 的并行个数达到资源限制的目的。

sql 并行限制规则是基于 `user_db` 的，包括基于某条语句的限制规则和基于某类语句的限制规则。同时又支持个别规则（针对于某个 `user_db` 的限制规则）和普适规则（针对所有 `user_db` 适用的限制规则）。

用户可以设置某条语句的并行执行条数及对应规则是否开启。具体参考[\[2\]](#)限制某类（个）sql 并行执行的个数

### 1: AddParaLimit

#### 功能说明

AddParaLimit 添加一条指定的限制规则。

## 命令语法

```
AddParaLimit --limit-type=individual/global --filter-type=single/global
--username=user --database=db_name --filter-sql= --para-limit=n
--rule-switch=on/off --save-option=all/mem/disk;
```

## 选项说明

--limit-type: 指定并发限制的类别，可以取值为 individual 和 global。individual: 指在某个指定的 user\_db 上面配置的限定规则；global:指统配的一个限定规则；[必选项]

--filter-type: 指定限制的 sql 粒度，可以取值为 single 和 template。single 指不替换绑定变量，template 指替换掉绑定变量；[必选项]

--username: 指规则对应的用户；[当添加 individual 规则时是必选项]

--database: 指规则对应的数据库名；[当添加 individual 规则时是必选项]

--filter-sql: 规则对应的 sql 语句；[必选项]

--para-limit: 并发限制的条数可取值为-1、0 及正整数。取-1 标示不允许语句的执行；取 0 标示没有限制；取正整数值标示实际的限制数；[必选项]

--rule-switch: 指并发规则的开关。默认为开启；[可选项]

## 示例

```
mysql> AddParaLimit --limit-type=individual --filter-type=single --username=test
--database=test --filter-sql="select * from help" --para-limit=2;

Query OK, 1 row affected (0.02 sec)

mysql> showparalimit;

+-----+
| User | Database | Normalized_sql      | Limit_type | Sql_normalie_type | Para_limit |
Limit_switch |
+-----+
| test | test     | select * from help | Individual | Single             | 2          | On
|
+-----+

1 row in set (0.03 sec)

mysql>
```

## 2: ModifyParaLimit

### 功能说明

更新并行限制数

## 命令语法

```
ModifyParaLimit --limit-type=individual/global --filter-type=single/template --username=  
--database= --filter-sql= --para-limit=n --save-option=all/mem/disk;
```

#### 选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名
--database:	规则对应的数据库
--filter-sql:	规则对应的 sql 语句
--para-limit:	规则限制值

#### 示例

```
mysql> showparalimit;  
  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Para_limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| test | test | select * from help | Individual | Single | 3 | On |  
|  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
  
1 row in set (0.01 sec)  
  
mysql> ModifyParaLimit --limit-type=individual --filter-type=single --username=test  
--database=test --filter-sql="select * from help" --para-limit=100;  
  
Query OK, 1 row affected (0.00 sec)  
  
mysql> showparalimit;  
  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Para_limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| test | test | select * from help | Individual | Single | 100 | On |  
|  
+-----+-----+-----+-----+-----+-----+-----+  
-----+
```

```
1 row in set (0.00 sec)
```

### 3: ModifyLimitSwitch

#### 功能说明

更新并行限制启动开关

#### 命令语法

```
ModifyLimitswitch --limit-type=individual/global  
--filter-type=single/template --username=... --database=...  
--filter-sql="select * from help" --rule-switch=off --save-option=all/mem/disk;
```

#### 选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名
--database:	规则对应的数据库
--filter-sql:	规则对应的 sql 语句
--rule-switch:	规则的启用标志 (on 为规则启用, off 规则不启用)

#### 示例

```
mysql> showparalimit;  
+-----+-----+-----+-----+-----+-----+  
-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Para_limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+  
-----+  
| test | test | select * from help | Individual | Single | 100 | On |  
|  
+-----+-----+-----+-----+-----+-----+  
-----+  
1 row in set (0.00 sec)  
mysql> ModifyLimitswitch --limit-type=individual --filter-type=single --username=test  
--database=test --filter-sql="select * from help" --rule-switch=off;  
Query OK, 1 row affected (0.01 sec)  
  
mysql> showparalimit;  
+-----+-----+-----+-----+-----+-----+  
-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Para_limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+  
-----+
```

test   test	select * from help	Individual	Single	100	Off
+-----+					
-----+					

#### 4: DeleteParaLimit

功能说明

删除某个并行限制规则

命令语法

```
delparalimit --limit-type=individual --filter-type=single --username=test
--database=test --filter-sql="select * from help";
```

选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名(当--limit-type 取 global 时, 不需要加该选项)
--database:	规则对应的数据库(当--limit-type 取 global 时, 不需要加该选项)
--filter-sql:	规则对应的 sql 语句

示例

```
mysql> showparalimit;
+-----+
+-----+
| User | Database | Normalized_sql      | Limit_type | Sql_normalie_type | Para_limit |
Limit_switch |
+-----+
+-----+
| test | test     | select * from help  | Individual | Single             | 100        | Off
|
+-----+
+-----+

1 row in set (0.00 sec)

mysql> delparalimit --limit-type=individual --filter-type=single --username=test
--database=test --filter-sql="select * from help";
Query OK, 1 row affected (0.02 sec)

mysql> showparalimit;
Empty set (0.00 sec)
```

#### 5: ShowParaLimit

功能说明



查询所有的并发限制规则

命令语法

```
showparalimit;
```

字段说明

User:	规则对应的用户名
Database:	规则对应的数据库
Normalized_sql:	规则对应的标准化的 sql
Limit_type:	限制规则类别（包括 Individual 和 global）
Sql_normalie_type:	限制 sql 类别
Para_limit:	并发限制数
Limit_switch:	限制是否启用

举例说明

```
mysql> showparalimit;
```

User	Database	Normalized_sql	Limit_type	Sql_normalie_type	Para_limit
test	test	select * from test	Individual	Single	1

```
1 row in set (0.00 sec)
```

## 6: SetParaLimitFlag

功能说明

动态设置并发限制的是否启用

命令语法

```
SetParaLimitFlag --flag=on/off
```

选项说明

--flag:	设置并发限制是否启用（on 为启用；off 为关闭不启用）
---------	-------------------------------

举例说明

```
mysql> showparalimitflag;
```

ParaLimit_flag
on

```
1 row in set (0.00 sec)
```

```
mysql> SetParaLimitFlag --flag=off;
Query OK, 1 row affected (0.02 sec)

mysql> showparalimitflag;
+-----+
| ParaLimit_flag |
+-----+
| off            |
+-----+
1 row in set (0.01 sec)
```

## 7: ShowParaLimitFlag

功能说明

查看并发限制的启用标志

命令语法

**ShowParaLimitFlag**

举例说明

```
mysql> showparalimitflag;
+-----+
| ParaLimit_flag |
+-----+
| off            |
+-----+
1 row in set (0.01 sec)
```

## 十二：设置 sql 执行时间限制

sql 超时时间限制说明

sql 超时限制的引入是为了避免过度消耗资源的 sql 执行时间过长，通过限制这种 sql 的执行时间达到资源限制的目的。

sql 超时限制规则是基于 user\_db 的，包括基于某条语句的限制规则和基于某类语句的限制规则。同时又支持个别规则（针对于某个 user\_db 的限制规则）和普适规则（针对所有 user\_db 适用的限制规则）。

用户可以设置某条语句的超时时间及对应规则是否开启。具体参考[\[3\]](#)限制某类（个）sql 执行的超时时间]

### 1: AddDuraLimit

功能说明

添加超时时间限制

命令语法

```
AddDuraLimit --limit-type=individual/global --filter-type=single/global
--username=user --database=db_name--filter-sql= --posi-limit=n
--rule-switch=on/off;
```

#### 选项说明

--limit-type: 指定并发限制的类别，可以取值为 individual 和 global。individual: 指在某个指定的 user\_db 上面配置的限定规则；global:指统配的一个限定规则；[必选项]

--filter-type: 指定限制的 sql 粒度，可以取值为 single 和 template。single 指不替换绑定变量，template 指替换掉绑定变量；[必选项]

--username: 指规则对应的用户；[当添加 individual 规则时是必选项]

--database: 指规则对应的数据库名；[当添加 individual 规则时是必选项]

--filter-sql: 规则对应的 sql 语句；[必选项]

--posi-limit: 超时限制的执行时间可取值为正整数。[必选项]单位 us

--rule-switch: 指超时规则的开关。默认为开启；[可选项]

#### 示例

```
mysql> AddduraLimit --limit-type=individual --filter-type=single --username=test
--database=test --filter-sql="select sleep(10)" --posi-limit=2000000 --rule-switch=on;
Query OK, 1 row affected (0.00 sec)

mysql> AddduraLimit --limit-type=global --filter-type=template --filter-sql="select sleep(10)"
--posi-limit=2000000 --rule-switch=on;
Query OK, 1 row affected (0.00 sec)

mysql> showduralimit;
+-----+-----+-----+-----+-----+-----+-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit | Limit_switch |
+-----+-----+-----+-----+-----+-----+-----+
| test | test | select sleep (10) | Individual | Single | 2000000 | On |
| NULL | NULL | select sleep (?) | Global | Template | 2000000 | On |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## 2: DelDuraLimit

#### 功能说明

删除超时时间限制

#### 命令语法

```
delduralimit --limit-type=individual --filter-type=single --username=test
--database=test --filter-sql="select * from help";
```

选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名(当--limit-type 取 global 时，不需要加该选项)
--database:	规则对应的数据库(当--limit-type 取 global 时，不需要加该选项)
--filter-sql:	规则对应的 sql 语句

示例

```
mysql> showduralimit;
+-----+-----+-----+-----+-----+-----+-----+
-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit | Limit_switch |
+-----+-----+-----+-----+-----+-----+-----+
-----+
| test | test | select sleep (100) | Individual | Single | 20000000 | On |
|
| test | test | select sleep (10) | Individual | Single | 20000000 | Off |
|
+-----+-----+-----+-----+-----+-----+-----+
-----+
2 rows in set (0.00 sec)

mysql> DelDuralimit --limit-type=Individual --filter-type=Single --username=test
--database=test --filter-sql='select sleep (100)';
Query OK, 1 row affected (0.00 sec)

mysql> showduralimit;
+-----+-----+-----+-----+-----+-----+-----+
-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit | Limit_switch |
+-----+-----+-----+-----+-----+-----+-----+
-----+
| test | test | select sleep (10) | Individual | Single | 20000000 | Off |
+-----+-----+-----+-----+-----+-----+-----+
-----+
1 row in set (0.00 sec)
```

3: ModifyDuraLimit

功能说明

修改超时时间

#### 命令语法

```
ModifyParaLimit --limit-type=individual/global --filter-type=single/template --username=  
--database= --filter-sql= --para-limit=n;
```

#### 选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名
--database:	规则对应的数据库
--filter-sql:	规则对应的 sql 语句
--para-limit:	规则限制值

#### 示例

```
mysql> showduralimit;  
  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| test | test | select sleep (100) | Individual | Single | 20000000 | On  
|  
| test | test | select sleep (10) | Individual | Single | 20000000 | Off  
|  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> ModifyDuraLimit --limit-type=Individual --filter-type=Single --username=test  
--database=test --filter-sql="select sleep (100)" --posi-limit=1000000;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> showduralimit;  
  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit |  
Limit_switch |  
+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| test | test | select sleep (100) | Individual | Single | 1000000 | On  
|
```

```
| test | test | select sleep (10) | Individual | Single | 20000000 | Off |
|
+-----+
2 rows in set (0.00 sec)
```

#### 4: ModifyDuraLimitSwitch

功能说明

修改超时限制开关

命令语法

```
ModifyLimitswitch --limit-type=individual/global
--filter-type=single/template --username=... --database=...
--filter-sql="select * from help" --rule-switch=off;
```

选项说明

--limit-type:	并行限制类别包括 individual 及 global
--filter-type:	并行限制 sql 类别包括 single 及 template
--username:	规则对应的用户名
--database:	规则对应的数据库
--filter-sql:	规则对应的 sql 语句
--rule-switch:	规则的启用标志 (on 为规则启用, off 规则不启用)

示例

```
mysql> showduralimit;
+-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit |
Limit_switch |
+-----+
| test | test | select sleep (100) | Individual | Single | 1000000 | On |
|
| test | test | select sleep (10) | Individual | Single | 20000000 | Off |
|
+-----+
2 rows in set (0.00 sec)

mysql> ModifyDuraLimitswitch --limit-type=Individual --filter-type=Single --username=test
--database=test --filter-sql="select sleep (100)" --rule-switch=off;
Query OK, 1 row affected (0.00 sec)

mysql> showduralimit;
```

```

+-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit | Limit_switch |
+-----+
| test | test | select sleep (100) | Individual | Single | 1000000 | Off |
| test | test | select sleep (10) | Individual | Single | 20000000 | Off |
+-----+
2 rows in set (0.00 sec)

```

### 5: ShowDuraLimit

功能说明

查询超时限制规则

字段说明

User:	规则对应的用户名
Database:	规则对应的数据库
Normalized_sql:	规则对应的标准化的 sql
Limit_type:	限制规则类别（包括 Individual 和 global）
Sql_normalie_type :	限制 sql 类别
Limit:	超时限制时间（单位 us）
Limit_switch:	限制是否启用

举例说明

```

mysql> showduralimit;
+-----+
| User | Database | Normalized_sql | Limit_type | Sql_normalie_type | Limit | Limit_switch |
+-----+
| test | test | select sleep (10) | Individual | Single | 2000000 | On |
| NULL | NULL | select sleep (?) | Global | Template | 2000000 | On |
+-----+
2 rows in set (0.00 sec)

```

### 6: SetDuraLimitFlag

### 功能说明

设置超时执行开关（on/off）

### 命令语法

**SetDuraLimitFlag --flag=on/off**

### 选项说明

**--flag:** 设置并发限制是否启用（on 为启用；off 为关闭不启用）

### 举例说明

```
mysql> setduralimitflag --flag=off;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> showduralimitflag;
```

```
+-----+
| DuraLimit_flag |
+-----+
| off            |
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> setduralimitflag --flag=on;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> showduralimitflag;
```

```
+-----+
| DuraLimit_flag |
+-----+
| on             |
+-----+
```

```
1 row in set (0.00 sec)
```

## 7: ShowDuraLimitFlag

### 功能说明

显示超时执行限制开关

### 命令语法

**ShowDuraLimitFlag**

### 举例说明

```
mysql> showduralimitflag;
```

```
+-----+
```



ParaLimit_flag
off
1 row in set (0.01 sec)

### 十三：连接状态统计信息

#### 1: ShowConnectionState

##### 功能

显示所有或指定连接的状态统计信息

##### 语法

```
ShowConnectionState [--connection_id=?] [--connectionstatefull=true|false]
```

##### 输入

connection\_id 可选。连接标识。默认代表所有连接

connectionstatefull 可选。是否显示全零的状态。默认不显示

##### 输出

connection\_id 连接标识

state 状态名称

cpu\_count CPU 占用次数

cpu\_time CPU 占用时长（微秒）

iowait\_count IO 等待次数

iowait\_time IO 等待时长（微秒）

##### 举例

```
mysql> ShowConnectionState;
```

connection_id	state	cpu_count	cpu_time	iowait_count	iowait_time
6	CONNECTION_STATE_NEW_CREATED	1	3	0	
0					
6	CONNECTION_STATE_ACCEPT	1	28	0	
0					
6	CONNECTION_STATE_INIT	1	14	0	
0					
6	CONNECTION_STATE_SEND_HANDSHAKE	1	20	0	
0					
6	CONNECTION_STATE_READ_AUTH	2	112	1	
1920					

6	CONNECTION_STATE_SEND_AUTH_RESULT	1	17	0
0				
6	CONNECTION_STATE_READ_QUERY	3	90	1
96				
6	CONNECTION_STATE_PROCESS_READ_QUERY	1	117	0
0				
6	CONNECTION_STATE_GET_SERVER_LIST	1	21	0
0				
6	CONNECTION_STATE_GET_SERVER_CONNECTION_LIST	1	74	0
0				
6	CONNECTION_STATE_SEND_QUERY	1	35	0
0				
6	CONNECTION_STATE_READ_QUERY_RESULT	2	71	1
130				
6	CONNECTION_STATE_SEND_QUERY_RESULT	1	84	0
0				
7	CONNECTION_STATE_NEW_CREATED	1	5	0
0				
7	CONNECTION_STATE_INIT	1	13	0
0				
7	CONNECTION_STATE_SEND_HANDSHAKE	1	21	0
0				
7	CONNECTION_STATE_READ_AUTH	2	35	1
1978				
7	CONNECTION_STATE_SEND_AUTH_RESULT	1	12	0
0				
7	CONNECTION_STATE_READ_QUERY	3	228	2
3611422				
7	CONNECTION_STATE_SEND_QUERY_RESULT	1	16	0
0				
+-----+-----+-----+-----+				
-----+-----+				
20 rows in set (0.00 sec)				

2: FlushConnectionState

功能

清理所有或指定连接的状态统计信息

语法

```
FlushConnectionState [--connection_id=?]
```

输入

```
connection_id 可选。连接标识。默认代表所有连接
```

## 输出

无

## 举例

```
mysql> FlushConnectionState --connection_id=7;
Query OK, 1 row affected (0.00 sec)
```

## 3: ShowThreadConnectionState

### 功能

显示所有或指定线程的连接状态统计信息

### 语法

```
ShowThreadConnectionState [--thread_name=?] [--connectionstatefull=true|false]
```

### 输入

thread\_name 可选。线程名称。默认代表所有线程

connectionstatefull 可选。是否显示全零的状态。默认不显示

### 输出

thread\_name 线程名称  
state 状态名称  
cpu\_count CPU 占用次数  
cpu\_time CPU 占用时长（微秒）  
iowait\_count IO 等待次数  
iowait\_time IO 等待时长（微秒）

## 举例

```
mysql> ShowThreadConnectionState;
+-----+-----+-----+-----+-----+
| thread_name | state | cpu_count | cpu_time | iowait_count | iowait_time |
+-----+-----+-----+-----+-----+
| main | CONNECTION_STATE_NEW_CREATED | 1 | 11 | 0 |
| 0 |
| main | CONNECTION_STATE_ACCEPT | 1 | 37 | 0 |
| 0 |
| main | CONNECTION_STATE_INIT | 1 | 17 | 0 |
| 0 |
| main | CONNECTION_STATE_SEND_HANDSHAKE | 1 | 22 | 0 |
| 0 |
| main | CONNECTION_STATE_READ_AUTH | 2 | 122 | 1 |
| 1987 |
```



## 输出

无

## 举例

```
mysql> FlushThreadConnectionState --thread_name=admin_2;  
Query OK, 1 row affected (0.00 sec)
```

## 5: ShowGlobalConnectionState

### 功能

显示全局的连接状态统计信息

### 语法

```
ShowGlobalConnectionState [--connectionstatefull=true|false]
```

### 输入

connectionstatefull 可选。是否显示全零的状态。默认不显示

## 输出

state 状态名称  
cpu\_count CPU 占用次数  
cpu\_time CPU 占用时长（微秒）  
iowait\_count IO 等待次数  
iowait\_time IO 等待时长（微秒）

## 举例

```
mysql> ShowGlobalConnectionState;  
+-----+-----+-----+-----+-----+  
| state | cpu_count | cpu_time | iowait_count | iowait_time |  
+-----+-----+-----+-----+-----+  
| CONNECTION_STATE_NEW_CREATED | 3 | 32 | 0 | 0 |  
| CONNECTION_STATE_ACCEPT | 1 | 37 | 0 | 0 |  
| CONNECTION_STATE_INIT | 3 | 78 | 0 | 0 |  
| CONNECTION_STATE_SEND_HANDSHAKE | 3 | 72 | 0 | 0 |  
| CONNECTION_STATE_READ_AUTH | 6 | 245 | 3 | 5918 |  
| CONNECTION_STATE_SEND_AUTH_RESULT | 3 | 46 | 0 | 0 |
```

256473408	CONNECTION_STATE_READ_QUERY	54	5055	28	
	CONNECTION_STATE_PROCESS_READ_QUERY	3	216	0	0
	CONNECTION_STATE_GET_SERVER_LIST	2	31	0	0
	CONNECTION_STATE_GET_SERVER_CONNECTION_LIST	2	153	0	0
	CONNECTION_STATE_SEND_QUERY	5	219	0	0
	CONNECTION_STATE_READ_QUERY_RESULT	10	631	5	73107
	CONNECTION_STATE_SEND_QUERY_RESULT	25	813	0	0
-----+					
13 rows in set (0.00 sec)					

## 6: FlushGlobalConnectionState

### 功能

清理全局的连接状态统计信息

### 语法

```
FlushGlobalConnectionState
```

### 输入

无

### 输出

无

### 举例

```
mysql> FlushGlobalConnectionState;
Query OK, 1 row affected (0.00 sec)
```

## 十四：负载均衡算法

### 1: ShowLBAlgo

#### 功能

显示当前使用的负载均衡算法

#### 语法

```
ShowLBAlgo [--port-type=rw|ro]
```

#### 输入

port-type 类型。rw 或 ro。默认全部

## 输出

port\_type 类型。rw 或 ro

lbalgo 负载均衡算法。wrr 代表加权轮询，lc 代表最小连接

## 举例

```
mysql> ShowLBAlgo;
+-----+-----+
| port_type | lbalgo |
+-----+-----+
| rw        | wrr    |
| ro        | wrr    |
+-----+-----+
2 rows in set (0.00 sec)
```

## 2: SetLBAlgo

### 功能

置负载均衡算法

### 语法

SetLBAlgo --lbalgo=wrr|lc [--port-type=rw|ro]

## 输入

lbalgo 负载均衡算法。wrr 代表加权轮询，lc 代表最小连接

port-type 类型。rw 或 ro。默认全部

## 输出

无

## 举例

```
mysql> SetLBAlgo --lbalgo=lc;
Query OK, 1 row affected (0.00 sec)
```

## 十五：日志级别

### 1: SetLogLevel

#### 功能

配置日志级别

### 语法

SetLogLevel --loglevel=debug|info|message|warning|critical|error

## 输入

loglevel 日志的级别。

## 输出

无

## 举例

```
mysql> SetLogLevel --loglevel=debug;  
Query OK, 1 row affected (0.00 sec)
```

## 2: ShowLogLevel

### 功能

显示日志级别

### 语法

showloglevel

## 输入

无

## 输出

loglevel 日志级别

## 举例

```
mysql> ShowLogLevel;  
+-----+  
| loglevel |  
+-----+  
| warning  |  
+-----+  
1 row in set (0.00 sec)
```

## 十六：慢查询管理命令

### 1: SetSlowLogConf

#### 功能

修改慢查询配置

#### 语法

SetSlowLogConf --slowlogswitch=on|off --slowlogtime=秒 --slowlogfile=日志名

## 输入

slowlogswitch 开关  
slowlogtime 大于等于此时间(秒)的记录到日志  
slowlogfile 慢查询日志名。可以是相对路径

## 输出



无

### 举例

```
mysql> SetSlowLogConf --slowlogswitch=on;  
Query OK, 1 row affected (0.00 sec)
```

## 2: ShowSlowLogConf

### 功能

显示慢查询配置

### 语法

ShowSlowLogConf

### 输入

无

### 输出

enabled 开关  
execute\_time 大于等于此时间(秒)的记录到日志  
file 慢查询日志名。可以是相对路径

### 举例

```
mysql> ShowSlowLogConf;  
+-----+-----+-----+  
| enabled | execute_time | file |  
+-----+-----+-----+  
| on      | 2.000000    | /opt/sohu/DBProxy/var/log/slow.log |  
+-----+-----+-----+  
1 row affected (0.00 sec)
```

## 十七：流量统计信息管理命令

### 1: sql 累计执行条数的统计及控制

#### 1.1: showsqlaccnum 查询用户的累计执行条数

### 功能说明

列出对应用户的 sql 累计执行条数。

### 命令语法

ShowSqlAccNum [--username=test]

### 选项说明

--username: 指定要查询的用户名  
--isbanned: 指定查询的类型包括被封禁的、未被封禁的及所有

示例

```
mysql> showsqlaccnum;
+-----+-----+-----+-----+-----+-----+-----+
| user | total_sql_num | rw_sql_num | ro_sql_num | dropped_sql_num | is_banned | banned_time |
+-----+-----+-----+-----+-----+-----+-----+
| test | 21            | 10         | 11         | 4              | Off       | 2014-04-09 18:08:15 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

### 1.2: setusersqlaccswitch 封禁或解禁某个用户查询请求

功能说明

将某个用户的查询请求封禁或者解禁。

命令语法

```
SetUserSqlAccSwitch --username=test --is-banned=on|off
```

选项说明

--username: 指定要操作的对象用户名  
--is-banned: 指定动作是解禁还是封禁

示例

```
mysql> setusersqlaccswitch --username=test --is-banned=on;
Query OK, 1 row affected (0.00 sec)
```

### 1.3: showsqlaccswitch 查看 sql 累计统计功能开启状态

功能说明

列出对应用户的 sql 累计执行条数。

命令语法

```
ShowSqlAccSwitch
```

示例

```
mysql> showsqlaccswitch;
+-----+
| sqlaccswitch |
+-----+
| On           |
+-----+
```

```
1 row in set (0.00 sec)
```

#### 1.4: setsqlaccswitch 开启或关闭 sql 累计统计功能

##### 功能说明

列出对应用户的 sql 累计执行条数

##### 命令语法

```
SetSqlAccSwitch --flag=on|off
```

##### 选项说明

--flag: 指定查询的类型包括被封禁的、未被封禁的及所有

##### 示例

```
mysql> setsqlaccswitch --flag=off;  
Query OK, 1 row affected (0.00 sec)
```

#### 2: sql 累计流入流量的统计和控制

##### 2.1: showthroughoutacc 查询用户的累计输入输出流量

##### 功能说明

列出对应用户的 sql 累计输入输出流量

##### 命令语法

```
ShowSqlInBytes --username=<username> --is-banned=on|off
```

##### 选项说明

--username: 指定要查询的用户名

--isbanned: 指定查询的类型包括被封禁的、未被封禁的及所有

##### 示例

```
mysql> showthroughoutacc;  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| user | inbytes | is_in_banned | in_banned_time | outbytes | is_out_banned | out_banned_time |  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
| test | 488 | Off | 2014-04-09 18:16:01 | 2033 | Off | 2014-04-09  
18:15:57 |  
+-----+-----+-----+-----+-----+-----+-----+  
-----+  
1 row in set (0.00 sec)
```

##### 2.2: setinbytesbanned

## 功能

修改用户流入数的限制开关

## 输入:

--username=<username> 指定用户名  
--is-banned=on|off 开关。on: 禁止, off: 不禁止

## 举例

```
mysql> setinbytesbanned --username=test --is-banned=on;  
Query OK, 1 row affected (0.01 sec)
```

## 2.3: setoutbytesbanned

### 功能

修改用户流出数的限制开关

## 输入:

--username=<username> 指定用户名  
--is-banned=on|off 开关。on: 禁止, off: 不禁止

## 举例

```
mysql> setoubytesbanned --username=test --is-banned=on;  
Query OK, 1 row affected (0.01 sec)
```

## 2.4: showthroughoutswitch

### 功能

显示是否统计用户流入流出数

## 举例

```
mysql> showthroughoutswitch;  
+-----+-----+  
| in_through_switch | out_through_switch |  
+-----+-----+  
| On                | On                |  
+-----+-----+  
1 row in set (0.00 sec)
```

## 2.5: setinbytesaccswitch

### 功能

修改是否统计用户流入数

## 输入

--flag=on|off 开关

示例

```
mysql> setinbytesaccswitch --flag=off;  
Query OK, 1 row affected (0.00 sec)
```

## 2.6: setoutbytesaccswitch

功能

修改是否统计用户流出数

输入

--flag=on|off 开关

示例:

```
mysql> setoutbytesaccswitch --flag=off;  
Query OK, 1 row affected (0.00 sec)
```

## 2.7: showsqlaccswitch :

功能说明

列出对应用户的 sql 累计执行条数。

命令语法

ShowSqlAccSwitch

示例

```
mysql> showsqlaccswitch;  
+-----+  
| sqlaccswitch |  
+-----+  
| On           |  
+-----+  
1 row in set (0.00 sec)
```

## 十八：限制类操作

### 1: 限制黑名单语句

限制 set password 和 drop database 语句

#### 1.1: showblacklistflag 查询功能是否开启

功能说明

查询功能是否开启

命令语法

showblacklistflag

示例

```
mysql> showblacklistflag;
```

```
+-----+
| blacklist switch |
+-----+
| on              |
+-----+
1 row in set (0.00 sec)
```

## 1.2: setblacklistflag 开启或关闭功能

功能说明

开启或关闭功能

命令语法

```
setblacklistflag --flag=on|off
```

示例

## 2: 用户 db 空间超过限额而被限制 dml 操作

主要实现空间限制的开关管理及某个用户 dml 操作的开启关闭。

### 2.1: showsqldmlswitch

功能

查询封禁 dml 的功能是否开启。

命令语法

```
showsqldmlswitch
```

示例

```
mysql> showsqldmlswitch;
```

```
+-----+
| sqldmlswitch |
+-----+
| On           |
+-----+
1 row in set (0.00 sec)
```

### 2.2: setsqldmlswitch

功能

开启或开闭 dml 封禁功能

输入

```
setsqldmlswitch --flag=on|off
```

### 2.3: showsqldmlkind

功能:

查看 dml 封禁中包含的操作列表

命令语法

```
showsqldmlkind
```

示例

```
mysql> showsqldmlkind ;
```

	H_ALTER	H_CREATE	H_DELETE	H_DROP	H_INSERT	H_REPLACE	H_RENAME	H_TRUNCATE	H_UPDATE
	Off	On	On	Off	On	On	On	Off	On

```
1 row in set (0.00 sec)
```

## 2.4: showusersqldml 查询用户的是否因 db 文件超限而被封禁

功能说明

列出对应用户的 dml 封禁状态和状态更新的时间。

命令语法

```
showusersqldml [--username=test] [--is-banned=on|off]
```

选项说明

--username: 指定要查询的用户名

--is-banned: 指定查询的类型包括被封禁的、未被封禁的及所有

示例

```
mysql> showusersqldml;
```

user	is_banned	banned_time
test	Off	2014-05-15 14:34:34

```
1 row in set (0.01 sec)
```

## 2.5: setusersqldmlswitch 封禁或解封某个用户的 dml 操作权限

功能

## 封禁或解封某个用户的 dml 操作权限

输入

```
--username 要封禁的用户名
--is-banned 是否要封禁[on|off]
```

### 示例

```
mysql> setusersqldmlswitch --username=test --is-banned=off;
Query OK, 1 row affected (0.00 sec)
```

## 十九: backend 错误统计信息查看

## 1: ShowBKErrorStatic

### 功能说明

查看后端 backend 的错误统计信息。

## 命令语法

```
ShowBKErrorStatic ----backend=ip:port --username=....
```

### 选项说明

```
--backend: 要查询错误统计信息的 backend 的 ip: port
--user:    需要查询错误信息的 user
```

### 字段说明

user:	错误信息对应的用户名
backend:	错误信息对应的后端 ip:port
bk_pool_conn_full:	后端连接池满
bk_pool_conn_not_enough:	后端连接池空闲连接不足
bk_conn_interrupted:	连接被中断（主要是 server 端超时将连接 kill）
bk_backend_down:	后端 down 导致的连接中断错误
bk_trans_vacant:	事务空闲时间太长导致的错误
bk_prep_vacant:	prepare 空闲时间太长导致的错误
bk_context_restore_error:	上下文恢复错误
bk_too_many_concurrence:	sql 并发超限错误
bk_too_long_exec:	sql 执行时间超过限制错误
bk_auth_info_error:	后端认证用户名密码错误
bk_error_other:	其他错误

### 示例

```
mysql> ShowBKErrorState;
```

user	backend	bk_pool_conn_full	bk_pool_conn_not_enough
bk conn interrupted	bk backend down	bk trans vacant	bk prep vacant



```

bk_context_restore_error | bk_too_many_concurrence | bk_too_long_exec | bk_auth_info_error |
bk_error_other |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| test | X.X.X.X:3401 | 0 | 8 | 0 | 0 |
| 0 | 0 | 0 | 0 | 9 |
| 0 | 0 | | | |
| test | X.X.X.X:3402 | 0 | 10 | 0 | 0 |
| 0 | 0 | 0 | 0 | 11 |
| 0 | 0 | | | |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

## 2: ResetBKErrorStatic

### 功能说明

将 backend 的错误统计信息清零，可以具体到只清空某个 backend 的统计信息

### 命令语法

```
ResetBKErrorStatic --backend=ip:port
```

### 选项说明

**--backend:** 要清零错误统计信息的 backend 的 ip: port[可选项，若没指定则会清空所有 backend 上面的统计数据]

### 示例

```

mysql> ShowBKErrorState;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| user | backend | bk_pool_conn_full | bk_pool_conn_not_enough |
bk_conn_interrupted | bk_backend_down | bk_trans_vacant | bk_prep_vacant |
bk_context_restore_error | bk_too_many_concurrence | bk_too_long_exec | bk_auth_info_error |
bk_error_other |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| test | X.X.X.X:3401 | 0 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 7 |
| 0 | 0 | | | |

```

```

| test | X.X.X.X:3402 | 0 | 2 | 0 | 0
| 0 | 0 | 0 | 0 | 3
| 0 | 0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> ResetBKErrorState --backend=X.X.X.X:3402;
Query OK, 1 row affected (0.00 sec)

mysql> ShowBKErrorState;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| user | backend | bk_pool_conn_full | bk_pool_conn_not_enough |
bk_conn_interrupted | bk_backend_down | bk_trans_vacant | bk_prep_vacant |
bk_context_restore_error | bk_too_many_concurrency | bk_too_long_exec | bk_auth_info_error |
bk_error_other |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| test | X.X.X.X:3401 | 0 | 4 | 0 | 0
| 0 | 0 | 0 | 0 | 7
| 0 | 0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

### 3: SetBKErrorStaticFlag

功能说明

错误统计信息开关管理

命令语法

```
SetBKErrorStaticFlag --flag=off/on
```

选项说明

```
--flag: 开启标志
```

示例

```

mysql> ShowBKErrorStateFlag;
+-----+
| BKErrorStatic_flag |

```

```
+-----+
| off |
+-----+
1 row in set (0.01 sec)

mysql> SetBKErrStaticFlag --flag=on;
Query OK, 1 row affected (0.00 sec)

mysql> ShowBKErrStaticFlag;
+-----+
| BKErrStatic_flag |
+-----+
| on |
+-----+
1 row in set (0.00 sec)
```

#### 4: ShowBKErrStaticFlag

功能说明

查询错误统计信息开关参数

命令语法

ShowBKErrStaticFlag

## 二十：查询结果累计流出流量的统计控制

安全需要，DBProxy 的读写端口，不推荐绑定 0.0.0.0:port；并且现在线上的实例，应用端读写是通过读写来 ip 区分，读写端口是相同的，为了尽量少的影响应用，DBProxy 的读写端口需要相同，这样就不能同时监听 0.0.0.0。但是，现在的大集群的方案是通过一对读写虚 ip 来标示不同的用户的，向已有的实例中添加用户，必然需要添加一对读写虚 ip，但是 DBProxy 没有监听 0.0.0.0, 因而需要在 DBProxy 中添加监听的 ip: port。不想每次都要重启 DBProxy 来时先该功能，因而需要有动态添加监听 ip 和 port 的功能。

#### 1: 动态添加监听地址: AddListenAddr

功能说明

动态添加 DBProxy 的监听地址，分读写地址两类

命令语法

AddListenAddr

选项说明

--backend:	指定要增加的监听地址
--bktype:	指定要增加的监听地址的类别

示例

```
mysql> showlistenaddr;
```

```

+-----+-----+
| port_type | listenAddr_list |
+-----+-----+
| rw        | 127.0.0.1:3456   |
| ro        | 127.0.0.1:3455   |
+-----+-----+
2 rows in set (0.00 sec)

mysql> addlistenaddr --backend=127.0.0.1:3454 --bktype=ro;
Query OK, 1 row affected (0.00 sec)

mysql> showlistenaddr;
+-----+-----+
| port_type | listenAddr_list          |
+-----+-----+
| rw        | 127.0.0.1:3456           |
| ro        | 127.0.0.1:3455, 127.0.0.1:3454 |
+-----+-----+
2 rows in set (0.00 sec)

```

## 2: 动态删除监听地址: DelListenAddr

### 功能说明

动态删除 DBProxy 的监听地址，注意现在实现可能只是将监听的 con 索引删除，然后将监听 socket 的注册的时间 del，然后将 socket close 掉。注意 close 的同时会不会有用户请求事件过来，通过添加标志，让连接自己退出，毕竟是跨线程的，设置标志比较安全。

### 命令语法

**DelListenAddr**

### 选项说明

--backend: 指定要删除的监听地址  
 --bktype: 指定要删除的监听地址的类别

### 示例

```

mysql> showlistenaddr;
+-----+-----+
| port_type | listenAddr_list          |
+-----+-----+
| rw        | 127.0.0.1:3456           |
| ro        | 127.0.0.1:3455, 127.0.0.1:3453, 127.0.0.1:3454 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> dellistenaddr --backend=127.0.0.1:3453 --bktype=ro;

```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> showlistenaddr;
```

port_type	listenAddr_list
rw	127.0.0.1:3456
ro	127.0.0.1:3455, 127.0.0.1:3454

```
2 rows in set (0.00 sec)
```

### 注意事项

执行删除后并没有立即释放监听地址，必须有用户请求触发后，DBProxy 才能释放该地址。

如可以通过 telnet, 触发监听端口的释放, 这样后面可以继续绑定使用对应的 ip:port。

### 3: 查看监听的地址: ShowListenAddr

#### 功能说明

查看 DBProxy 现在监听的地址，及对应的读写类型

#### 命令语法

```
ShowListenAddr
```

#### 字段说明

**port\_type:** 监听端口的类别  
**listenAddr\_list:** 对应的监听地址的列表，逗号分隔

#### 示例

```
mysql> showlistenaddr;
```

port_type	listenAddr_list
rw	127.0.0.1:3456
ro	127.0.0.1:3455, 127.0.0.1:3453, 127.0.0.1:3454

```
2 rows in set (0.00 sec)
```