# MySQL日志

**MySQL中的日志包括**：错误日志、二进制日志、通用查询日志、慢查询日志等等。这里主要介绍下比较常用的两个功能：通用查询日志和慢查询日志。

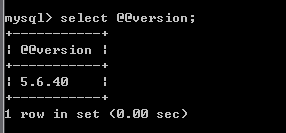
**通用查询日志：**记录建立的客户端连接和执行的语句。

**慢查询日志：**记录所有执行时间超过long\_query\_time秒的所有查询或者不使用索引的查询。

## 通用查询日志

查询当前数据库中与版本号相关的东西的一些命令。

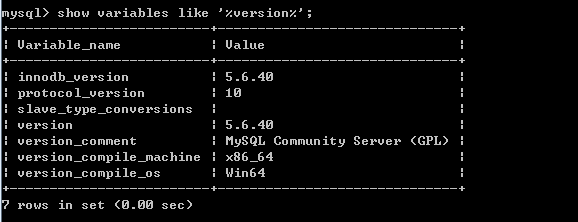
1. 查看当前数据库的版本



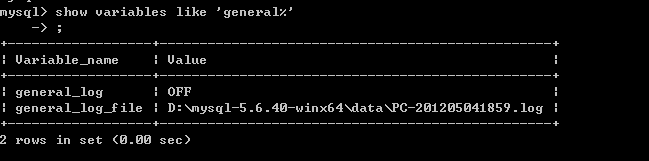
2.查看数据库的状态



3.查看与版本号有关的

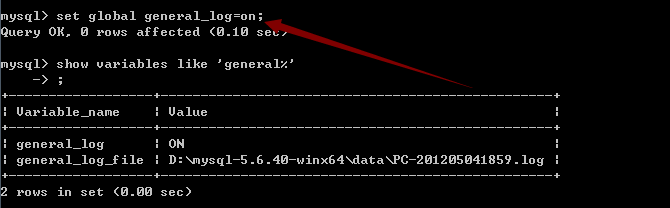


4.查看通用日志是否开启



如果general\_log的值为ON则为开启，为OFF则为关闭（**默认情况下是关闭的**）。

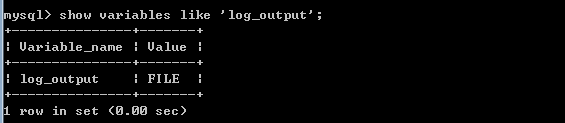
5.开启通用日志



## 慢查询日志

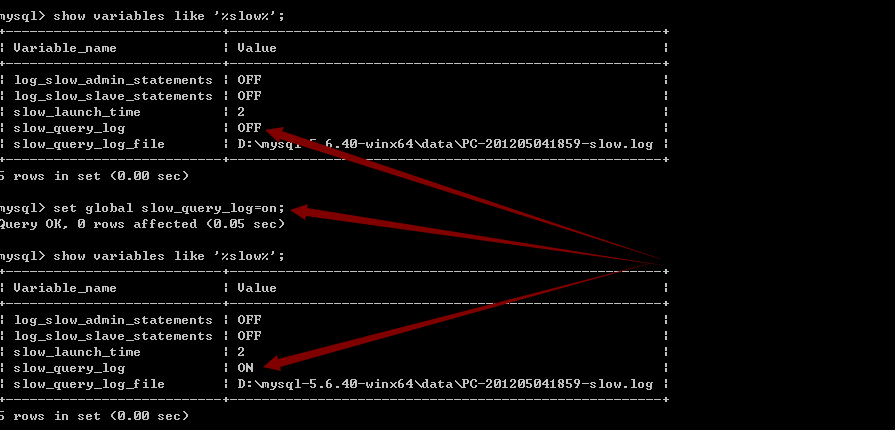
MySQL的慢查询日志是MySQL提供的一种日志记录，用来记录在MySQL中响应时间超过阈值的语句，具体指运行时间超过long\_query\_time值的SQL，则会被记录到慢查询日志中（日志可以写入文件或者数据库表，如果对性能要求高的话，建议写文件）。默认情况下，MySQL数据库是不开启慢查询日志的，long\_query\_time的默认值为10（即10秒，通常设置为1秒），即运行10秒以上的语句是慢查询语句。

1. 查看慢查询输出格式



可以是FILE（存储在数数据库的数据文件中的hostname.log），也可以是TABLE（存储在数据库中的mysql.general\_log）。

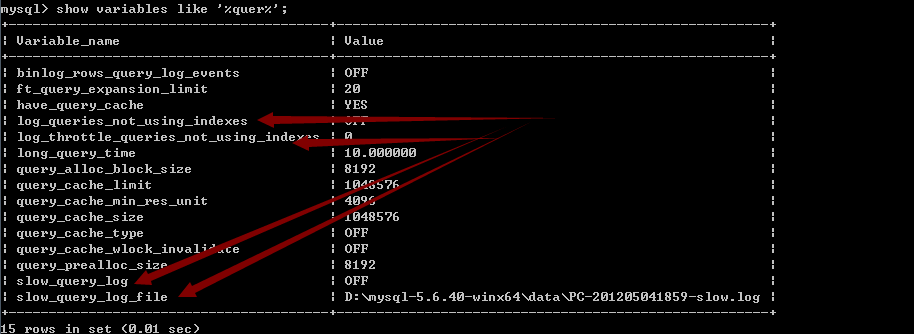
1. 开启慢查询。
2. 在线开启，如果MySQL发生重启，就会失效，如果要永久生效，就必选修改配置文件。



1. 配置文件开启。

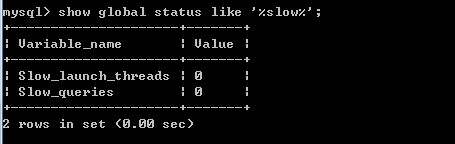
在配置文件[mysqld]中添加slow\_query\_log = ON和long\_query\_time = 1，然后重启MySQL即可生效。

3.查看慢查询相关参数。



主要参数：

1. **slow\_query\_log**的值为ON为开启慢查询日志，OFF则为关闭慢查询日志。
2. **slow\_query\_log\_file** 的值是记录的慢查询日志到文件中（**注意：**默认名为主名.log，慢查询日志是否写入指定文件中，需要指定慢查询的输出日志格式为文件，相关命令为：show variables like ‘%log\_output%’；去查看输出的格式）。
3. **long\_query\_time** 指定了慢查询的阈值，即如果执行语句的时间超过该阈值则为慢查询语句，默认值为10秒。
4. log\_queries\_not\_using\_indexes 如果值设置为ON，则会记录所有没有利用索引的查询（注意：如果只是将log\_queries\_not\_using\_indexes设置为ON，而将slow\_query\_log设置为OFF，此时该设置也不会生效，即该设置生效的前提是slow\_query\_log的值设置为ON），一般在性能调优的时候会暂时开启。
5. 查看当前慢查询个数。



## 日志分析工具

1. mysqlsla是hackmysql.com推出的一款MySQL的日志分析工具，功能非常强大. 数据报表,非常有利于分析慢查询的原因, 包括执行频率, 数据量, 查询消耗等。

2. MySQL自带的慢查询日志分析工具mysqldumpslow.pl分析日志

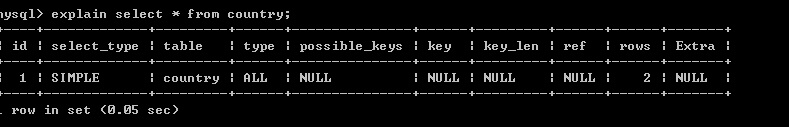
具体参数：

-s： 表示按何种方式排序，c、t、l、r分别是按照记录次数、时间、查询时间、返回的记录数来排序，ac、at、al、ar，表示相应的倒叙；

-t： 表示top的意思，后面跟着的数据表示返回前面多少条；

-g：后面可以写正则表达式匹配，大小写不敏感。

# EXPLAIN详解



expain出来的信息有10列，分别是:id、select\_type、table、type、possible\_keys、key、key\_len、ref、rows、Extra。

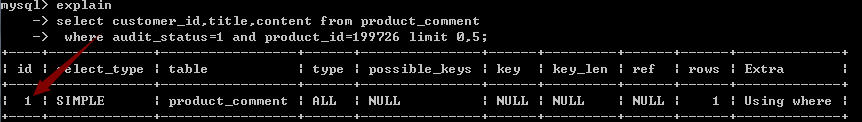
## id

1. id相同时，执行顺序由上至下。

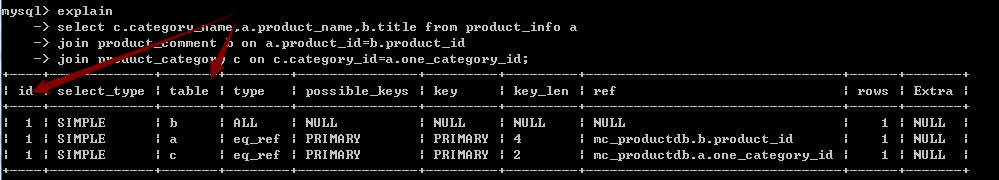
2.如果是子查询，id的序号会递增，id值越大优先级越高，越先被执行。

3.id如果相同，可以认为是一组，从上往下顺序执行,在所有组中，id值越大，优先级越高，越先执行。

单表查询：

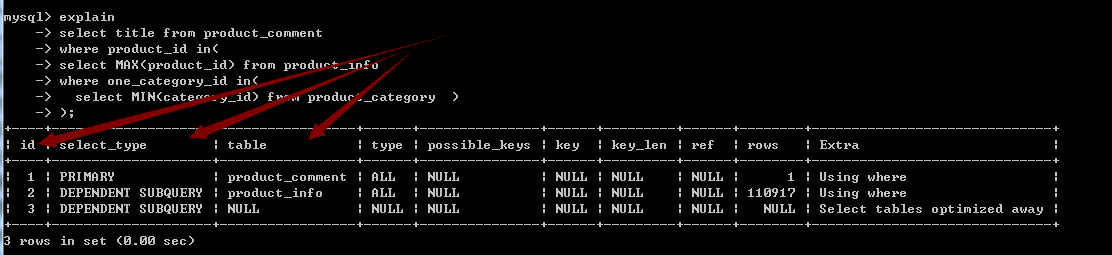


多表查询：

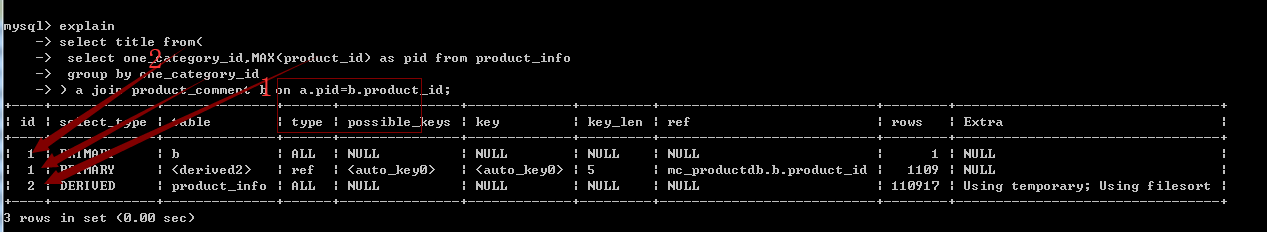


Id值相等，所以从上往下一次执行。

子查询：



Id值大的先执行。



在id值大的执行后，相等的id从上到下执行。

## Select\_type

1. SIMPLE(简单SELECT,不使用UNION或子查询等)



1. PRIMARY(查询中若包含任何复杂的子部分,最外层的select被标记为PRIMARY)

(3) UNION(UNION中的第二个或后面的SELECT语句)

(4) DEPENDENT UNION(UNION中的第二个或后面的SELECT语句，取决于外面的查询)

(5) UNION RESULT(UNION的结果)

(6) SUBQUERY(子查询中的第一个SELECT)

(7) DEPENDENT SUBQUERY(子查询中的第一个SELECT，取决于外面的查询)

(8) DERIVED(派生表的SELECT, FROM子句的子查询)

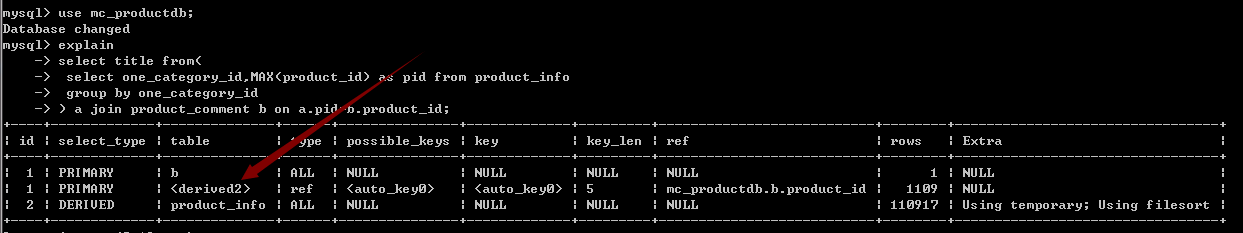
(9) UNCACHEABLE SUBQUERY(一个子查询的结果不能被缓存，必须重新评估外链接的第一行)

## Table

显示这一行的数据是关于哪张表的，有时不是真实的表名字,看到的是derivedx(x是个数字,我的理解是第几步执行的结果).

<unionM,N>由ID为M,N查询union产生的结果集。

<derivedN>/<subqueryN>由ID为N查询的结果集。



## Type

表示MySQL在表中找到所需行的方式，又称“访问类型”。

常用的类型有： ALL, index,  range, ref, eq\_ref, const, system, NULL（从左到右，性能从差到好）

ALL：Full Table Scan， MySQL将遍历全表以找到匹配的行

index: Full Index Scan，index与ALL区别为index类型只遍历索引树

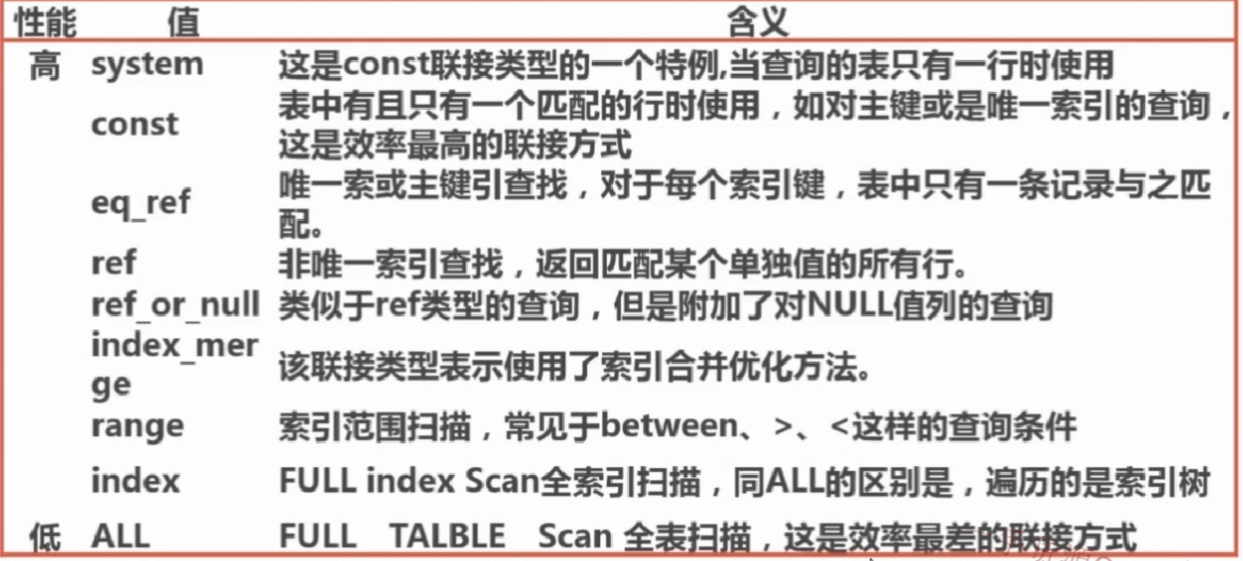
range:只检索给定范围的行，使用一个索引来选择行

ref: 表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

eq\_ref: 类似ref，区别就在使用的索引是唯一索引，对于每个索引键值，表中只有一条记录匹配，简单来说，就是多表连接中使用primary key或者 unique key作为关联条件

const、system: 当MySQL对查询某部分进行优化，并转换为一个常量时，使用这些类型访问。如将主键置于where列表中，MySQL就能将该查询转换为一个常量,system是const类型的特例，当查询的表只有一行的情况下，使用system

NULL: MySQL在优化过程中分解语句，执行时甚至不用访问表或索引，例如从一个索引列里选取最小值可以通过单独索引查找完成。

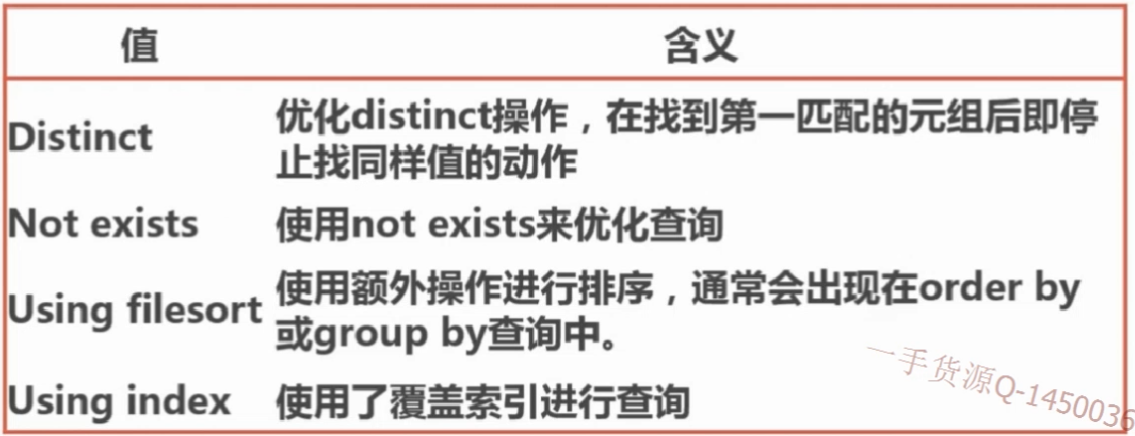


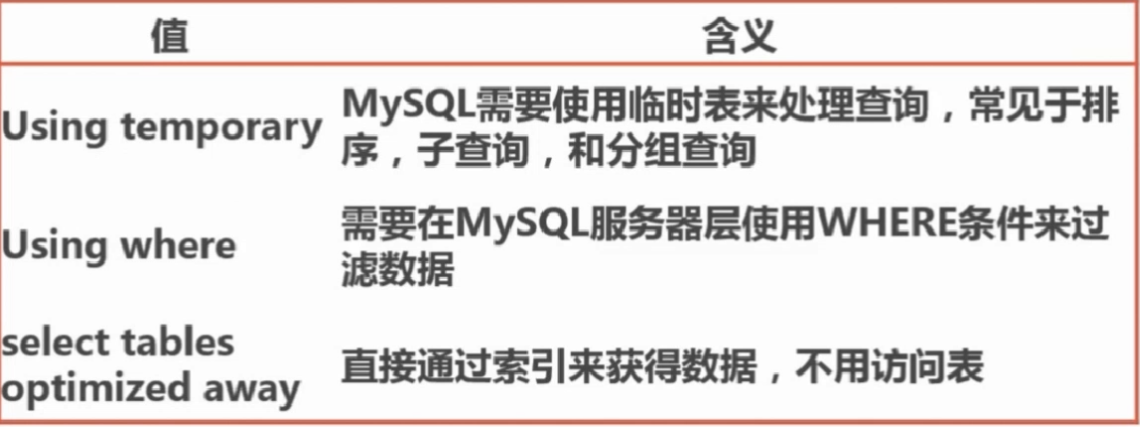
## Possible\_keys

指出MySQL能使用哪个索引在表中找到记录，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询使用

该列完全独立于EXPLAIN输出所示的表的次序。这意味着在possible\_keys中的某些键实际上不能按生成的表次序使用。  
如果该列是NULL，则没有相关的索引。在这种情况下，可以通过检查WHERE子句看是否它引用某些列或适合索引的列来提高你的查询性能。如果是这样，创造一个适当的索引并且再次用EXPLAIN检查查询

## Extra





## Key\_len

表示索引字段的最大长度。

Ken\_len的长度由字段定义计算而来，并非字段的实际长度。

# MySQL分区表

**分区：**

mysql数据库中的数据是以文件的形势存在磁盘上的，一张表主要对应着三个文件，一个扩展名为.frm的文件，存放表结构的，一个扩展名.myd的文件，存放表数据的，一个是扩展名为.myi的文件，存表索引的。如果一张表的数据量太大的话，那么myd,myi就会变的很大，查找数据就会变的很慢，这个时候我们可以利用mysql的分区功能，在物理上将这一张表对应的三个文件，分割成许多个小块，这样呢，我们查找一条数据时，就不用全部查找了，只要知道这条数据在哪一块，然后在那一块找就行了。

如果表的数据太大，可能一个磁盘放不下，这个时候，我们可以把数据分配到不同的磁盘里面去

**分区的好处**：

1. 与单个磁盘或文件系统分区相比，可以存储更多的数据。
2. 一些查询可以得到极大的[优化](http://www.xprogrammer.com/1197.html)，这主要是借助于满足一个给定WHERE语句的数据可以只保存在一个或多个分区内，这样在查找时就不用查找其他剩余的分区。因为分区可以在创建了分区表后进行修改，所以在第一次配置分区方案时还不曾这么做时，可以重新组织数据，来提高那些常用查询的效率。
3. 涉及到例如SUM()和COUNT()这样聚合函数的查询，可以很容易地进行并行处理，查询可以在每个分区上同时进行，最后总计所有分区的结构。
4. 通过跨多个磁盘来分散数据查询，来获得更大的查询吞吐量。

**分区的一些限制：**

（1）一个表最多只能有1024个分区（mysql5.6之后支持8192个分区）

（2）在mysql5.1中分区表达式必须是整数，或者是返回整数的表达式，在5.5之后，某些场景可以直接使用字符串列和日期类型列来进行分区（使用varchar字符串类型列时，一般还是字符串的日期作为分区）。

（3）如果分区字段中有主键或者唯一索引列，那么所有主键列和唯一索引列都必须包含进来，如果表中有主键或唯一索引，那么分区键必须是主键或唯一索引

（4）分区表中无法使用外键约束

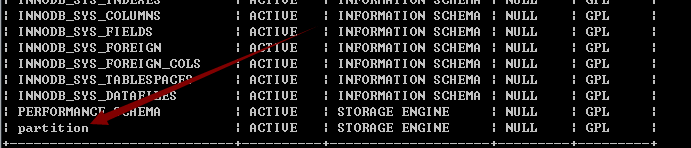
（5）mysql数据库支持的分区类型为水平分区，并不支持垂直分区，因此，mysql数据库的分区中索引是局部分区索引，一个分区中既存放了数据又存放了索引，而全局分区是指的数据库放在各个分区中，但是所有的数据的索引放在另外一个对象中

（6）目前mysql不支持空间类型和临时表类型进行分区，持全文索引。

**注意事项：**

* 1. 结合业务场景选择分区键，避免跨分区查询
  2. 对分区键进行查询最好在WHERE从句中包含分区键。
  3. 具有主键或唯一索引的表，主键或唯一索引必须是分区键的一部分。

查看mysql服务器是否支持分区：show plugins;



## 分区的类型

（1）**RANGE分区**：基于属于一个给定连续区间的列值，把多行分配给分区。

（2） **LIST分区：**类似于按RANGE分区，区别在于LIST分区是基于列值匹配一个离值集合中的某个值来进行选择。LIST分区通过使用“PARTITION BY LIST(expr)”来实现，其中“expr”是某列值或一个基于某个列值、并返回一个整数值的表达式，然后通过“VALUES IN (value\_list)”的方式来定义每个分区，其中“value\_list”是一个通过逗号分隔的整数列表。 注释MySQL5.1中，当使用LIST分区时，有可能只能匹配整数列表。

**（3） HASH分区**：基于用户定义的表达式的返回值来进行选择的分区，该表达式使用将要插入到表中的这些行的列值进行计算。这个函数可以包含MySQL 中有效的、产生非负整数值的任何表达式。

**（4） KEY分区**：类似于按HASH分区，区别在于KEY分区只支持计算一列或多列，且MySQL服务器提供其自身的哈希函数。必须有一列或多列包含整数值。

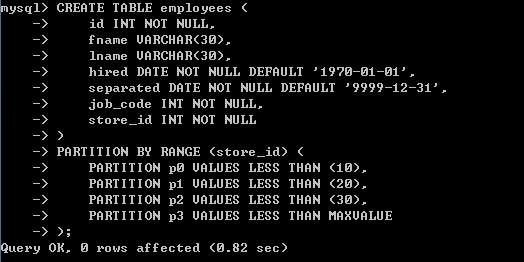
**（5）LINER HASH分区：**MySQL还支持线性哈希功能，它与常规哈希的区别在于，线性哈希功能使用的一个线性的2的幂（powers-of-two）运算法则，而常规哈希使用的是求哈希函数值的模数。线性哈希分区和常规哈希分区在语法上的唯一区别在于，在“PARTITION BY”子句中添加“LINEAR”关键字。

### RANGE分区

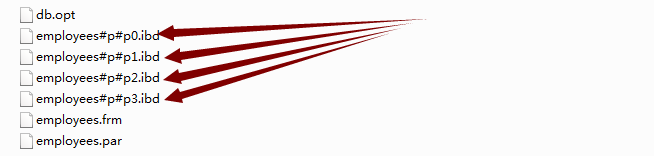
特点：根据分区键值的范围把数据行存储到表的不同分区中。

多个分区的范围要连续，但是不能重叠

默认情况下使用VALUES LESS THAN属性，即每个分区不包括指定的那个值。



MAXVALUE 表示最大的可能的整数值。



日期作为分区键：RANG分区针对日期字段进行分区可以使用时间类型的函数进行转换成整形，针对TIMESTAMP的日期类型的字段需要使用专门的UNIX\_TIMESTAMP（）函数进行转换。

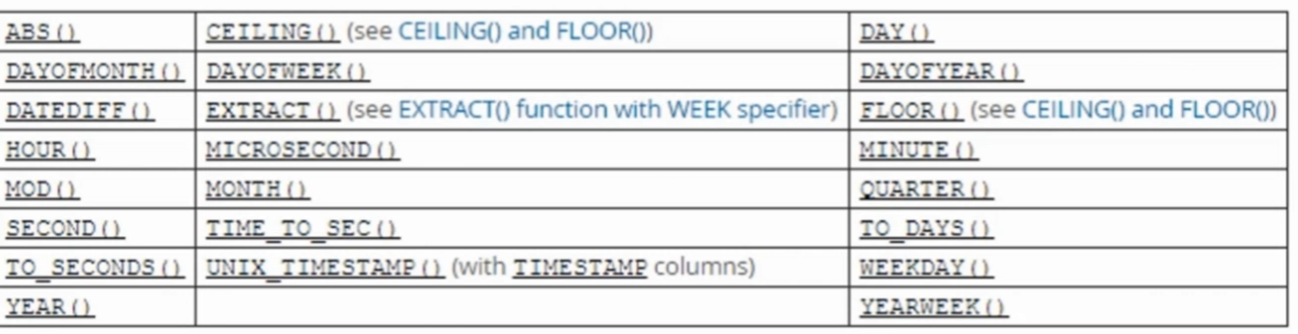
使用场景：分区键为日期或时间类型

所有查询中都包括分区键

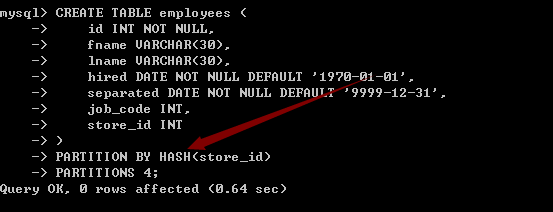
定期按分区范围清理历史记录。

### HASH分区

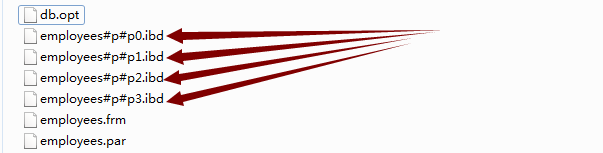
创建HASH分区时可能使用的函数：



创建HASH哈希：



如果没有包括一个PARTITIONS子句，那么分区的数量将默认为1。



特点：根据MOD(分区键，分区数)的值把数据行存储到表的不同分区中

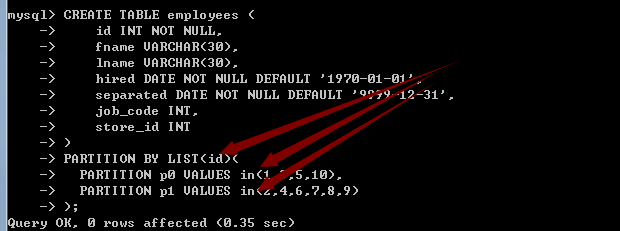
数据可以平均的分布在各个分区中。

HASH分区键值必须是一个INT类型的值，后者通过函数可以转化成INT类型。

### LIST分区

LIST分区的特点：

* 按分区键取值的列表进行分区。
* 同范围分区一样，各分区的列表值不能重复。
* 每一行数据必须能找到对应的分区列表，否则插入失败。



## 移除分区

语法：ALTER TABLE tablename REMOVE PARTITIONING ;



ALTER TABLE tablename drop PARTITION partitioname;

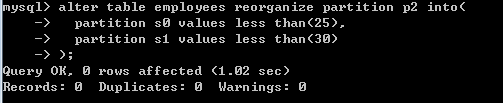


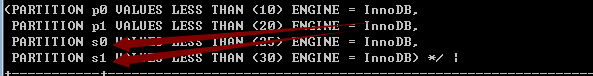
Ps: 使用remove移除分区是仅仅移除分区的定义，并不会删除数据和drop PARTITION不一样，后者会连同数据一起删除。

## 拆分合并分区

拆分合并分区统称为重新定义分区，拆分分为不会造成数据的丢失，只将会将数据从一个分区移动到另一个分区，合并分区将数据合并到一个分区中。

拆分分区：





合并分区：

