# 概述

参考：<https://www.codenong.com/cs106667755/>

# 源码分析

参考：<http://mysql.taobao.org/monthly/2014/08/05/>

# information\_schema

## 概述

information\_schema提供了对数据库元数据、统计信息、以及有关MySQL Server的信息访问（例如：数据库名或表名，字段的数据类型和访问权限等）。该库中保存的信息也可以称为MySQL的数据字典或系统目录。

在每个MySQL 实例中都有一个独立的information\_schema，用来存储MySQL实例中所有其他数据库的基本信息。information\_schema数据库下包含多个只读表(非持久表)，所以在磁盘中的数据目录下没有对应的关联文件，且不能对这些表设置触发器。虽然在查询时可以使用USE语句将默认数据库设置为information\_schema，但该库下的所有表是只读的，不能执行INSERT、UPDATE、DELETE等数据变更操作。

information\_schema是MySQL数据库中的一个特殊数据库，它存储了关于数据库、表、列、索引、用户权限等元数据信息。它并不存储实际的用户数据，而是提供了一套查询接口，用于查询和获取数据库系统的元数据信息。

## 原理

下面是information\_schema库的一般实现原理：

1. 元数据收集：MySQL数据库在运行过程中会收集和维护各种元数据信息，如数据库、表、列、索引、用户权限等。这些元数据信息存储在内部的数据结构中，并被信息模式收集器定期更新。

2. 信息模式表：information\_schema库是一个虚拟数据库，其中包含了多个信息模式表（Information Schema Tables）。这些信息模式表是视图（Views）或基于内部数据结构生成的虚拟表，用于展示数据库系统的元数据信息。

3. 查询接口：通过查询information\_schema库中的信息模式表，可以获取数据库系统的元数据信息。用户可以使用标准的SELECT语句查询这些表，以获取关于数据库、表、列、索引、用户权限等方面的信息。information\_schema库提供了一套标准化的查询接口，使得元数据的查询变得简单和统一。

4. 访问控制：与其他数据库对象一样，information\_schema库也受到访问控制的限制。只有具有足够权限的用户或角色才能够查询和访问information\_schema库中的元数据信息。这样可以确保元数据的安全性和保密性。

通过information\_schema库，用户可以方便地查询和获取数据库系统的元数据信息，包括数据库、表、列、索引、用户权限等。它提供了一种标准化的方式来访问和管理元数据，使得对数据库结构和配置的了解更加简单和统一。同时，由于information\_schema库是一个虚拟数据库，不会占用额外的存储空间，而是通过查询内部数据结构实时生成元数据信息。

## 特点

### 优点

针对information\_schema下的表的查询操作可以替代一些show查询语句（例如：SHOW DATABASES，SHOW TABLES等），与使用show语句相比，通过查询information\_schema下的表获取数据有以下优势：

1. 它符合"Codd法则"，所有的访问都是基于表的访问完成的
2. 可以使用SELECT语句的SQL语法，只需要学习你要查询的一些表名和列名的含义即可
3. 基于SQL语句的查询，对来自information\_schema中的查询结果可以做过滤、排序、联结操作，查询的结果集格式对应用程序来说更友好
4. 这种技术实现与其他数据库系统中类似的实现更具互操作性。例如：Oracle数据库的用户熟悉查询Oracle数据字典中的表，那么在MySQL中查询数据字典的表也可以使用同样的方法来执行查询获取想要的数据

### 权限

访问information\_schema需要的权限

所有用户都有访问information\_schema下的表权限(但只能看到这些表中用户具有访问权限的对象相对应的数据行)，但只能访问Server层的部分数据字典表，Server层中的部分数据字典表以及InnoDB层的数据字典表需要额外授权才能访问，如果用户权限不足，当查询Server层数据字典表时将不会返回任何数据，或者某个列没有权限访问时，该列返回NULL值。当查询InnoDB数据字典表时将直接拒绝访问(要访问这些表需要有process权限，注意不是select权限)

从information\_schema中查询相关数据需要的权限也适用于SHOW语句。无论使用哪种查询方式，都必须拥有某个对象的权限才能看到相关的数据。

在MySQL 5.6版本中总共有59张表，其中10张MyISAM引擎临时表(数据字典表)，49张Memory引擎临时表(保存统计信息和一些临时信息)。在MySQL 5.7版本中，该schema下总共有61张表，其中10个InnoDB存储引擎临时表(数据字典表)，51个Memory引擎临时表。在MySQL 8.0中该schema下数据字典表(包含部分原memory引擎临时表)都迁移到了mysql schema下，且在mysql schema下这些数据字典表被隐藏，无法直接访问，需要通过information\_schema下的同名表进行访问(统计信息表保留在information\_schema下且仍然为Memory引擎)

虽然直接通过查询information\_schema中的表获取数据有众多优势，但是因为SHOW语法已经耳熟能详且被广泛使用，所以SHOW语句仍然是一个备选方法，且随着information\_schema的实现，SHOW语句中的功能还有所增强(可以使用like或where子句进行过滤)。

## 组成对象

information\_schema下的所有表都是使用的Memory和InnoDB存储引擎，且都是临时表，不是持久表，在数据库重启之后这些数据会丢失，在MySQL 的4个系统库中，也是唯一一个在文件系统上没有对应库表的目录和文件的系统库。

下面我们按照这些表的各自用途的相似度，我们把information\_schema下的表做了如下归类，本期我们先大致了解下information\_schema系统库中都有哪些表，这些表大致都有什么用途。

### Server层统计信息字典表

#### COLUMNS

提供查询表中的列(字段)信息

该表为InnoDB存储引擎的临时表

#### KEY\_COLUMN\_USAGE

提供查询哪些索引列存在约束条件

该表中的信息包含主键、唯一索引、外键等约束的信息，例如：所在库表列名，引用的库表列名等。表中的信息与TABLE\_CONSTRAINTS表中记录的信息有些类似，但TABLE\_CONSTRAINTS表中没有记录约束引用的库表列信息。但是却记录了TABLE\_CONSTRAINTS表中所没有的约束类型信息

该表为Memory引擎临时表

#### REFERENTIAL\_CONSTRAINTS

提供查询关于外键约束的一些信息

该表为Memory引擎临时表

#### STATISTICS

提供查询关于索引的一些统计信息，一个索引对应一行记录

该表为Memory引擎临时表

#### TABLE\_CONSTRAINTS

提供查询表相关的约束信息

该表为Memory引擎临时表

#### FILES

提供查询MySQL的数据表空间文件相关的信息，包含InnoDB存储引擎和NDB存储引擎相关的数据文件信息，由于NDB存储引擎在国内较少使用，我们大多数场景(95%以上场景InnoDB存储引擎都满可以使用)都是使用InnoDB存储引擎

该表为Memory存储引擎表

#### ENGINES

提供查询MySQL Server支持的引擎相关的信息

该表为Memory引擎临时表

#### TABLESPACES

提供查询关于活跃表空间的相关信息（主要记录的是NDB存储引擎表空间信息）

注意：该表不提供有关InnoDB存储引擎的表空间的信息。对于InnoDB表空间元数据信息，请查询INNODB\_SYS\_TABLESPACES和INNODB\_SYS\_DATAFILES表。另外，从MySQL 5.7.8开始，INFORMATION\_SCHEMA.FILES表也提供查询InnoDB表空间的元数据信息

该表为Memory引擎临时表。

#### SCHEMATA

提供查询MySQL Server中的数据库列表信息，一个schema就代表一个database

该表为Memory引擎临时表

### Server层表级别对象字典表

#### VIEWS

提供查询数据库中的视图相关的信息，查询该表的帐号需要拥有show view权限

该表为InnoDB引擎临时表

#### TRIGGERS

提供查询关于某个数据库下的触发器相关的信息，要查询某个表的触发器，查询的账户必须要有trigger权限

该表为InnoDB引擎临时表

#### TABLES

提供查询数据库内的表相关的基本信息

该表为Memory引擎临时表

参考：

<https://blog.csdn.net/weixin_39004901/article/details/84616689>

#### ROUTINES

提供查询关于存储过程和存储函数的信息（不包括用户自定义函数UDF），该表中的信息与“mysql.proc”中记录的信息相对应（如果该表中有值的话）

该表为InnoDB引擎临时表。

#### PARTITIONS

提供查询关于分区表的信息

该表为InnoDB引擎临时表

#### EVENTS

提供查询计划任务事件相关的信息

该表是InnoDB引擎临时表。

#### PARAMETERS

提供有关存储过程和函数的参数信息，以及有关存储函数的返回值的信息

这些参数信息与mysql.proc表中的param\_list列记录的内容类似

该表为InnoDB引擎临时表

### Server 层混杂信息字典表

#### GLOBAL\_STATUS、GLOBAL\_VARIABLES、SESSION\_STATUS、SESSION\_VARIABLES

提供查询全局、会话级别的的状态变量与系统变量信息，这些表为Memory引擎临时表

#### OPTIMIZER\_TRACE

提供优化程序跟踪功能产生的信息。

跟踪功能默认关闭，使用optimizer\_trace系统变量启用跟踪功能。如果开启该功能，则每个会话只能跟踪他自己执行的语句，不能看到其他会话执行的语句，且每个会话只能记录最后一个跟踪的SQL语句

该表为InnoDB引擎临时表

#### PLUGINS

提供查询关于MySQL Server中支持哪些插件的信息

该表为InnoDB引擎临时表

#### PROCESSLIST

提供查询一些关于线程运行过程中的状态信息

该表为InnoDB引擎临时表

#### PROFILING

提供查询关于语句性能分析的信息。其记录内容对应于SHOW PROFILES和SHOW PROFILE语句产生的信息。该表需要在会话变量 profiling=1时才会记录语句性能分析信息，否则该表不记录。

注意：从MySQL 5.7.2开始，此表不再推荐使用，在未来的MySQL版本中删除。改用Performance Schema;代替

该表为Memory引擎临时表

#### CHARACTER\_SETS

提供查询MySQL Server支持的可用字符集有哪些

该表为Memory引擎临时表

#### COLLATIONS

提供查询MySQL Server支持的可用校对规则有哪些

该表为Memory引擎临时表

#### COLLATION\_CHARACTER\_SET\_APPLICABILITY

提供查询MySQL Server中哪种字符集适用于什么校对规则。查询结果集相当于从SHOW COLLATION获得的结果集中的前两个字段值。该表目前并没有发现有太大作用，为Memory引擎临时表

#### COLUMN\_PRIVILEGES

提供查询关于列(字段)的权限信息，表中的内容来自mysql.column\_priv列权限表（需要针对一个表的列单独授权之后才会有内容）

该表为Memory引擎临时表

#### SCHEMA\_PRIVILEGES

提供查询关于库级别的权限信息，每种类型的库级别权限记录一行信息，该表中的信息来自mysql.db表

该表为Memory引擎临时表

#### TABLE\_PRIVILEGES

提供查询关于表级别权限信息，该表中的内容来自mysql.tables\_priv

该表为Memory引擎临时表

#### USER\_PRIVILEGES

提供查询全局权限的信息，该表中的信息来自mysql.user表

该表为Memory引擎临时表

### InnoDB层系统字典表

#### INNODB\_SYS\_DATAFILES

提供查询InnoDB file-per-table和常规表空间数据文件的路径信息，等同于InnoDB数据字典中SYS\_DATAFILES表中的信息

该表中的信息包含InnoDB所有表空间类型的元数据，包括独立表空间、常规表空间、系统表空间、临时表空间和undo表空间（如果开启了独立表空间的话）

该表为memory引擎临时表，查询该表的用户需要有process权限。

#### INNODB\_SYS\_VIRTUAL

提供查询有关InnoDB虚拟生成列和与之关联的列的元数据信息，等同于InnoDB数据字典内部SYS\_VIRTUAL表中的信息。INNODB\_SYS\_VIRTUAL表中展示的行信息是虚拟生成列相关联列的每个列的信息。

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_INDEXES

提供查询有关InnoDB索引的元数据信息，等同于InnoDB数据字典内部SYS\_INDEXES表中的信息

该表为memory引擎临时表，查询该表的用户需要具有process权限

#### INNODB\_SYS\_TABLES

提供查询有关InnoDB表的元数据，等同于InnoDB数据字典内部**SYS\_TABLES表的信息。**

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_FIELDS

提供查询有关InnoDB索引键列（字段）的元数据信息，等同于InnoDB数据字典内部SYS\_FIELDS表的信息

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_TABLESPACES

提供查询有关InnoDB独立表空间和普通表空间的元数据信息（也包含了全文索引表空间），等同于InnoDB数据字典内部SYS\_TABLESPACES表中的信息

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_FOREIGN\_COLS

提供查询有关InnoDB外键列的状态信息，等同于InnoDB数据字典内部SYS\_FOREIGN\_COLS表的信息

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_COLUMNS

提供查询有关InnoDB表列的元数据信息，等同于InnoDB数据字典内部SYS\_COLUMNS表的信息

该表为memory引擎临时表，查询该表的用户需要具有process权限

#### INNODB\_SYS\_FOREIGN

提供查询有关InnoDB外键的元数据信息，等同于InnoDB数据字典内部SYS\_FOREIGN表的信息

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_SYS\_TABLESTATS

提供查询有关InnoDB表的较低级别的状态信息视图。MySQL优化器会使用这些统计信息数据来计算并确定在查询InnoDB表时要使用哪个索引。这些信息保存在内存中的数据结构中，与存储在磁盘上的数据无对应关系。InnoDB内部也无对应的系统表。

该表为memory引擎临时表，查询该表的用户需要有process权限

### InnoDB 层锁、事务、统计信息字典表

#### INNODB\_LOCKS

参考：

<https://blog.csdn.net/yuyinghua0302/article/details/82318408>

<https://blog.csdn.net/wanbin6470398/article/details/81865690>

提供查询innodb引擎事务中正在请求的且并未获得的且同时阻塞了其他事务的锁信息(即没有发生不同事务之间的锁等待的锁信息，在这里是查看不到的，例如，只有一个事务时，该事务所加的锁信息无法查看到)。该表中的内容可以用于诊断高并发下的锁争用信息。

该表为memory引擎临时表，访问该表需要拥有具有process权限。

#### INNODB\_TRX

参考：

<https://blog.csdn.net/yuyinghua0302/article/details/82318408>

<https://blog.csdn.net/wanbin6470398/article/details/81865690>

提供查询当前在InnoDB引擎中执行的每个事务（不包括只读事务）的信息，包括事务是否正在等待锁、事务什么时间点开始、以及事务正在执行的SQL语句文本信息等（如果有SQL的话）。

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_BUFFER\_PAGE\_LRU

提供查询缓冲池中的页面信息，与INNODB\_BUFFER\_PAGE表不同，INNODB\_BUFFER\_PAGE\_LRU表保存有关innodb buffer pool中的页如何进入LRU链表以及在buffer pool不够用时确定需要从缓冲池中逐出哪些页

该表为Memory引擎临时表

#### INNODB\_LOCK\_WAITS

参考：

<https://blog.csdn.net/yuyinghua0302/article/details/82318408>

<https://blog.csdn.net/wanbin6470398/article/details/81865690>

提供查询关于每个被阻塞的InnoDB事务的锁等待记录，包括发生锁等带事务所请求的锁和阻止该锁请求被授予的锁

该表为memory引擎表，访问该表用户需要有process权限

#### INNODB\_TEMP\_TABLE\_INFO

提供查询有关在InnoDB实例中当前处于活动状态的用户(已建立连接的用户，断开的用户连接对应的临时表会被自动删除)创建的InnoDB临时表的信息。它不提供查询优化器使用的内部InnoDB临时表的信息查询。INNODB\_TEMP\_TABLE\_INFO表在首次查询时创建。

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_BUFFER\_PAGE

提供查询关于buffer pool中的页相关的信息

查询该表需要用户具有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_METRICS

提供查询InnoDB更为详细细致的性能信息，是对InnoDB的PERFORMANCE\_SCHEMA的补充。通过对该表的查询，可用于检查innodb的整体健康状况。也可用于诊断性能瓶颈、资源短缺和应用程序的问题等。

该表为memory引擎临时表，查询该表的用户需要有process权限

#### INNODB\_BUFFER\_POOL\_STATS

提供查询一些Innodb buffer pool中的状态信息，该表中记录的信息与SHOW ENGINE INNODB STATUS输出的信息类似相同，另外，innodb buffer pool的一些状态变量也提供了部分相同的值

查看该表需要有process权限，该表为Memory引擎临时表

### InnoDB 层全文索引字典表

#### INNODB\_FT\_CONFIG

提供查询有关InnoDB表的FULLTEXT索引和关联的元数据信息。查询此表之前，需要先设置innodb\_ft\_aux\_table='db\_name/tb\_name'，db\_name/tb\_name为包含全文索引的表名和库名。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_FT\_BEING\_DELETED

该表仅在OPTIMIZE TABLE语句执行维护操作期间作为INNODB\_FT\_DELETED表的快照数据存放使用。运行OPTIMIZE TABLE语句时，会先清空INNODB\_FT\_BEING\_DELETED表中的数据，保存INNODB\_FT\_DELETED表中的快照数据到INNODB\_FT\_BEING\_DELETED表，并从INNODB\_FT\_DELETED表中删除DOC\_ID。由于INNODB\_FT\_BEING\_DELETED表中的内容通常生命周期较短，因此该表中的数据对于监控或者调试来说用处并不大。

表中默认不记录数据，需要设置系统配置参数innodb\_ft\_aux\_table=string（string表示db\_name.tb\_name字符串），并创建好全文索引，设置好停用词等。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_FT\_DELETED

提供查询从InnoDB表的FULLTEXT索引中删除的行信息。它的存在是为了避免在InnoDB FULLTEXT索引的DML操作期间进行昂贵的索引重组操作，新删除的全文索引中单词的信息将单独存储在该表中，在执行文本搜索时从中过滤出搜索结果，该表中的信息仅在执行OPTIMIZE TABLE语句时清空。

该表中的信息默认不记录，需要使用innodb\_ft\_aux\_table选项(该选项默认值为空串)指定需要记录哪个innodb引擎表的信息，例如：test/test。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_FT\_DEFAULT\_STOPWORD

该表为默认的全文索引停用词表，提供查询停用词列表值。启用停用词表需要开启参数innodb\_ft\_enable\_stopword=ON，该参数默认为ON，启用停用词功能之后，如果innodb\_ft\_user\_stopword\_table选项（针对指定的innodb引擎表中的全文索引生效）自定义了停用词库表名称值，则停用词功能使用innodb\_ft\_user\_stopword\_table选项指定的停用词表，如果innodb\_ft\_user\_stopword\_table选项未指定，而innodb\_ft\_server\_stopword\_table选项（针对所有的innodb引擎表中的全文索引生效）自定义了停用词库表名称值，则同停用词功能使用innodb\_ft\_server\_stopword\_table选项指定的停用词表，如果innodb\_ft\_server\_stopword\_table选项也未指定，则使用默认的停用词表，即INNODB\_FT\_DEFAULT\_STOPWORD表。

查询该表需要账户有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_FT\_INDEX\_TABLE

提供查询关于innodb表全文索引中用于反向文本查找的倒排索引的分词信息。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_FT\_INDEX\_CACHE

提供查询包含FULLTEXT索引的innodb存储引擎表中新插入行的全文索引标记信息。它存在的目的是为了避免在DML操作期间进行昂贵的索引重组，新插入的全文索引的单词的信息被单独存储在该表中，直到对表执行OPTIMIZE TABLE语句时、或者关闭服务器时、或者当高速缓存中存放的信息大小超过了innodb\_ft\_cache\_size或innodb\_ft\_total\_cache\_size系统配置参数指定的大小才会执行清理。默认不记录数据，需要使用innodb\_ft\_aux\_table系统配置参数指定需要记录哪个表中的新插入行的全文索引数据。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

### InnoDB 层压缩相关字典表

#### INNODB\_CMP

#### INNODB\_CMP\_RESET

这两个表中的数据包含了与压缩的InnoDB表页有关的操作的状态信息。表中记录的数据为测量数据库中的InnoDb表压缩的有效性提供参考。

查询表的用户必须具有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_CMP\_PER\_INDEX

#### INODB\_CMP\_PER\_INDEX\_RESET

这两个表中记录着InnoDB压缩表数据和索引相关的操作状态信息，对数据库、表、索引的每个组合使用不同的统计信息，以便为评估特定表的压缩性能和实用性提供参考数据。

对于InnoDB压缩表，会对表中的数据和所有二级索引都进行压缩。此时表中的数据被视为另一个索引(包含所有数据列的聚集索引）。

注意：由于为每个索引收集单独的度量值会导致性能大幅度降低，因此默认情况下不收集INNODB\_CMP\_PER\_INDEX和INNODB\_CMP\_PER\_INDEX\_RESET表统计信息。如果确有需要，启用系统配置参数innodb\_cmp\_per\_index\_enabled即可（该配置参数为动态变量，默认为OFF）。

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

#### INNODB\_CMPMEM

#### INNODB\_CMPMEM\_RESET

这两个表中记录着InnoDB缓冲池中压缩页上的状态信息，为测量数据库中InnoDB表压缩的有效性提供参考

查询该表的账户需要有PROCESS权限，该表为Memory引擎临时表

## 实现

参考：

INFORMATION\_SCHEMA系统表的实现：

<http://mysql.taobao.org/monthly/2018/02/08/>

# performance\_schema

参考：<http://mysql.taobao.org/monthly/2021/09/03/>

Performance\_schema内存分配：<http://mysql.taobao.org/monthly/2020/04/05/>

## 概述

你可以使用performance\_schema在运行时检查服务器的内部执行情况。不应该把它与用于检查元数据的information schema混为一谈。

performance\_schema中有许多影响服务器计时的事件消费者，例如函数调用、对操作系统的等待、SQL语句执行中的某个阶段（例如解析或排序）、一条语句或一组语句。所有收集的信息都存储在performance\_schema中，不会被复制。

默认情况下，performance\_schema是启用的；如果要禁用它，可以在my.cnf文件中设置performance\_schema=OFF。默认情况下，并非所有的消费者和计数器都处于启用状态；你可以通过更新 performance\_schema.setup\_instruments 和performance\_schema.setup\_consumers表来关闭/打开它们。

## 原理

performance\_schema是MySQL数据库中的一个特殊数据库，用于收集和展示数据库系统的性能相关信息。它提供了一系列的性能监控表，可以查询和分析数据库系统的运行状态、资源消耗、性能指标等。

下面是performance\_schema库的一般实现原理：

1. 数据收集器：MySQL数据库在运行过程中会通过数据收集器收集各种性能相关的信息。这些信息包括查询执行时间、锁等待情况、线程活动状态、IO操作、表扫描等。数据收集器会在关键事件或操作发生时记录相关信息。

2. 内存表和文件表：performance\_schema库使用内存表和文件表存储收集到的性能信息。内存表存储了较为频繁和实时的信息，以提供快速的查询和访问。文件表则存储了较为稀缺或长期的信息，以避免内存压力。

3. 事件触发器：performance\_schema库中的表是通过事件触发器来更新和维护的。当某个事件或操作发生时，触发器会将相应的性能信息记录到对应的表中。这些触发器会根据预定义的事件类型和条件来触发数据的收集和更新。

4. 查询接口：通过查询performance\_schema库中的性能监控表，可以获取数据库系统的性能信息。用户可以使用标准的SELECT语句查询这些表，以获取关于查询性能、资源消耗、锁等待、线程状态等方面的信息。performance\_schema库提供了一套标准化的查询接口，使得性能信息的查询变得简单和统一。

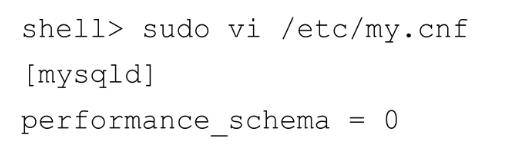
5. 配置参数：performance\_schema库的行为和数据收集范围可以通过配置参数进行调整。用户可以根据需要启用或禁用特定的性能监控组件，并设置相关的采样频率、阈值等参数。

通过performance\_schema库，用户可以对MySQL数据库系统的性能进行监控和分析。它提供了丰富的性能监控表和查询接口，用于收集和展示数据库系统的性能相关信息。通过分析这些信息，用户可以了解数据库系统的瓶颈、优化查询性能和资源消耗等方面。performance\_schema库的实现原理主要包括数据收集器、内存表和文件表、事件触发器、查询接口和配置参数等组件。

## 操作

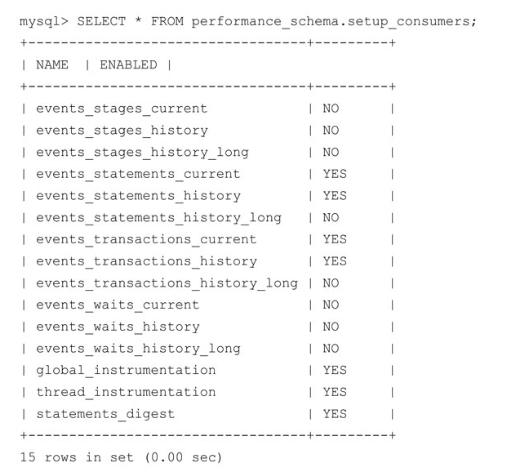
### 启用/禁用performance\_schema

要禁用performance\_schema，请将其设置为0：



### 启用/禁用消费者和计数器

你可以在setup\_consumers表中看到可用的消费者列表，如下所示：



假设要启用events\_waits\_current：

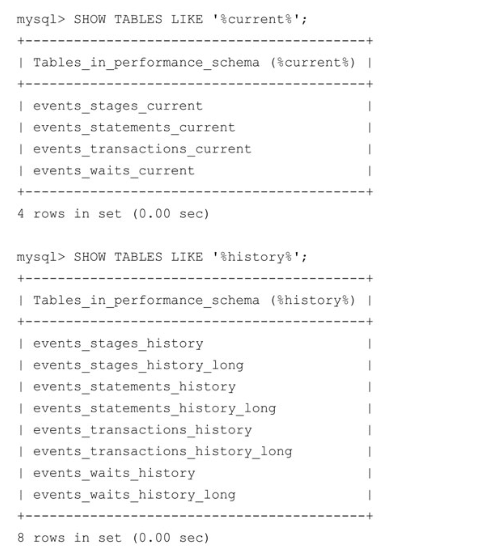


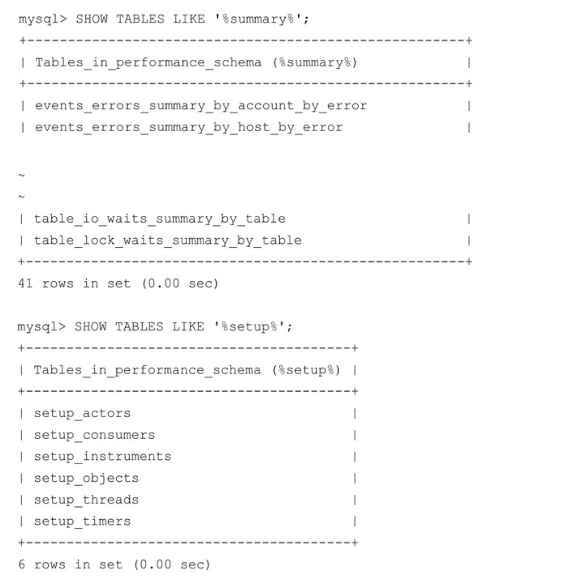
同样，你可以从setup\_instruments表禁用或启用计数器，大约有1182种计数器（视MySQL版本而定）：



### performance\_schema表

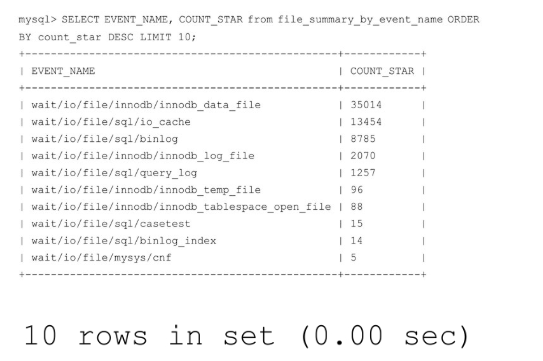
performance\_schema中有5种主要的表类型。它们是当前事件表、事件历史表、事件摘要表、对象实例表和设置（配置）表：





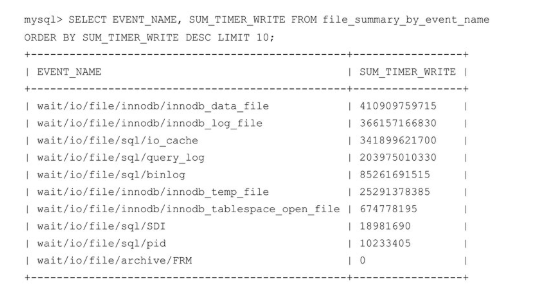
#### 被访问得最多的文件

假设你要找出被访问得最多的文件：



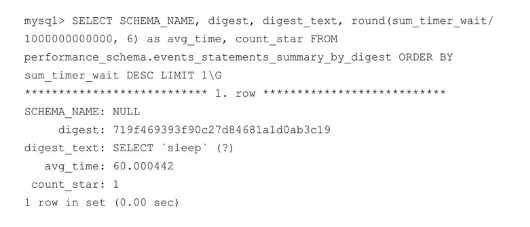
#### 文件的写入时间最长

或者你想知道哪一个文件的写入时间最长：



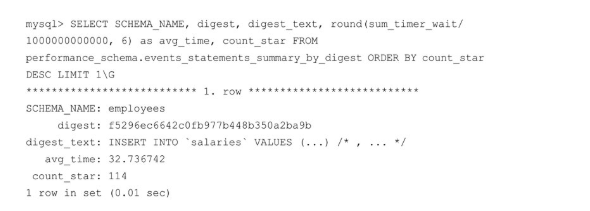
#### 查询花费的时间

你可以使用events\_statements\_summary\_by\_digest表来获取查询报告，就像你对pt-query-digest所做的那样。按所花费的时间列出排名靠前的查询：



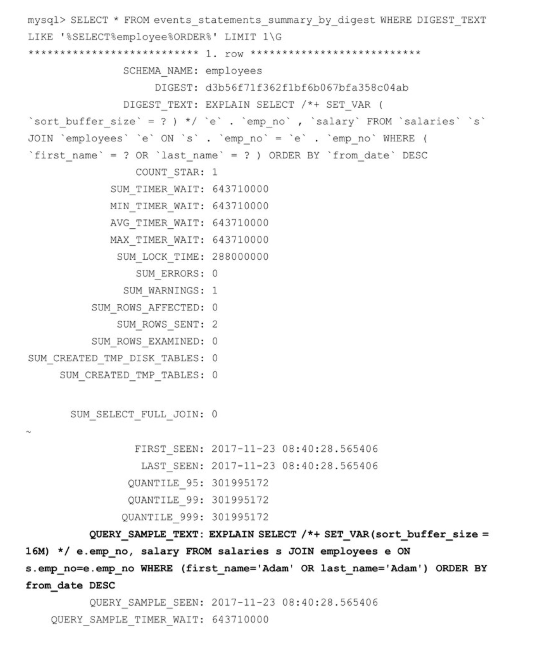
#### 执行次数

按执行次数列出排名靠前的查询：



#### 查找特定查询的统计信息

假设你要查找特定查询的统计信息。你可以使用performance\_schema检查所有统计信息，而不是依赖于mysqlslap基准数据：



# sys\_schema

## 概述

sys库里面的表、视图、函数、存储过程可以使我们更方便、快捷的了解到MySQL的一些信息，比如哪些语句使用了临时表、哪个SQL没有使用索引、哪个schema中有冗余索引、查找使用全表扫描的SQL、查找用户占用的IO等，sys库里这些视图中的数据，大多是从performance\_schema里面获得的。目标是把performance\_schema的复杂度降低，让我们更快的了解DB的运行情况。

sys schema帮助你以一种更简单和更易理解的形式解释从performance\_schema收集来的数据。

**为了使sys schema能工作，应该启用performance\_schema。**

如果想最大限度地使用sys schema，你需要启用performance\_schema上的所有消费者和计时器，但这会影响服务器的性能。所以，仅启动你在寻找的消费者。

带有x$前缀的视图以皮秒为单位显示数据，供其他工具做进一步的处理；其他表是人类可阅读的。

## 分类

我们发现sys schema里的视图主要分为两类，一类是正常以字母开头的，共52个，一类是以x$开头的，共48个。字母开头的视图显示的是格式化数据，更易读，而x$开头的视图适合工具采集数据，显示的是原始未处理过的数据。

下面我们将按类别来分析以字母开头的52个视图：

host\_summary：这个是服务器层面的，以IP分组，比如里面的视图host\_summary\_by\_file\_io；

user\_summary：这个是用户层级的，以用户分组，比如里面的视图user\_summary\_by\_file\_io；

innodb：这个是InnoDB层面的，比如视图innodb\_buffer\_stats\_by\_schema；

io：这个是I/O层的统计，比如视图io\_global\_by\_file\_by\_bytes；

memory：关于内存的使用情况，比如视图memory\_by\_host\_by\_current\_bytes；

schema：关于schema级别的统计信息，比如schema\_table\_lock\_waits；

session：关于会话级别的，这类视图少一些，只有session和session\_ssl\_status；

statement：关于语句级别的，比如statements\_with\_errors\_or\_warnings；

wait：关于等待的，比如视图waits\_by\_host\_by\_latency。

## 常用查询

1,查看每个客户端IP过来的连接消耗了多少资源。  
 mysql> select \* from host\_summary;

2,查看某个数据文件上发生了多少IO请求。  
 mysql> select \* from io\_global\_by\_file\_by\_bytes;

3,查看每个用户消耗了多少资源。  
 mysql> select \* from user\_summary;

4,查看总共分配了多少内存。  
 mysql> select \* from memory\_global\_total;

5,数据库连接来自哪里，以及这些连接对数据库的请求情况是怎样的？  
查看当前连接情况。  
 mysql> select host, current\_connections, statements from host\_summary;

6,查看当前正在执行的SQL和执行show full processlist的效果相当。  
 mysql> select conn\_id, user, current\_statement, last\_statement from session;

7,数据库中哪些SQL被频繁执行？  
执行下面命令查询TOP 10最热SQL。  
 mysql> select db,exec\_count,query from statement\_analysis order by exec\_count desc limit 10;

8,哪个文件产生了最多的IO，读多，还是写的多？  
 mysql> select \* from io\_global\_by\_file\_by\_bytes limit 10;

9,哪个表上的IO请求最多？  
 mysql> select \* from io\_global\_by\_file\_by\_bytes where file like ‘%ibd’ order by total desc limit 10;

10,哪个表被访问的最多？先访问statement\_analysis，根据热门SQL排序找到相应的数据表。  
 mysql> select \* from statement\_analysis order by avg\_latency desc limit 10;

11,哪些SQL执行了全表扫描或执行了排序操作？  
 mysql> select \* from statements\_with\_sorting;  
 mysql> select \* from statements\_with\_full\_table\_scans;

12,哪些SQL语句使用了临时表，又有哪些用到了磁盘临时表？  
查看statement\_analysis中哪个SQL的tmp\_tables 、tmp\_disk\_tables值大于0即可。  
 mysql> select db, query, tmp\_tables, tmp\_disk\_tables from statement\_analysis where tmp\_tables>0 or tmp\_disk\_tables >0 order by (tmp\_tables+tmp\_disk\_tables) desc limit 20;

13,哪个表占用了最多的buffer pool？  
 mysql> select \* from innodb\_buffer\_stats\_by\_table order by allocated desc limit 10;

14,每个库（database）占用多少buffer pool？  
 mysql> select \* from innodb\_buffer\_stats\_by\_schema order by allocated desc limit 10;

15,每个连接分配多少内存？  
 利用session表和memory\_by\_thread\_by\_current\_bytes分配表进行关联查询。  
 mysql> select b.user, current\_count\_used, current\_allocated, current\_avg\_alloc, current\_max\_alloc, total\_allocated,current\_statement from memory\_by\_thread\_by\_current\_bytes a, session b where a.thread\_id = b.thd\_id;

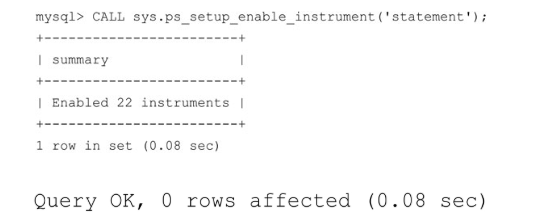
16,MySQL自增长字段的最大值和当前已经使用到的值？  
 mysql> select \* from schema\_auto\_increment\_columns;

17,MySQL有哪些冗余索引和无用索引？  
 mysql> select \* from schema\_redundant\_indexes;  
 mysql> select \* from schema\_unused\_indexes;

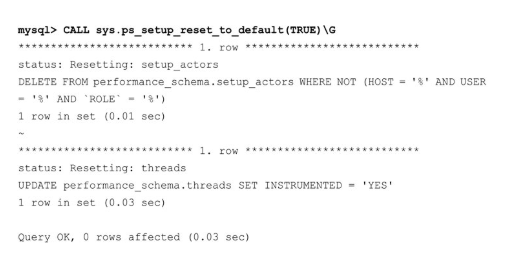
18,查看事务等待情况  
 mysql> select \* from innodb\_lock\_waits；

## 操作

从sys schema中启用一个计数器：



如果要重置为默认值，请执行以下操作：



sys schema中有许多表，本节展示了其中一些最常用的表。

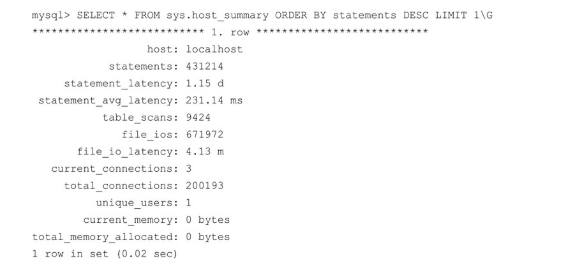
### 客户端/主机连接消耗资源

查看每个客户端IP过来的连接消耗了多少资源。  
 mysql> select \* from host\_summary;

### 当前连接情况

数据库连接来自哪里，以及这些连接对数据库的请求情况是怎样的？  
查看当前连接情况。  
 mysql> select host, current\_connections, statements from host\_summary;

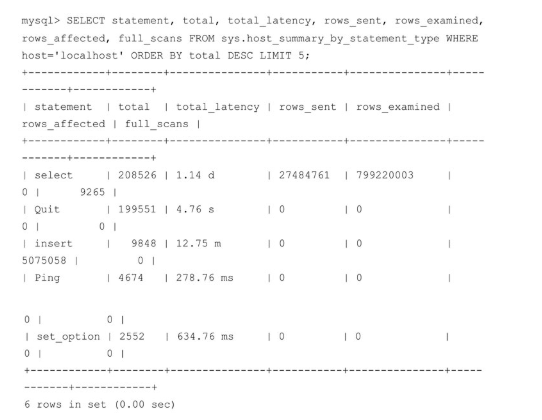
### 每个主机执行的语句



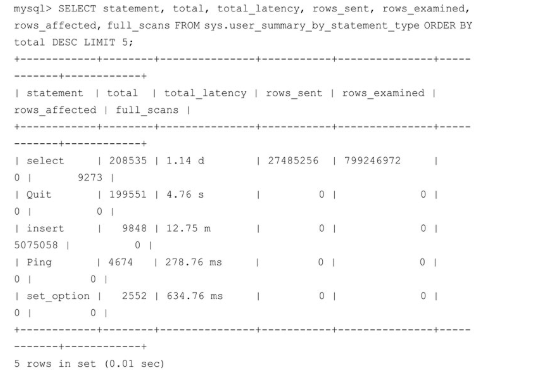
### 用户消耗资源

查看每个用户消耗了多少资源。  
 mysql> select \* from user\_summary;

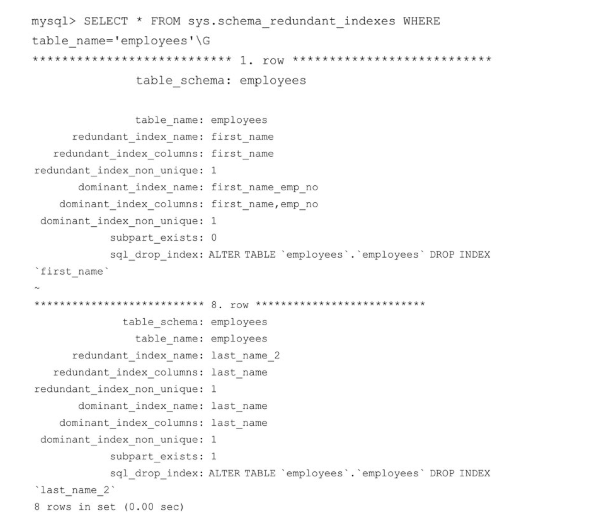
### 按类型列出每个主机的语句（INSERT和SELECT）



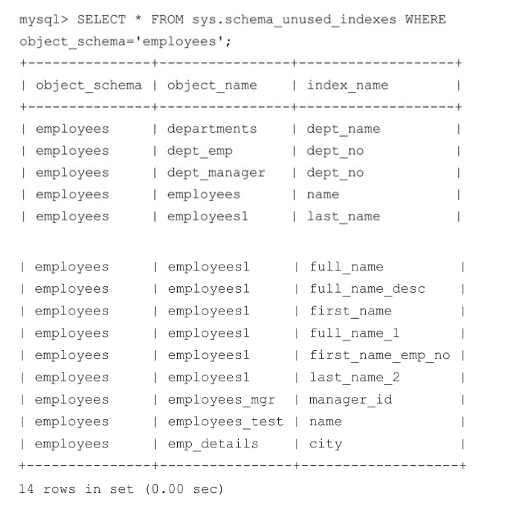
### 按类型列出每个用户的语句



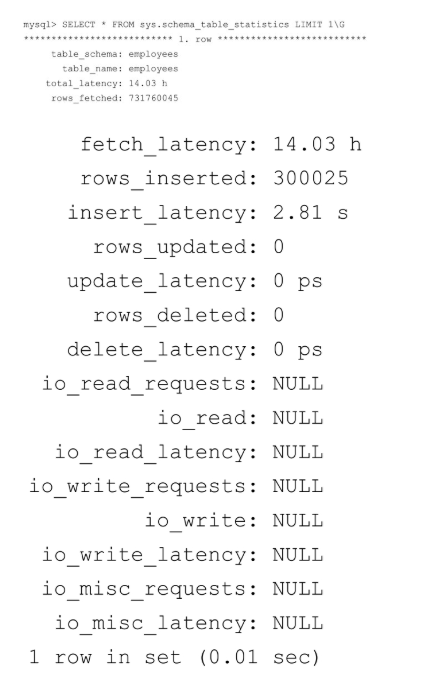
### 冗余索引

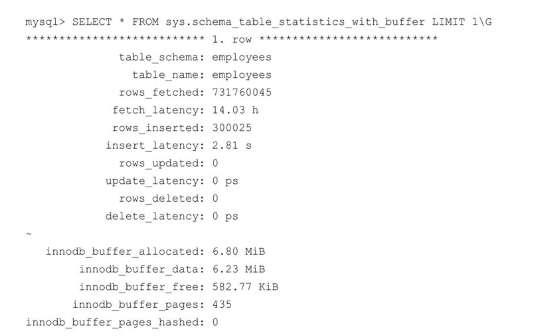


### 未使用的索引

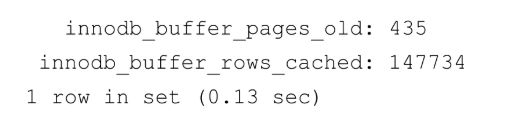


### 对表的统计





### 对带缓冲区（buffer）的表的统计



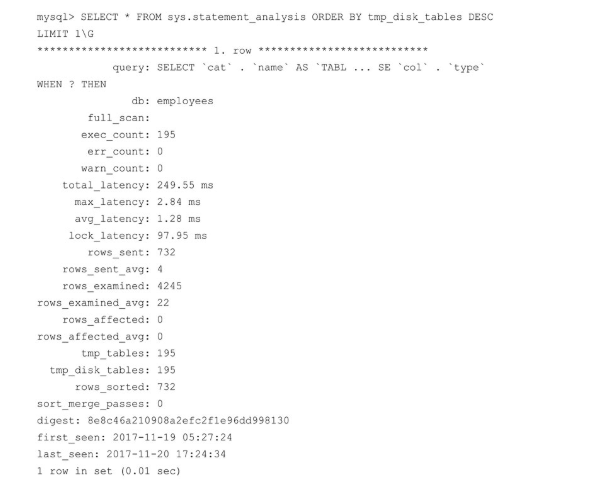
### 语句分析

此输出类似于performance\_schema.events\_statements\_summary\_by\_digest和pt-query-digest的输出。

根据查询的执行次数，排在前几位的查询如下：



消耗了最大的tmp\_disk\_tables的语句为：



要了解有关sys\_schema对象的更多信息，请参阅

https：//dev.mysql.com/doc/refman/8.0/en/sys-schema-object-index.html。参见http：//jorgenloland.blogspot.in/2012/04/improvements-for-many-table-join-in.html。

# mysql

## event

SELECT \* FROM mysql.event;

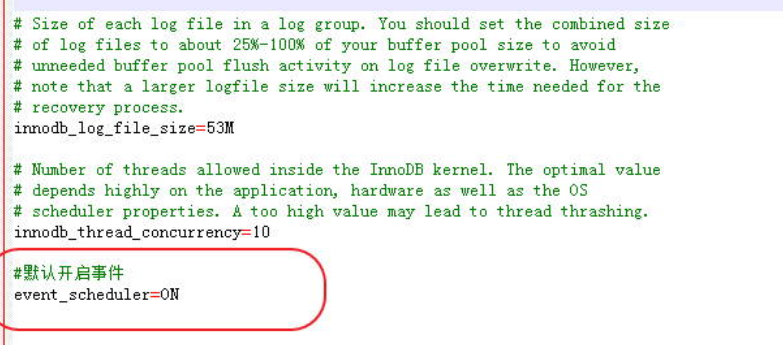
SET GLOBAL event\_scheduler = 1; -- 开启定时器 0：off 1：on

SHOW VARIABLES LIKE 'event\_scheduler';-- 查看是否开启定时器

如果显示OFF，则输入以下语句开启：

set global event\_scheduler = on;

提醒：虽然这里用set global event\_scheduler = on语句开启了事件，但是每次重启电脑或重启mysql服务后，会发现，事件自动关闭（event\_scheduler=OFF），所以想让事件一直保持开启，最好修改配置文件，让mysql服务启动的时候开启时间，只需要在my.ini配置文件的[mysqld]部分加上event\_scheduler=ON 即可，如下：



MySQL从5.1开始支持event功能，类似oracle的job功能。有了这个功能之后我们就可以让MySQL自动的执行数据汇总等功能，不用像以前需要操作的支持了。如linux crontab功能 。

**查看是否开启定时器**

SHOW VARIABLES LIKE '%sche%';

+-----------------+-------+

| Variable\_name | Value |

+-----------------+-------+

| event\_scheduler | ON |

+-----------------+-------+

1 row in set

开启定时器 0：off 1：on

SET GLOBAL event\_scheduler = 1;

**创建事件**

每隔一秒自动调用e\_test()存储过程

CREATE EVENT IF NOT EXISTS event\_test

ON SCHEDULE EVERY 1 SECOND

ON COMPLETION PRESERVE

DO CALL e\_test();

**开启事件**

ALTER EVENT event\_test ON

COMPLETION PRESERVE ENABLE;

**关闭事件**

ALTER EVENT event\_test ON

COMPLETION PRESERVE DISABLE;

**参考：**

<https://www.cnblogs.com/gaogaoxingxing/p/9909970.html>

## servers

相关语法：CREATE SERVER、DELETE SERVER、ALTER SERVER

## tables

## columns