目录

[第7章　sys系统库初相识 1](#_Toc7202)

[7.1 sys系统库使用基础环境 1](#_Toc7212)

[7.2　sys系统库初体验 1](#_Toc7133)

[7.3　sys系统库的进度报告功能 4](#_Toc15181)

## 第7章　sys系统库初相识

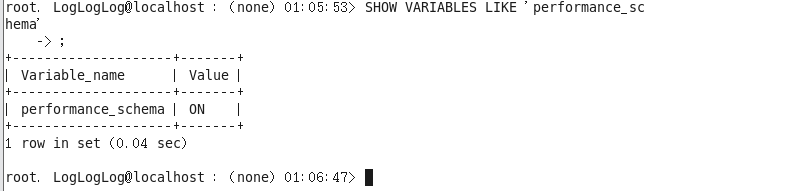
## 7.1 sys系统库使用基础环境

总结一些要点：

0. sys库是通过**视图的形式把information\_schema 和performance\_schema结合起来**，查询出令人更容易理解的数据。

注：视图不是数据表，可简单理解为一个复杂但可读性更强的select语句结果。早在学习sysbench压测时，已经接触过sys视图。

1.要启用performance\_schema：



“sys系统库的大部分功能才能正常使用”

2. 使用sys系统库本身来启用所有需要的instruments和consumers

3. 使用存储过程CALL sys.ps\_setup\_reset\_to\_default(TRUE)可快速恢复performance\_schema的默认配置

4. 查询或多或少都会对性能开销有一定的影响

## 7.2　sys系统库初体验

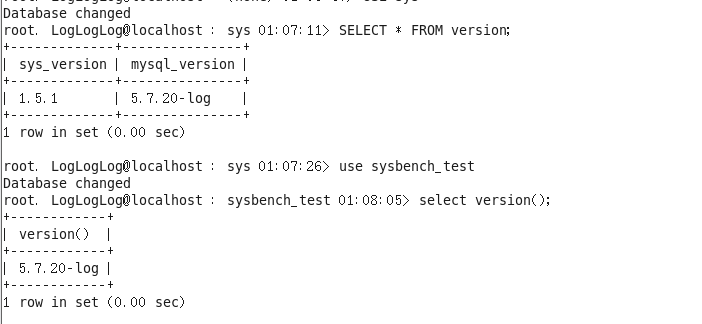
注：本节“初体验”用于了解sys的基本作用。书中举了一些例子，包括系统版本、IO性能、sys库视图、函数、存储过程的整体信息。

#查看**sys 系统库**和MySQL Server的**版本号**(注：视图、函数和存储过程可理解为内置的脚本)。直接使用sys系统库下的视图进行查询，例如使用version视图可以：

mysql> USE sys;

mysql> SELECT \* FROM version;

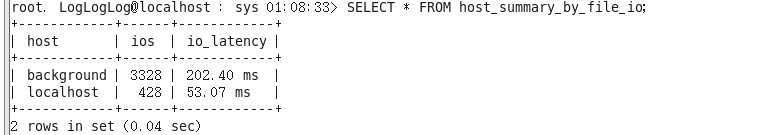
对比：



#在sys系统库下包含很多视图，它们以各种方式对performance\_schema表进行聚合计算展示。这些视图大部分是**成对出现**的，两个视图名称相同，但有一个视图是带“x$”前缀的，例如：host\_summary\_by\_file\_io(注：host\_summary\_by\_file\_io是按照主机进行汇总统计的文件I/O性能数据)和x$host\_summary\_by\_file\_io，代表按照主机进行汇总统计的文件I/O性能数据，两个视图访问的数据源是相同的，但是在创建视图的语句中，不带“x$”前缀的视图显示的是相关数值**经过单位换算后的数据**（单位是毫秒、秒、分钟、小时、天等），带“x$”前缀的视图显示的是原始的数据（单位是皮秒）。

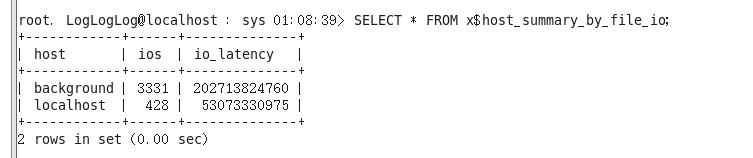
# 使用x$host\_summary\_by\_file\_io视图汇总数据，显示未格式化的延迟时间（单位是皮秒），没有“x$”前缀的视图输出的信息**经过单位换算可读性更好**。

mysql> SELECT \* FROM host\_summary\_by\_file\_io;



# **带“x$”前缀**的视图显示原始的数据（单位是皮秒），**对于程序或工具的获取及使用更易于数据处理**

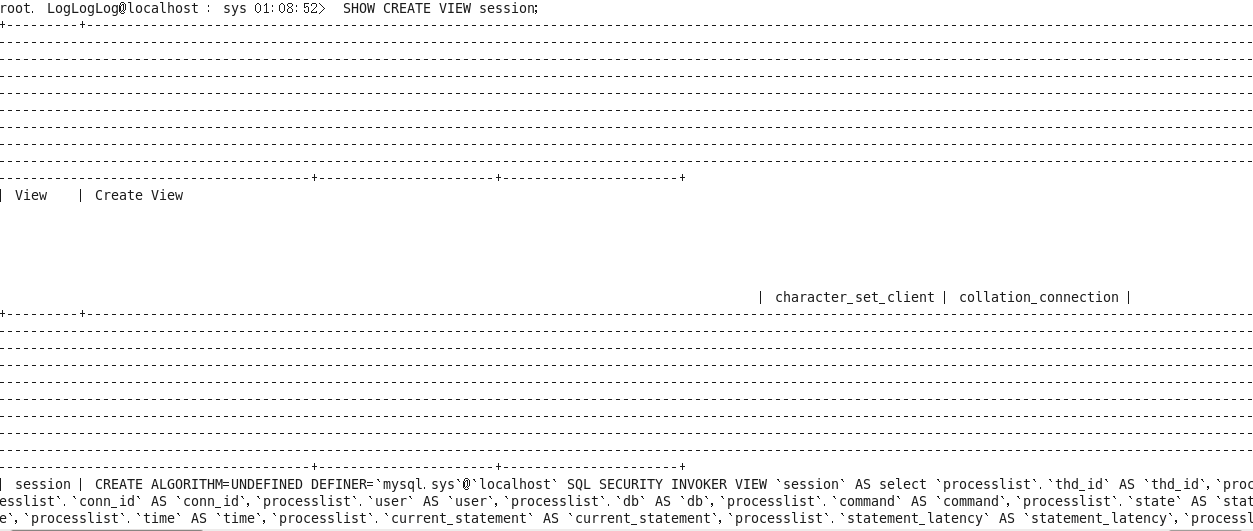
mysql> SELECT \* FROM **x$**host\_summary\_by\_file\_io;



#要查看sys系统库对象定义语句，可以使用适当的SHOW语句或INFORMATION\_SCHEMA库查询。例如，要查看session视图和format\_bytes（）函数的定义，可以使用如下语句：

mysql> SHOW CREATE VIEW session;

mysql> SHOW CREATE FUNCTION format\_bytes;

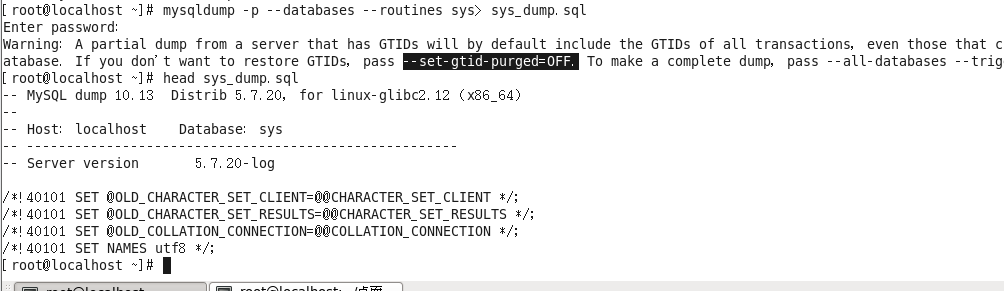


注：语句文本是经过格式化的，可读性比较差。

#使用mysqldump和mysqlpump工具**导出sys系统库**

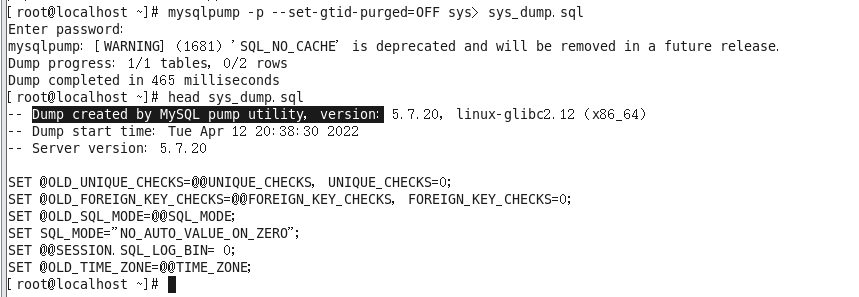
注：mysqldump和mysqlpump用**于**迁移和备份数据库，后者更快。关于mysqldump的详细介绍，可参考本书第50章(MySQL主流备份工具之mydumper详解)。

[root@localhost～]# mysqldump -p --databases --routines sys> sys\_dump.sql

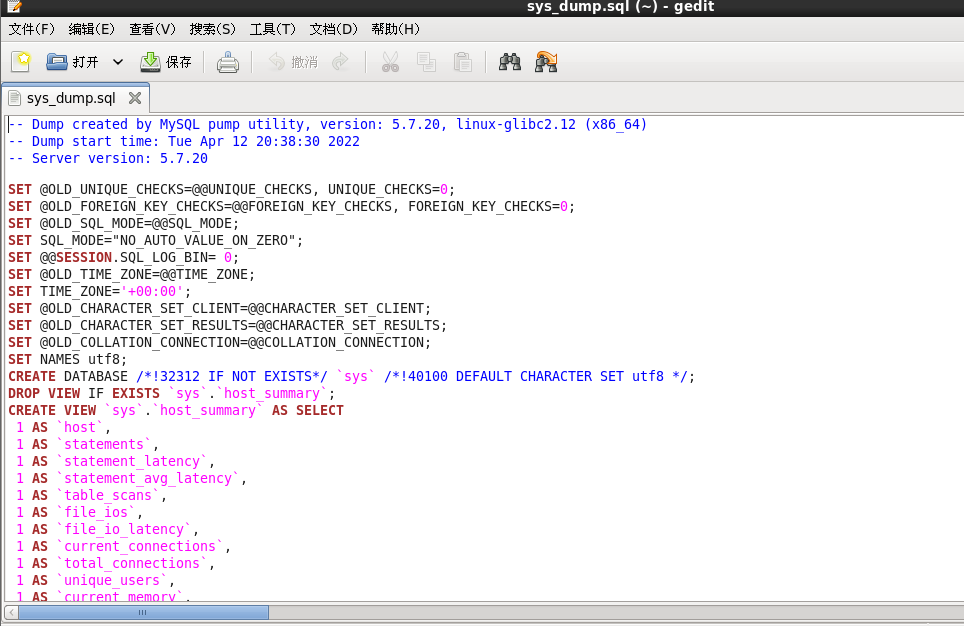


注：原命令没有考虑root设置了密码。如果只是部分备份，因此会警告备份涉及了全局事务ID（GTIDs）,可通过设置--set-gtid-purged=OFF避免。留意mysqlpump对该选项的使用。

[root@localhost～]# mysqlpump -p --set-gtid-purged=OFF **sys**> sys\_dump.sql



然后用编辑器打开：



#重新导入sys系统库(注：此步过程略)

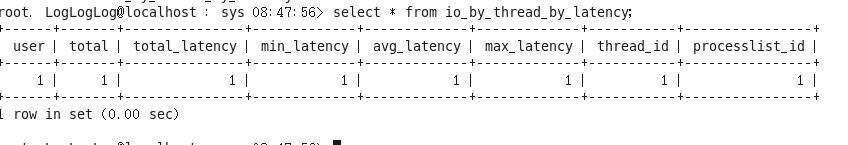
[root@localhost～]# mysql < sys\_dump.sql

注：如书中指出，这里只是希望通过mysqldump和mysqlpump提取sys系统库里面的信息。但实际内容不全。

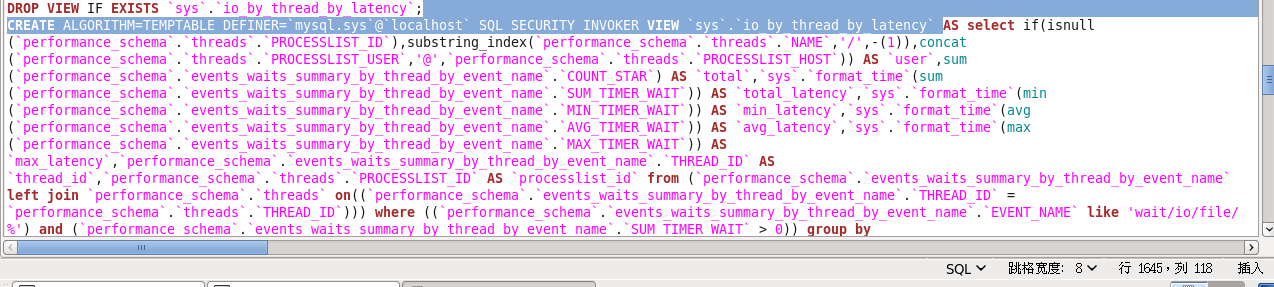
可以访问sys系统库开发网站：https://github.com/mysql/mysql-sys上的各个．sql文件。

sys初体验之总结：以sys.\*\*\*(自行选择)视图为例(最好在前面多次实验的环境下有真实数据统计结果)。它用于\*\*\*(用途解释)。字段含义如下\*\*\*\*(具体说明)。

select \* from io\_by\_thread\_by\_latency;



从备份文件中获取到sys.\*\*\*的定义如下：



它实质整合了\*\*\*\*、\*\*\*\*、\*\*\*\*等数据表的内容。

## 7.3　sys系统库的进度报告功能

注：该部分内容也参考了6.4.2.

#从MySQL 5.7.9开始，sys系统库视图提供了查看长时间运行的事务的进度报告，通过processlist和session以及带“x$”前缀的视图进行查看，其中processlist包含了后台线程和前台线程当前的事件信息；session不包含后台线程和command为Daemon的线程。

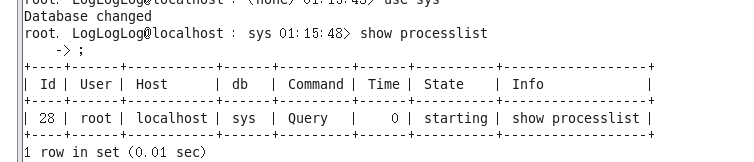
processlist

session

x$processlist

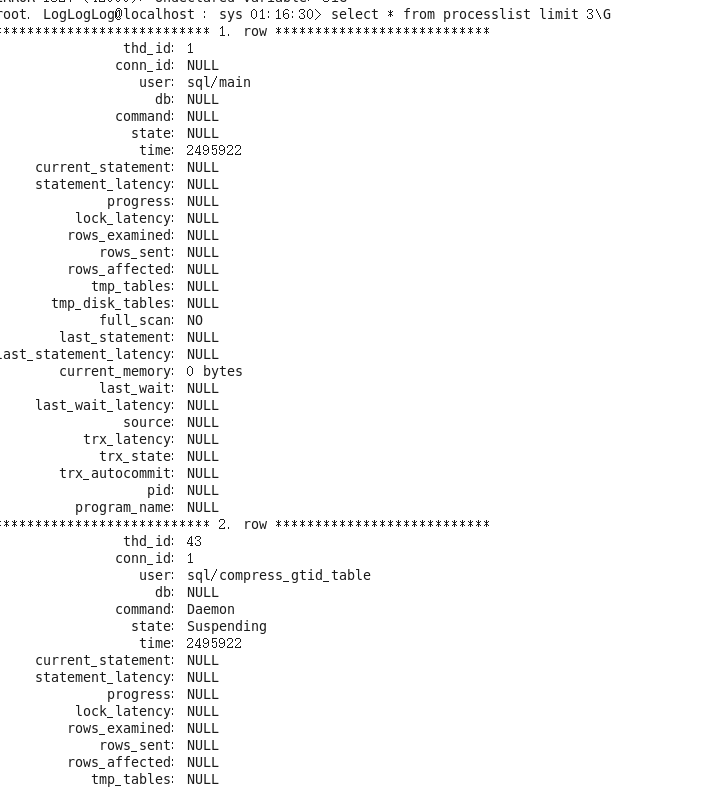
x$session

注：前面的实验我们都是用show processlist查看会话信息，

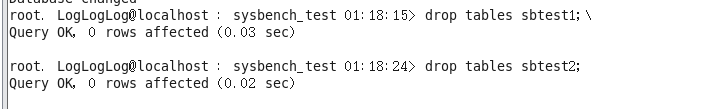


信息都是从sys.processlist中筛选，其中书中所说的” command为Daemon的线程” (比如压缩gtid\_table线程，见实训2)就不会显示：

select \* from processlist limit 3\G



此部分实验需要一些慢卡情景，比如sysbench压测之类，我们可以自由地在sysbench库中执行一些SQL语句。不过前提是这些语句应该有一定耗时。所以我们最好把原来压测后的表(sbtest1-5，压测后会有几百行)删掉：

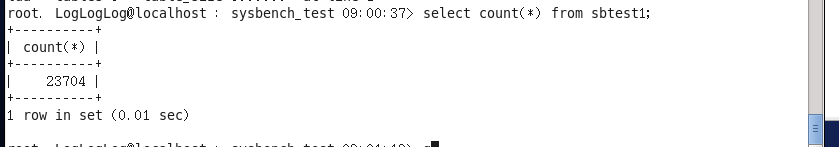


然后重新初始化为较大的表：

文本

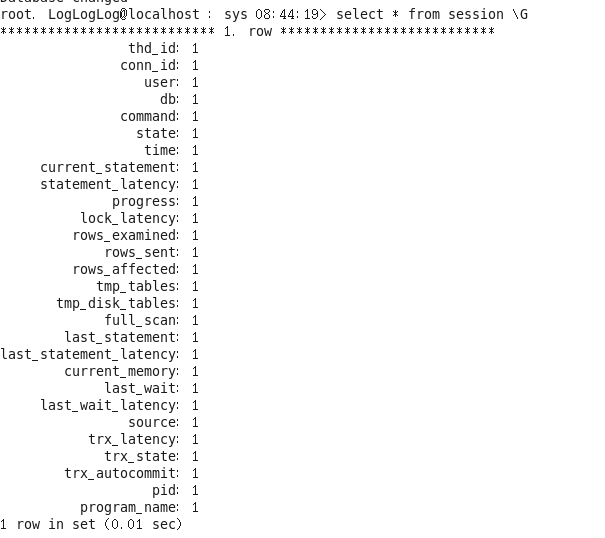
描述已自动生成

我们在mysql统计一下sbtest1的行数：



然后对应地在sys.session中(也包括sys.processlist等)会有相关记录，但进度信息是空的：

select \* from session \G



#**启用需要查看与进度相关的instruments**

stage/sql/Copying to tmp table

stage/innodb/alter table (end)

stage/innodb/alter table (flush)

stage/innodb/alter table (insert)

stage/innodb/alter table (log apply index)

stage/innodb/alter table (log apply table)

stage/innodb/alter table (merge sort)

stage/innodb/alter table (read PK and internal sort)

stage/innodb/buffer pool load

#对于不支持进度的stage事件，或者未启用所需的instruments或consumers的stage事件，则对应的进度信息字段值为NULL

注：根据以上要求执行如下语句：

UPDATE setup\_instruments SET ENABLED='YES', TIMED='YES' where name like 'stage/innodb%' or name like 'stage/sql/Copying to tmp table';

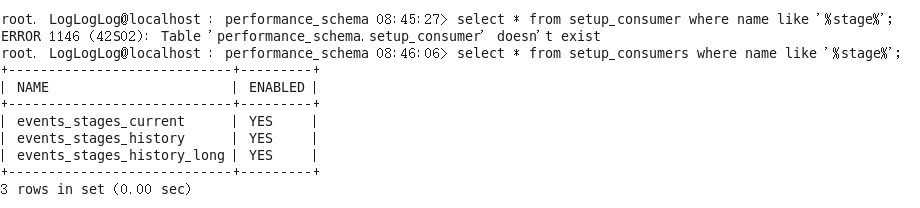
select \* from setup\_instruments where name like 'stage/innodb%’or name like 'stage/sql/Copying to tmp table';

图片包含 表格

描述已自动生成

还需要启用consumers的stage事件，执行如下语句：

update setup\_consumers set enabled='yes' where name like '%stage%';



执行测试语句：

mysql>alter table sbtest1 add index i\_c(c);

在语句没有执行完毕之前，通过session查看语句的进度：

select \* from session where conn\_id !=connection\_id()\G

表格

描述已自动生成

中间结果略，能看到刚才提交的语句的进度：

执行 select \* from sbtest1;语句 不知为何所有结果都是1



等语句执行完毕后，progress字段会又变为空：执行完毕也还是1

