目录

[第10章　information\_schema初相识 1](#_Toc1715)

[10.1　什么是information\_schema 1](#_Toc22490)

[10.2 information\_schema组成对象 2](#_Toc23739)

[第11章　information\_schema应用示例荟萃 4](#_Toc10951)

[11.1 使用Server层的字典表查询相关的元数据信息 5](#_Toc6535)

[11.1.1　查看数据库中是否使用了外键 5](#_Toc13775)

[11.1.2　查看InnoDB表空间文件信息 6](#_Toc3536)

[11.1.3　查看索引的统计信息 8](#_Toc3051)

[11.1.4　查看表的字段相关信息 8](#_Toc6450)

[11.1.6　查看数据库中的分区表信息 9](#_Toc19656)

[11.1.8　查看数据库中的计划任务 10](#_Toc2356)

[11.2 使用InnoDB层的字典表查询相关的元数据信息 11](#_Toc21016)

[11.2.1　查看索引列的信息 11](#_Toc12770)

[11.2.2　查看表的字段相关信息 11](#_Toc1105)

## 第10章　information\_schema初相识

## 10.1　什么是information\_schema

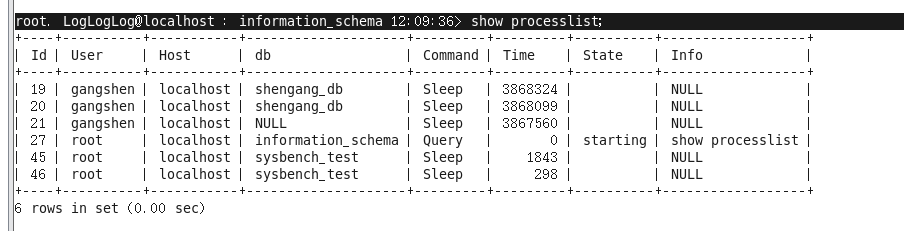
知识要点：

1. information\_schema是MYSQL的数据字典(关于数据库本身的数据)。

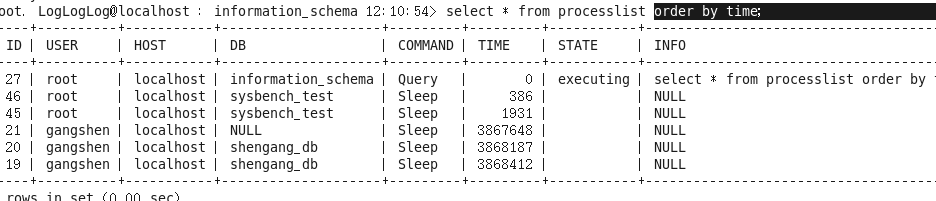
2. information\_schema的所有表是只读的。

3. 查询information\_schema的表可以规范且灵活地替代show语句。

例如，我们可以通过show processlist查看用户正在运行的线程：



正如书中比较show语句和查询information\_schema的表时指出，“基于SQL语句的查询，对来自information\_schema中的查询结果可以进行过滤、排序、联结操作”，我们可以按照TIME字段(线程处于当前状态的时间)排序：



**不过，如果没有特别的要求，show语句自然就是我们的一个常用选择。书中说到：**

#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**）

# 语法

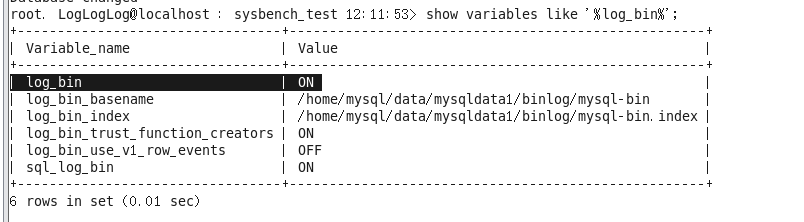
Syntax:

SHOW [GLOBAL | SESSION] VARIABLES

**[LIKE 'pattern' | WHERE expr]**

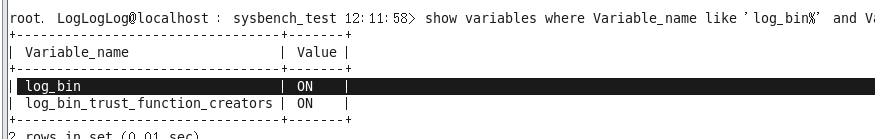
# 示例1

mysql> show variables like '%log\_bin%';



# 示例2

mysql> show variables where Variable\_name like 'log\_bin%' and Value='ON';



# 注意，like与where子句可单独使用，但要同时使用时，like子句必须在where之后

注：以上语句用于查询确认二进制日志(binary log)的状态。log\_bin记录INSERT、UPDATE、DELETE等操作(不管成功与否)，打开log\_bin之后，就能用二进制日志进行数据恢复和备份了。

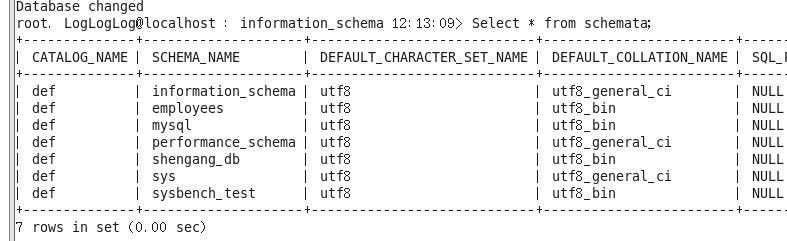
## 10.2 information\_schema组成对象

注：information\_schema下的表可以分为Server层和InnoDB层两类。

1. Server层的统计信息字典表：COLUMNS、STATISTICS、FILES、SCHEMATA等。

例：列出数据库的两种方式

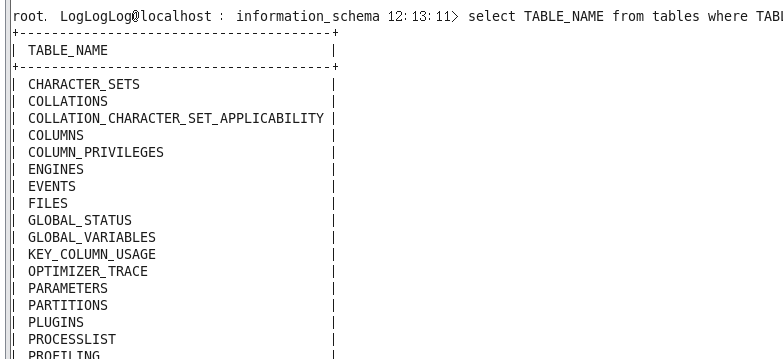
Select \* from schemata;



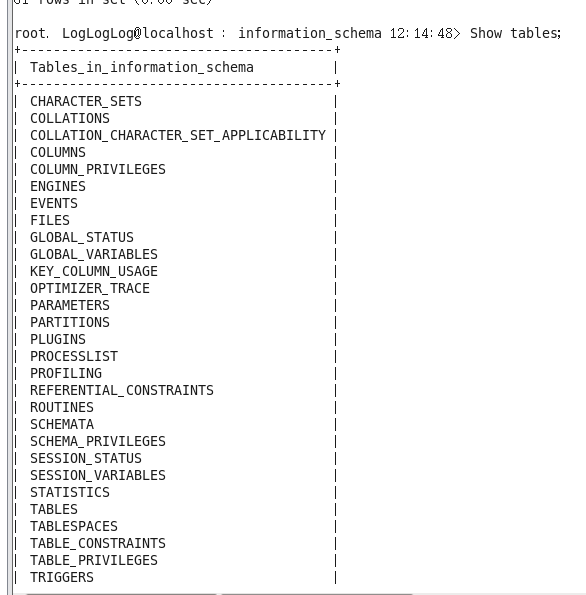
2. Server层的表级别对象字典表：VIEWS、TRIGGERS、TABLES、ROUTINES等。

**例：列出数据表的两种方式**

**select TABLE\_NAME from tables where TABLE\_SCHEMA='information\_schema';**

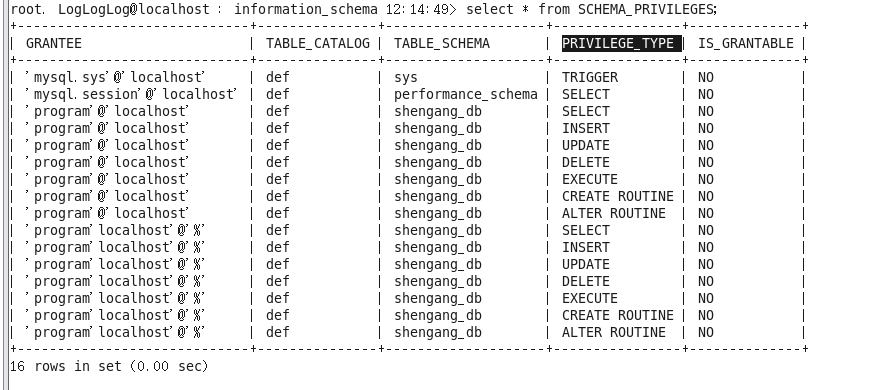


Show tables;



3. Server层的混杂信息字典表：GLOBAL\_STATUS、GLOBAL\_VARIABLES、SESSION\_STATUS、SESSION\_VARIABLES、**PROCESSLIST**等。

例：查询关于库级别的权限信息



注：InnoDB层方面暂略。

4. InnoDB层的系统字典表：INNODB\_SYS\_DATAFILES、INNODB\_SYS\_INDEXES、INNODB\_SYS\_TABLES、INNODB\_SYS\_TABLESTATS等。

5. InnoDB层的锁、事务、统计信息字典表：INNODB\_LOCKS、INNODB\_TRX、INNODB\_BUFFER\_PAGE\_LRU等。

6. InnoDB层的全文索引字典表：INNODB\_FT\_CONFIG、INNODB\_FT\_BEING\_DELETED、INNODB\_FT\_INDEX\_CACHE等。

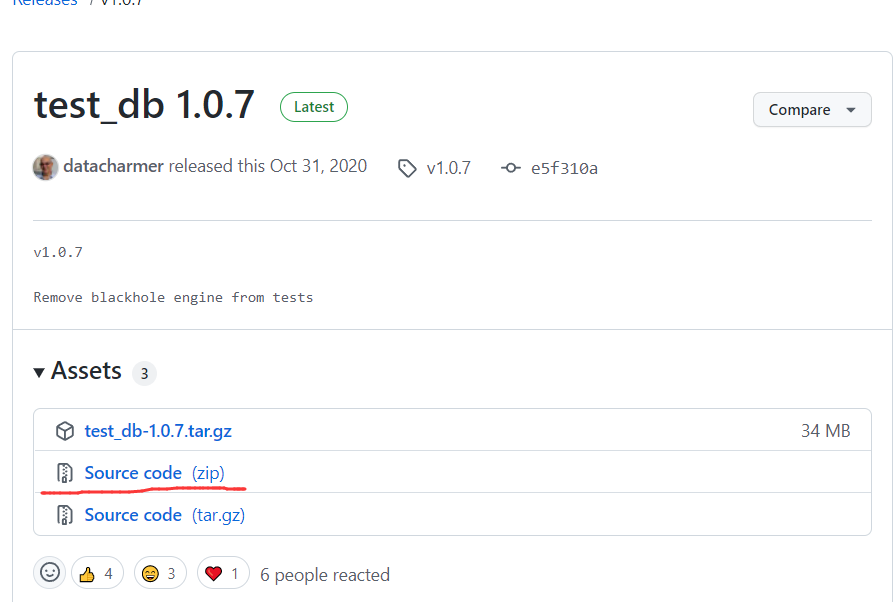
7. InnoDB层的压缩相关字典表：INNODB\_CMP和INNODB\_CMP\_RESET等。

## 第11章　information\_schema应用示例荟萃

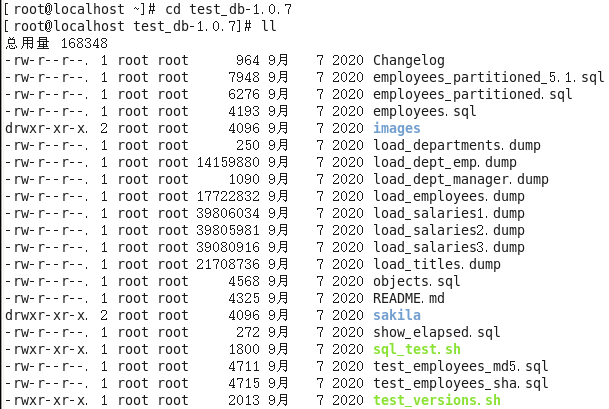
准备工作：本章需要安装mysql自带的员工数据库(employees)，步骤如下：

1. 获取数据压缩包，网址：

<https://github.com/datacharmer/test_db/releases/tag/v1.0.7>

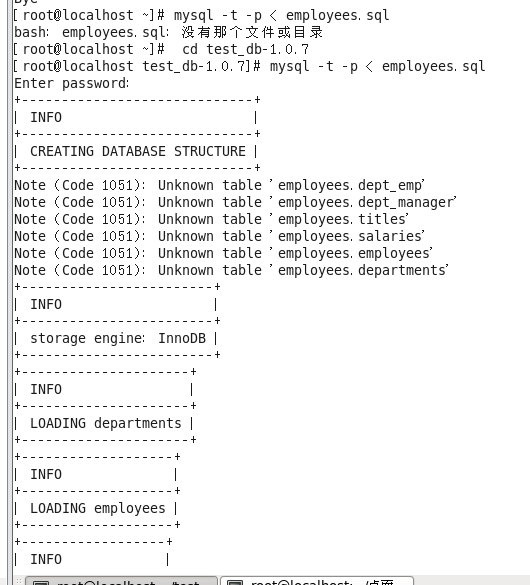


2. 桌面右键选取“解压缩到此处”解压缩，然后进入安装目录

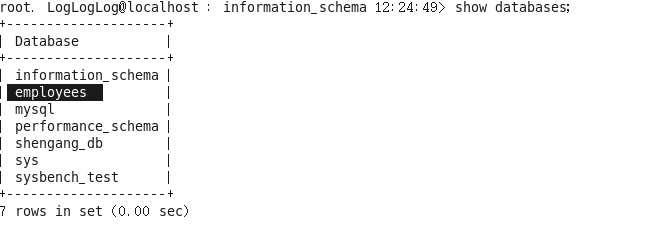


开始安装(部分安装结果显示略，安装时间较长)：

**[root@localhost test\_db-1.0.7]# mysql -t -p < employees.sql**



3. 安装后可以见到员工数据库：



相关完整介绍见：

<https://dev.mysql.com/doc/employee/en/employees-installation.html>

**重要：information\_schema之于性能优化有何用处？许多性能问题也许是因为数据库设计本身不合理所造成的，因此排查问题时必须考虑数据库设计是否不合理。其中，较容易出现设计不合理的出现在外键、字段设计等地方。当然，像索引、分表等设计实际是为优化性能而做的设计，但如果设计不合理也会产生性能问题。总之，排查性能问题离不开对数据库设计的综合考察，而相关信息正来自于information\_schema。**

## 11.1 使用Server层的字典表查询相关的元数据信息

## 11.1.1　查看数据库中是否使用了外键

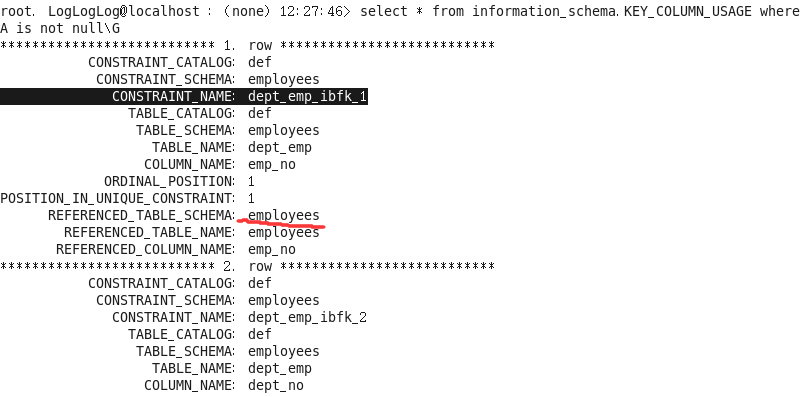
注：参考文章，数据库不推荐使用外键的 9 个理由，https://zhuanlan.zhihu.com/p/61457136

“在表上拥有活动的外键可以提高数据质量，但**会影响插入、更新和删除操作的性能**。在这些任务之前，数据库需要检查它是否违反数据完整性。”

#查询employees库是否存在外键

#通常在开发规范中**禁止使用外键**，如果是因为审核不规范导致使用了外键，那么就找开发人员沟通是否可以**使用程序逻辑来做一些数据上的约束**(代替外键)

mysql> select \* from information\_schema.**KEY\_COLUMN\_USAGE** where CONSTRAINT\_SCHEMA='employees' and REFERENCED\_TABLE\_SCHEMA is not null\G

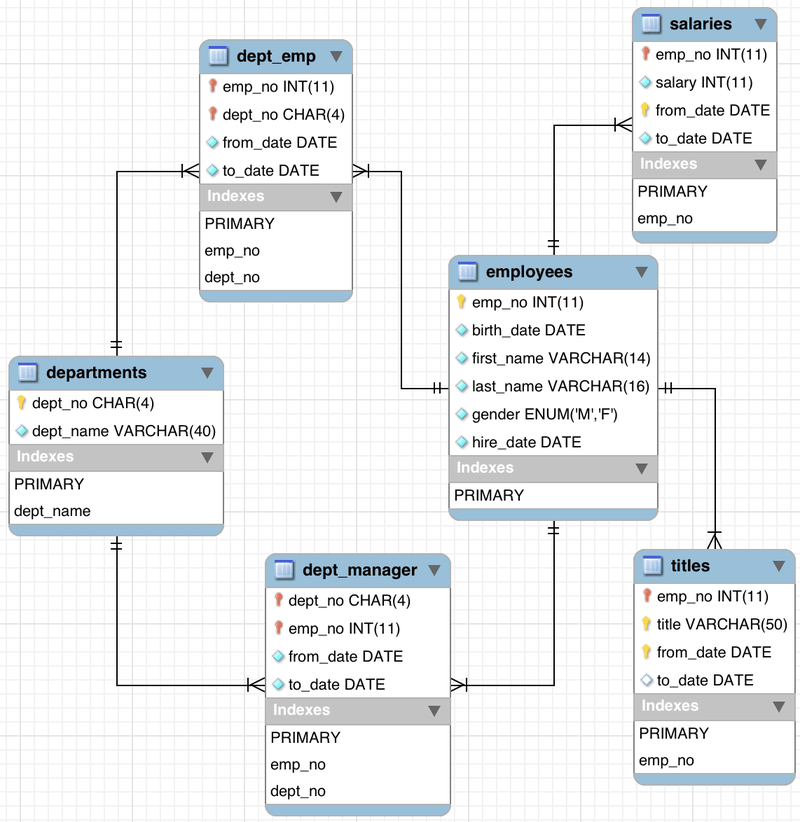


注(自行注解)：根据以上第1行返回结果可知，

在表dept\_emp中，

Employees(dept\_emp\_ibfk\_2)为外键

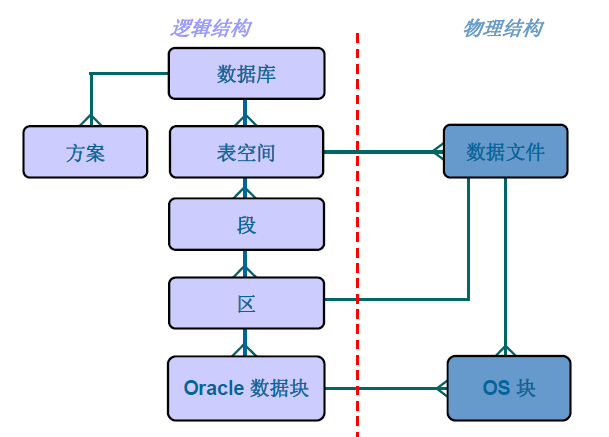
(实际是表dept\_emp(dept\_no)的主键)。



图片出处: https://www.cnblogs.com/hanshuai/p/14362639.html

## 11.1.2　查看InnoDB表空间文件信息

注：表空间是在数据库中开辟的一个空间，用于存放数据库的对象，一个数据库可以由多个表空间组成。

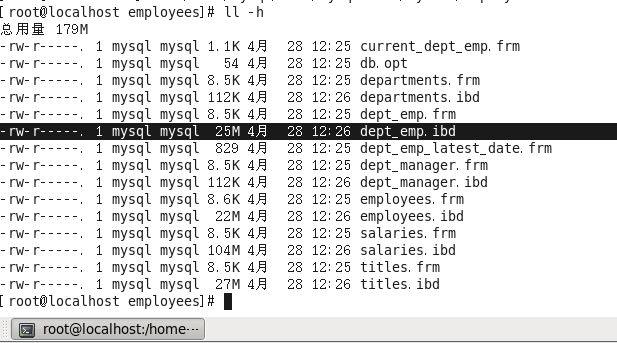


以上内容来源：<https://blog.csdn.net/jmilk/article/details/51599260>

#查询employees库下的InnoDB表dept\_emp的**表空间使用情况**

**注：表dept\_emp对应于文件dept\_emp.ibd**

**cd /home/mysql/data/mysqldata1/mydata/employees**

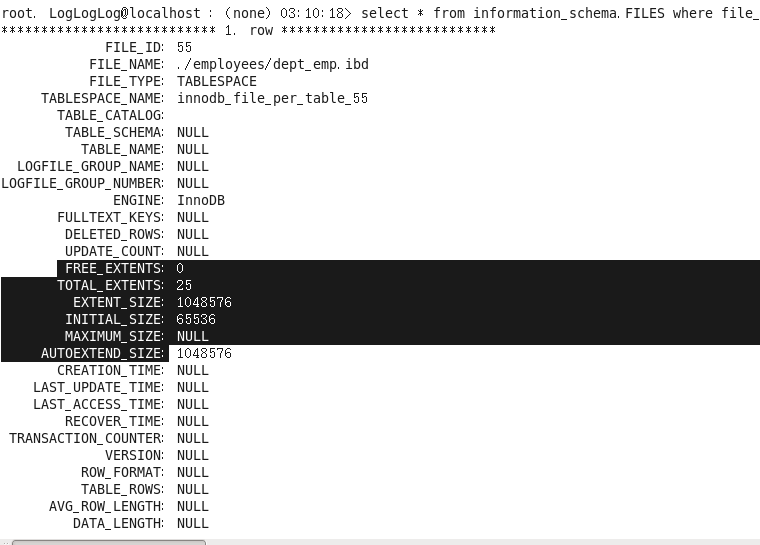


注： .ibd文件在mysql体系中的位置和作用：

图示

描述已自动生成

mysql> select \* from information\_schema**.FILES** where file\_name='./employees/**dept\_emp.ibd'**\G



**#**EXTENT\_SIZE：表示**数据文件的区块大小**。对于4KB、8KB或16KB页面大小的数据文件，区块大小是1048576字节（1MB）；MAXIMUM\_SIZE：表示数据文件所允许的最大字节数。DATA\_FREE：表示整个表空间的可用空间总量（以字节为单位）。

注：以上结果涉及一个术语“extent”，它被翻译为“区”。可参考如下内容(<https://zhuanlan.zhihu.com/p/48446222>)：

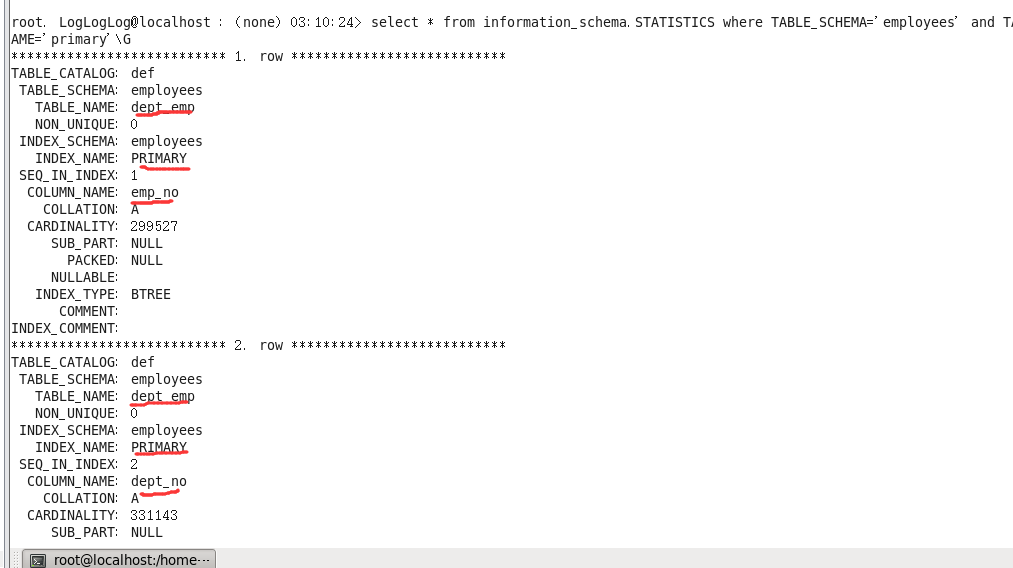
“Extent（区间/扩展），分配给对象（如表）的任何连续块叫区间；区间也叫扩展，因为当它用完已经分配的区间后，再有新的记录插入就必须再分配新的区间（即扩展一些块）；一旦区间分配给某个对象（表、索引及簇），则该区间就不能再分配给其它的对象.”

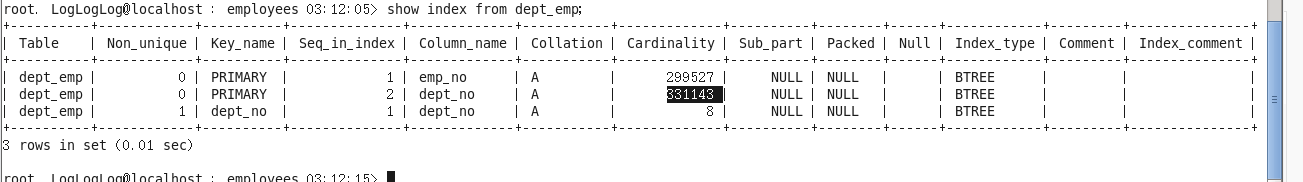
## 11.1.3　查看索引的统计信息

#STATISTICS表提供查询关于索引的**统计**信息

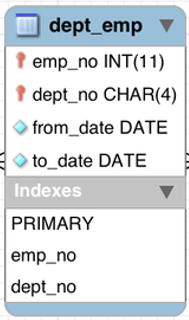
#查询employees库下的InnoDB表dept\_emp的**主键索引**统计信息

mysql> select \* from information\_schema.**STATISTICS** where TABLE\_SCHEMA='employees' and TABLE\_NAME='dept\_emp' and INDEX\_NAME=**'primary'**\G



注：实际平时我们用的就是对应的show语句：  


注：以上结果各字段信息解释参考教材。结合以下表设计理解上述结果。dept\_emp表设计如下：

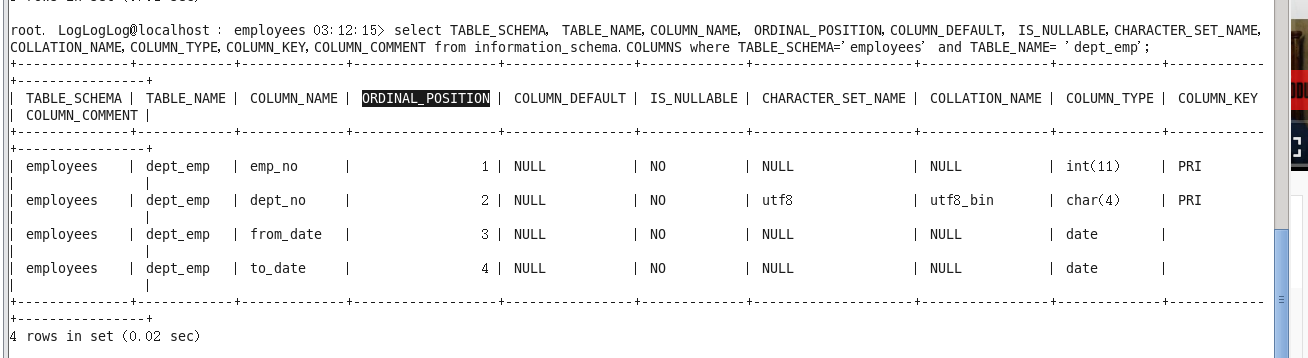


## 11.1.4　查看表的字段相关信息

# COLUMNS表提供查询表对象中的**字段**信息。

# 查询employees库下的InnoDB表dept\_emp的字段名称

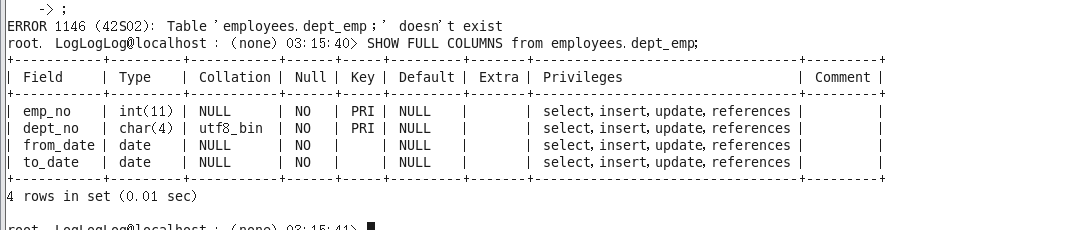
mysql> select TABLE\_SCHEMA, TABLE\_NAME,COLUMN\_NAME, ORDINAL\_POSITION,COLUMN\_DEFAULT, IS\_NULLABLE,CHARACTER\_SET\_NAME,COLLATION\_NAME,COLUMN\_TYPE,COLUMN\_KEY,COLUMN\_COMMENT from information\_schema.**COLUMNS** where TABLE\_SCHEMA='employees' and TABLE\_NAME= 'dept\_emp';



#ORDINAL\_POSITION：显示字段在表中的创建顺序。

注：show语句返回结果的信息更少：

SHOW FULL COLUMNS from employees.dept\_emp;



## 11.1.6　查看数据库中的分区表信息

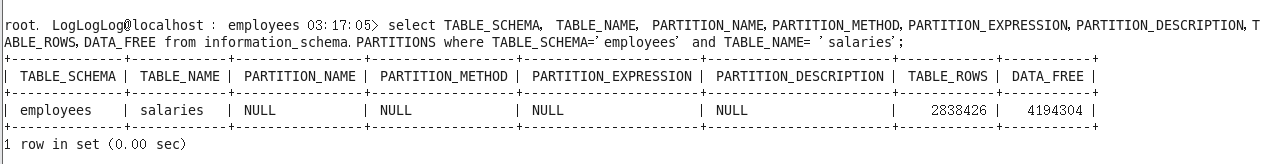
注：简单而言，对表按照某种依据划分为多个“小表”可提高性能，例如按日期时间段划分数据。详细介绍参见：

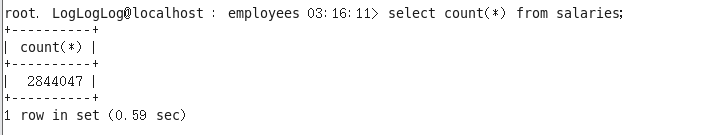
<https://blog.csdn.net/u013490280/article/details/111033099>

# PARTITIONS表提供查询关于分区表的信息。

#查询employees库下的salaries表的分区信息

mysql> select TABLE\_SCHEMA, TABLE\_NAME, PARTITION\_NAME,PARTITION\_METHOD,PARTITION\_EXPRESSION,PARTITION\_DESCRIPTION,TABLE\_ROWS,DATA\_FREE from information\_schema.**PARTITIONS** where TABLE\_SCHEMA='employees' and TABLE\_NAME= 'salaries';





注：这里放松了查询的限制条件，并不要求PARTITION\_NAME字段不为空。

## 11.1.8　查看数据库中的计划任务

注： MySQL5.1.x版本中引入了一项新特性EVENT，顾名思义就是事件、定时任务机制，在指定的时间单元内执行特定的任务，因此今后一些对数据定时性操作不再依赖外部程序，而直接使用数据库本身提供的功能。

以上参考：<https://cloud.tencent.com/developer/article/1053177>

# EVENTS表提供查询与计划任务事件相关的信息

# 查询sbtest库下是否存在计划任务

mysql> select \* from information\_schema.**events** where EVENT\_SCHEMA='sbtest'\G

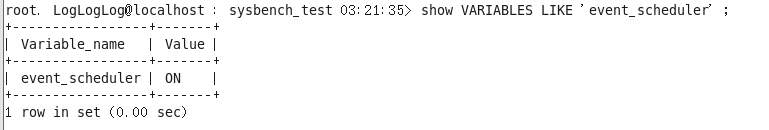
注：这里sysbench\_test本并没有计划任务，这里按照以上参考的资料给出另一个例子。

1. 开启计划任务调度

SET GLOBAL event\_scheduler = 1;

show VARIABLES LIKE 'event\_scheduler' ;

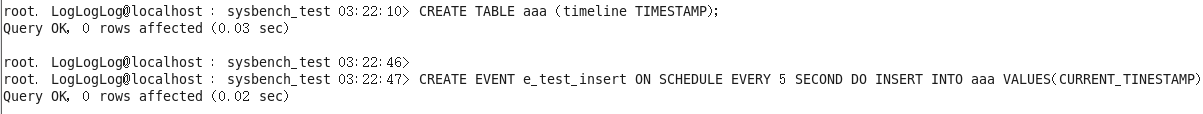


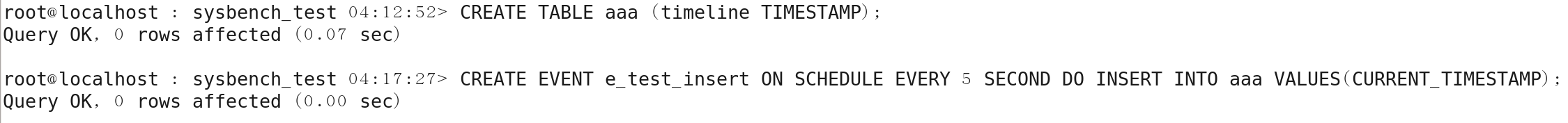


2. 在sysbench\_test中创建表aaa，然后创建一个任务，每5秒插入一行数据。

CREATE TABLE aaa (timeline TIMESTAMP);

CREATE EVENT e\_test\_insert ON SCHEDULE EVERY 5 SECOND DO INSERT INTO aaa VALUES(CURRENT\_TINESTAMP) ;





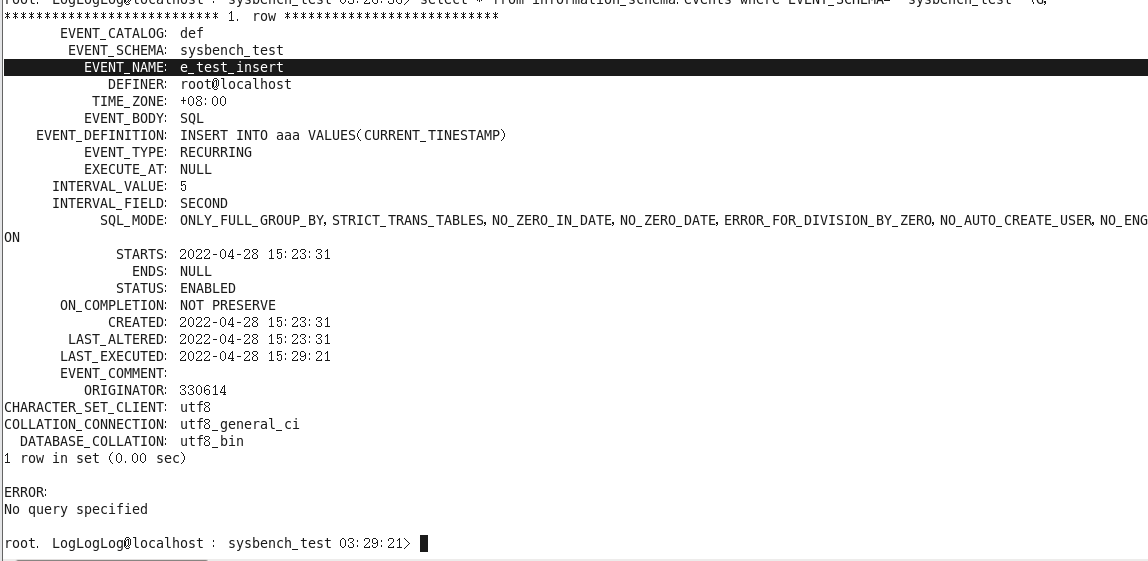
3. 任务生效

图形用户界面, 应用程序

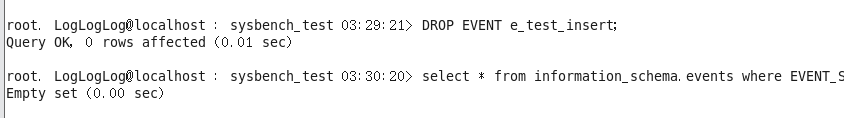
中度可信度描述已自动生成

4. 通过EVENTS表查询sysbench\_test库的计划任务，相关字段信息参见教材。

select \* from information\_schema.events where EVENT\_SCHEMA= 'sysbench\_test' \G;



5. 删除任务，这时EVENTS表也找不到该项任务记录了。



## 11.2 使用InnoDB层的字典表查询相关的元数据信息

## 11.2.1　查看索引列的信息

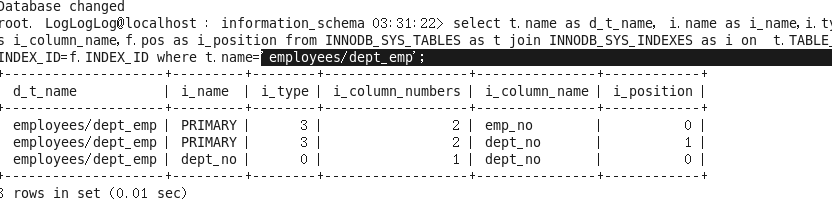
# INNODB\_SYS\_FIELDS表提供查询有关InnoDB索引列（字段）的元数据信息

# INNODB\_SYS\_INDEXES表提供查询有关InnoDB索引的元数据信息

# INNODB\_SYS\_TABLES表提供查询有关InnoDB表的元数据信息

# 查询employees库下的InnoDB表dept\_emp的索引列名称和索引列

mysql> select t.name as d\_t\_name, i.name as i\_name,i.type as i\_type,i.N\_FIELDS as i\_column\_numbers,f.name as i\_column\_name,f.pos as i\_position from **INNODB\_SYS\_TABLES** as t join **INNODB\_SYS\_INDEXES** as i on t.TABLE\_ID=i.TABLE\_ID left join **INNODB\_SYS\_FIELDS** as f on i.INDEX\_ID=f.INDEX\_ID where t.name='**employees/dept\_emp**';



(请根据实际解释)注：注意这里与书里结果有所不同。employees库下的dept\_emp表有两个索引，一个双列主键索引(emp\_no列和dept\_no列构成的索引)和一个单列普通索引(就是dept\_no本身)。i\_name是索引名称，i\_type是索引类型。

#下面对查询结果做一个简单的解读。

● d\_t\_name（INNODB\_SYS\_TABLES.name）：表名称。

● i\_name（INNODB\_SYS\_INDEXES.name）：索引名称。

● i\_type（INNODB\_SYS\_INDEXES.type）：0 =二级索引、1 =集群索引、2 =唯一索引、3 =主键索引、32 =全文索引(其余略，见教材)。

● i\_column\_numbers：索引key中的列数量

● i\_column\_name（INNODB\_SYS\_FIELDS.name）：索引列的名称

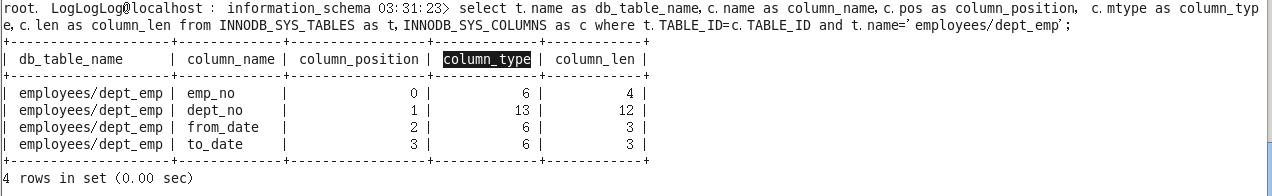
● i\_position（INNODB\_SYS\_FIELDS.pos）：索引列的序号位置，从0开始

## 11.2.2　查看表的字段相关信息

# INNODB\_SYS\_TABLES表提供查询有关InnoDB表的元数据

# INNODB\_SYS\_COLUMNS表提供查询有关InnoDB表字段的元数据信息

mysql> select t.name as db\_table\_name,c.name as column\_name,c.pos as column\_position, c.mtype as column\_type,c.len as column\_len from INNODB\_SYS\_TABLES as t,INNODB\_SYS\_COLUMNS as c where t.TABLE\_ID=c.TABLE\_ID and t.name='employees/dept\_emp';



#下面对查询结果做一个简单的解读

● db\_table\_name（INNODB\_SYS\_TABLES.name）：表名称

● column\_name（INNODB\_SYS\_COLUMNS.name）：字段名称

● column\_type（INNODB\_SYS\_COLUMNS.mtype）：表示字段类型的数字ID。1=VARCHAR、2 = CHAR、3 = FIXBINARY、4 = BINARY、5 = BLOB、6 = INT、7=SYS\_CHILD、8 = SYS、9 = FLOAT、10 = DOUBLE、11 = DECIMAL、12 =VARMYSQL、13 = MYSQL、14 = GEOMETRY。

● column\_len（INNODB\_SYS\_COLUMNS.len）：字段字节长度