mysql5.7增加了sys 系统数据库，通过这个库可以快速的了解系统的元数据信息

这个库确实可以方便DBA发现数据库的很多信息，解决性能瓶颈都提供了巨大帮助

这个库在mysql5.7中是默认存在的，在mysql5.6版本以上可以手动导入，数据库包请在github自行查找

这个库包括了哪些内容？

这个库是通过视图的形式把information\_schema 和performance\_schema结合起来，查询出更加令人容易理解的数据

存储过程可以可以执行一些性能方面的配置，也可以得到一些性能诊断报告内容

存储函数可以查询一些性能信息

分析每个视图和表之前先说明一下：关于带不带x$，去掉x$同名的视图他们的数据是相同的，区别在于不带x$的单位更加符合直接阅读经过了转换，而带x$是为了某些工具存在而使用的原始单位（多数应该是mysql默认的）

下面就结合mysql官方手册来详细分析sys库

1.表

    1.1 sys\_config 表

        这是在这个系统库上存在的唯一一个表了

        先看看表结构

|  |
| --- |
| CREATE TABLE `sys\_config` (    `variable` varchar(128) NOT NULL,    `value` varchar(128) DEFAULT NULL,    `set\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,    `set\_by` varchar(128) DEFAULT NULL,    PRIMARY KEY (`variable`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 |

        variable 配置选项名称

        value     配置选项值

        set\_time 该行配置修改的时间

        set\_by     该行配置信息修改者，如果从被安装没有修改过，那么这个数据应该为NULL

    表中默认数据为

|  |  |  |  |
| --- | --- | --- | --- |
| variable | value | set\_time | set\_by |
| diagnostics.allow\_i\_s\_tables | OFF | 2015-11-20 16:04:38 | [root@localhost](mailto:root@localhost) |
| diagnostics.include\_raw | OFF | 2015-11-20 16:04:38 | [root@localhost](mailto:root@localhost) |
| statement\_performance\_analyzer.limit | 100 | 2015-11-20 16:04:38 | [root@localhost](mailto:root@localhost) |
| statement\_performance\_analyzer.view |  | 2015-11-20 16:04:38 | [root@localhost](mailto:root@localhost) |
| statement\_truncate\_len | 64 | 2016-01-22 17:00:16 | [root@localhost](mailto:root@localhost) |

    以上值的会话变量为@sys.+表中variable字段，譬如：@sys.statement\_truncate\_len

可以set @sys.statement\_truncate\_len = 32 临时改变值，在会话中会一直使用这个值，如果想要恢复使用表的默认值，只需要将这个会话值设置为null；set @sys.statement\_truncate\_len = null;

diagnostics.allow\_i\_s\_tables

diagnostics.include\_raw

这两个值默认为OFF ，前者如果开启表示允许diagnostics() 存储过程执行扫描information\_schema.tables 表，如果表很多，那么可能会很耗性能，后者开启将会从metrics 视图输出未加工处理的数据 。diagnostics() 具体内容见下面对diagnostics()的解释。

statement\_performance\_analyzer.limit

视图在没有加limit限制时，返回的最大行数

statement\_performance\_analyzer.view

（略）

以上参数为mysql5.7.9加入

statement\_truncate\_len

通过format\_statement()函数返回值的最大长度

这个表非默认选项还有一个@sys.debug参数

可以手动加入

INSERT INTO sys\_config (variable, value) VALUES('debug', 'ON');

UPDATE sys\_config SET value = 'OFF' WHERE variable = 'debug';

SET @sys.debug = NULL;

具体内容请参考官方文档，此处不做介绍

关于这个表有两个触发器

1.1.1 sys\_config\_insert\_set\_user触发器

如果加入新行通过insert语句，那么这个触发器会把set\_by列设置为当前操作者

1.1.2 sys\_config\_update\_set\_user触发器

如果加入新行通过update语句，那么这个触发器会把set\_by列设置为当前操作者

2.视图

以下部分只介绍不包含x$的视图内容

2.1 host\_summary (主机概要)

有如下列：

• host

监听连接过的主机

• statements

当前主机执行的语句总数

• statement\_latency

语句等待时间（延迟时间）

• statement\_avg\_latency

执行语句平均延迟时间

• table\_scans

表扫描次数

• file\_ios

io时间总数

• file\_io\_latency

文件io延迟

• current\_connections

当前连接数

• total\_connections

总链接数

• unique\_users

该主机的唯一用户数

• current\_memory

当前账户分配的内存

• total\_memory\_allocated

该主机分配的内存总数

2.2  The host\_summary\_by\_file\_io\_type

•host

主机

•event\_name

IO事件名称

•total

该主机发生的事件

•total\_latency

该主机发生IO事件总延迟时间

•max\_latency

该主机IO事件中最大的延迟时间

2.3 The host\_summary\_by\_file\_io

•host

主机

•ios

IO事件总数

•io\_latency

IO总的延迟时间

2.4 The host\_summary\_by\_stages

• host

主机

• event\_name

stage event名称

• total

stage event发生的总数

• total\_latency

stage event总的延迟时间

• avg\_latency

stage event平均延迟时间

2.5 The host\_summary\_by\_statement\_latency

• host

主机

• total

这个主机的语句总数

• total\_latency

这个主机总的延迟时间

• max\_latency

主机最大的延迟时间

• lock\_latency

等待锁的锁延迟时间

• rows\_sent

该主机通过语句返回的总行数

• rows\_examined

在存储引擎上通过语句返回的行数

• rows\_affected

该主机通过语句影响的总行数

• full\_scans

全表扫描的语句总数

2.6  The host\_summary\_by\_statement\_type

• host

主机

• statement

最后的语句事件名称

• total

sql语句总数

• total\_latency

sql语句总延迟数

• max\_latency

最大的sql语句延迟数

• lock\_latency

锁延迟总数

• rows\_sent

语句返回的行总数

• rows\_examined

通过存储引擎的sql语句的读取的总行数

• rows\_affected

语句影响的总行数

• full\_scans

全表扫描的语句事件总数

2.7 The innodb\_buffer\_stats\_by\_schema

这个表是通过数据库统计innodb引擎的innodb缓存

• object\_schema

数据库名称

• allocated

分配给当前数据库的总的字节数

• data

分配给当前数据库的数据字节数

• pages

分配给当前数据库的总页数

• pages\_hashed

分配给当前数据库的hash页数

• pages\_old

分配给当前数据库的旧页数

• rows\_cached

当前数据库缓存的行数

2.8 The innodb\_buffer\_stats\_by\_table

这个表是通过每个表innodb引擎的innodb缓存

• object\_schema

数据库名称

• object\_name

表名称

• allocated

分配给表的总字节数

• data

分配该表的数据字节数

• pages

分配给表的页数

• pages\_hashed

分配给表的hash页数

• pages\_old

分配给表的旧页数

• rows\_cached

表的行缓存数

2.9 The innodb\_lock\_waits

这个表其实从视图的语句来看就是information\_schema这个数据库中的innodb\_locks、innodb\_trx这两个表的整合，能够更清晰的显示当前实例的锁情况

• wait\_started

锁等待发生的时间

• wait\_age

锁已经等待了多长时间

• wait\_age\_secs

以秒为单位显示锁已经等待的时间（5.7.9中添加此列）

• locked\_table

被锁的表

• locked\_index

被锁住的索引

• locked\_type

锁类型

• waiting\_trx\_id

正在等待的事务ID

• waiting\_trx\_started

等待事务开始的时间

• waiting\_trx\_age

已经等待事务多长时间

• waiting\_trx\_rows\_locked

正在等待的事务被锁的行数量

• waiting\_trx\_rows\_modified

正在等待行重定义的数量

• waiting\_pid

正在等待事务的线程id

• waiting\_query

正在等待锁的查询

• waiting\_lock\_id

正在等待锁的ID

• waiting\_lock\_mode

等待锁的模式

• blocking\_trx\_id

阻塞等待锁的事务id

• blocking\_pid

正在锁的线程id

• blocking\_query

正在锁的查询

•blocking\_lock\_id

正在阻塞等待锁的锁id.

•blocking\_lock\_mode

阻塞锁模式

• blocking\_trx\_started

阻塞事务开始的时间

• blocking\_trx\_age

阻塞的事务已经执行的时间

• blocking\_trx\_rows\_locked

阻塞事务锁住的行的数量

• blocking\_trx\_rows\_modified

阻塞事务重定义行的数量

• sql\_kill\_blocking\_query

kill 语句杀死正在运行的阻塞事务

在mysql5.7.9中被加入

• sql\_kill\_blocking\_connection

kill 语句杀死会话中正在运行的阻塞事务

在mysql5.7.9中被加入

2.10 The io\_by\_thread\_by\_latency

这个视图主要信息是通过IO的消耗展示IO等待的时间

• user

对于当前线程来说，这个值是线程被分配的账户，对于后台线程来讲，就是线程的名称

• total

IO事件的总数

• total\_latency

IO事件的总延迟

• min\_latency

单个最小的IO事件延迟

• avg\_latency

平均IO延迟

• max\_latency

最大IO延迟

• thread\_id

线程ID

• processlist\_id

对于当前线程就是此时的ID，对于后台就是null