1、查看mysql版本： select version()

分组降序查询第一条数据

SELECT tagid, total\_count, update\_time FROM (

SELECT \* FROM tag\_usergroup\_push WHERE target\_system = '1000002071728486'

ORDER BY update\_time DESC)

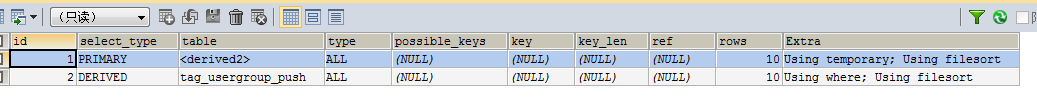
AS tdu GROUP BY tagid

explain或者describe查看sql执行计划

在mysql 版本5.7.16-log没有排序功能

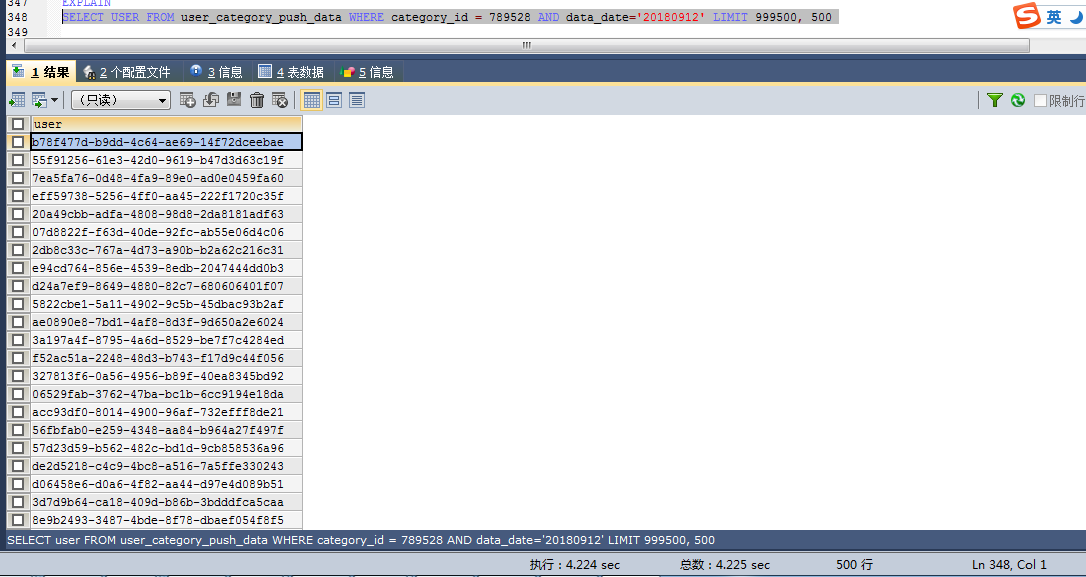


在mysql 版本5.6.39有排序功能，功能正常实现

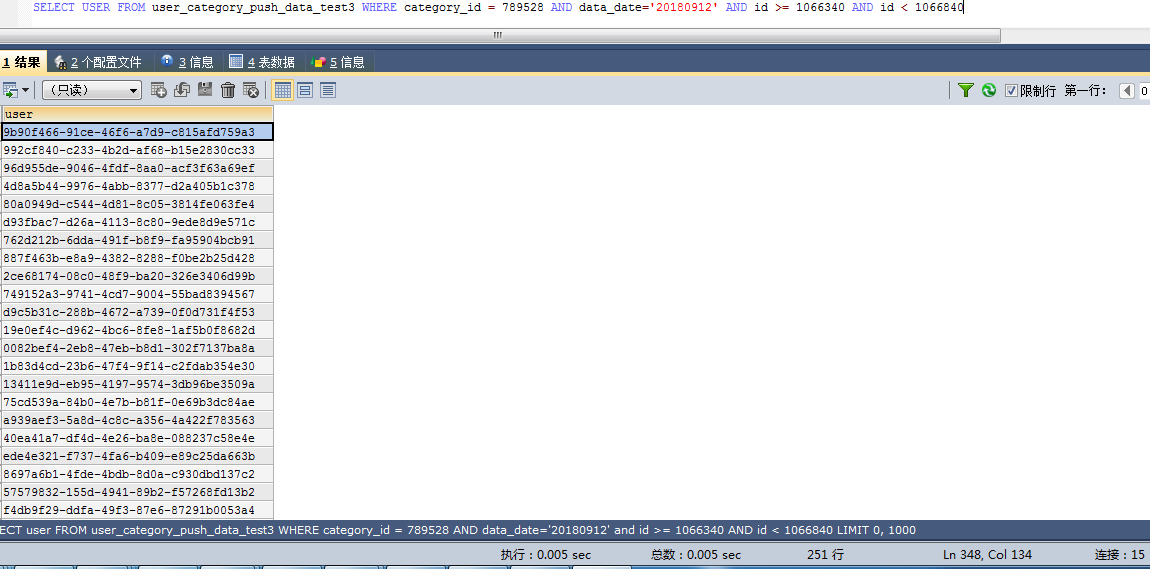


2、mysql分页查询，数据量很大的时候，页码越越大，查询速度越慢，到千万的时候查询一页数据大约要几分钟了。

分页查询速度：下面是100w最后500条查询时间是4.5秒左右，总数据量是1700w



解决办法：不适用limit分页，增加一个整数类型索引字段或者主键，让其取值连续，通过where条件大于等于，小于来替代分页效果好很多：



3、

增加字段

ALTER TABLE user\_category\_push\_data ADD COLUMN data\_date CHAR(8) COMMENT '数据日期';

增加索引

ALTER TABLE user\_category\_push\_data ADD INDEX DATE\_INDEX(DATE) COMMENT '日期索引';

删除索引

ALTER TABLE user\_category\_push\_data DROP INDEX category\_id\_index

快速清除表数据

TRUNCATE user\_category\_push\_data;

查看数据库最大连接

SHOW VARIABLES LIKE '%max\_connections%';

修改列名——将date列改名成data\_date也可修改类型

ALTER TABLE user\_category\_push\_data CHANGE DATE data\_date CHAR(8) COMMENT '日期';

-- 将test字段改为test1

-- ALTER TABLE 表名 CHANGE 原字段名 新字段名 字段类型 约束条件

ALTERTABLE user10 CHANGE test test1 CHAR(**32**) NOTNULLDEFAULT'123';

使用modify修改字段类型 (modify修改列属性)

alter table table\_name modify column\_name bigint not null;

-- 修改字段属性

-- ALTER TABLE tb\_name MODIFY 字段名称 字段类型 [完整性约束条件]

-- 将email字段 VARCHAR(50)修改成VARCHAR(200)

-- 注意，修改时如果不带完整性约束条件，原有的约束条件将丢失，如果想保留修改时就得带上完整性约束条件

ALTERTABLE user10 MODIFY email VARCHAR(**200**) NOTNULLDEFAULT'a@a.com';

-- 将card移到test后面

ALTERTABLE user10 MODIFY card CHAR(**10**) AFTER test;

-- 将test放到第一个，保留原完整性约束条件

ALTERTABLE user10 MODIFY test CHAR(**32**) NOTNULLDEFAULT'123' FIRST;

-- 给age添加默认值

ALTERTABLE user11 ALTER age SET DEFAUTL **18**;

-- 添加一个字段

ALTERTABLE user11 ADD email VARCHAR(**50**);

-- 给email添加默认值

ALTERTABLE user11 ALTER email SETDEFAULT'a@a.com';

-- 删除默认值

ALTERTABLE user11 ALTER age DROPDEFAULT;

ALTERTABLE user11 ALTER email DROPDEFAULT;

-- 添加主键

-- ALTER TABLE tb\_name ADD [CONSTRAINT [sysmbol]] PRIMARY KEY [index\_type] (字段名称,...)

ALTERTABLE test12 ADDPRIMARYKEY(id);

-- 添加复合主键

ALTERTABLE test13 ADD PRIMARYKEY(id,card);

-- 删除主键

ALTERTABLE test12 DROP PRIMARYKEY;

-- 再给test12添加主键, 完整形式

ALTERTABLE test12 ADD CONSTRAINT symbol PRIMARYKEY index\_type(id);

我们知道具有自增长的属性的字段必须是主键，如果表里的主键是具有自增长属性的；那么直接删除是会报错的。如果想要删除主键的话，可以先去年自增长属性，再删除主键

-- 再创建一个表，

CREATETABLE test14(

id INT UNSIGNED KEY AUTO\_INCREMENT

);

-- 删除主键，这样会报错，因为自增长的必须是主键

ALTERTABLE test14 DROPPRIMARYKEY;

-- 先用MODIFY删除自增长属性，注意MODIFY不能去掉主键属性

ALTERTABLE test14 MODIFY id INT UNSIGNED;

-- 再来删除主键

ALTERTABLE test14 DROPPRIMARYKEY;

-- 修改表的存储引擎

-- ALTER TABLE tb\_name ENGINE=存储引擎名称

ALTERTABLE user12 ENGINE=MyISAM;

ALTERTABLE user12 ENGINE=INNODB;

-- ALTER TABLE tb\_name AUTO\_INCREMENT=值

ALTERTABLE user12 AUTO\_INCREMENT=**100**;

-- 添加唯一性约束

-- ALTER TABLE tb\_name ADD [CONSTANT [symbol]] UNIQUE [INDEX | KEY] [索引名称](字段名称,...)

-- 创建测试表

CREATETABLE user12(

id TINYINT UNSIGNED KEY AUTO\_INCREMENT,

username VARCHAR(**20**) NOTNULL,

card CHAR(**18**) NOTNULL,

test VARCHAR(**20**) NOTNULL,

test1 CHAR(**32**) NOTNULL

);

-- username添加唯一性约束，如果没有指定索引名称，系统会以字段名建立索引

ALTERTABLE user12 ADDUNIQUE(username);

-- car添加唯一性约束

ALTERTABLE user12 ADDCONSTRAINT symbol UNIQUEKEY uni\_card(card);

-- 查看索引

SHOW CREATETABLE user12;

-- test,test1添加联合unique

ALTERTABLE user12 ADDCONSTRAINT symbol UNIQUEINDEX mulUni\_test\_test1(test, test1);

-- 删除唯一

-- ALTER TABLE tb\_name DROP {INDEX|KEY} index\_name;

-- 删除刚刚添加的唯一索引

ALTERTABLE user12 DROPINDEX username;

ALTERTABLE user12 DROPKEY uni\_card;

ALTERTABLE user12 DROPKEY mulUni\_test\_test1;

删除列

alter table table\_name drop column column\_name;

修改表明

alter table table\_name\_old rename table\_name\_new

**③创建表之后，为表追加主键。有两种方式，第一位修改表字段属性；第二直接追加。**

ALTER TABLE user\_category\_push\_data ADD PRIMARY KEY(`category\_id`,`user`, `data\_date`);

ALTER TABLE user\_category\_push\_data\_test2 ADD PRIMARY KEY(`category\_id`,`user`, `data\_date`);

alter TABLE my\_pri MODIFY number char(11) PRIMARY KEY;

添加唯一索引

alter table table\_name add unique unique\_name(column\_name)

删除主键

alter table table\_name drop primary key;

建表语句

CREATE TABLE `user\_category\_push\_data\_test1` (

`category\_id` INT(11) NOT NULL COMMENT '组合标签id',

`user` VARCHAR(64) DEFAULT NULL COMMENT '用户的cookieid或者userid',

`data\_date` CHAR(8) DEFAULT NULL COMMENT '日期',

KEY `DATE\_INDEX` (`data\_date`) COMMENT '日期索引',

KEY `CATEGORY\_DATE\_INDEX` (`category\_id`,`data\_date`) COMMENT '分群日期索引',

PRIMARY KEY (`category\_id`,`user`, `data\_date`)

) ENGINE=INNODB DEFAULT CHARSET=utf8 COMMENT='推送push数据表'

mysql 删除数据要用delete from

DELETE FROM user\_category\_push\_data\_test

清除表数据，包括自增长的 auto\_increment值

TRUNCATE user\_category\_push\_data\_test

mysql调优时，多用explain/describe，主要是索引，唯一索引，主键这些方面考虑。

sql\_no\_cache

1.对当前query不使用数据库已有缓存来查询，则当前query花费时间会多点

2.对当前query的产生的结果集不缓存至系统query cache里，则下次相同query花费时间会多点

select SQL\_NO\_CACHE count(\*) from users where email = 'hello';

sql\_cache

新建表同时拷贝表和数据

CREATE TABLE user\_category\_push\_data\_test3 SELECT \* FROM user\_category\_push\_data

新建表只拷贝表结构

CREATE TABLE user\_category\_push\_data\_test3 SELECT \* FROM user\_category\_push\_data where 1=2

CREATE TABLE 新表 SELECT \* FROM 旧表 WHERE 1=2;

CREATE TABLE 新表 LIKE 旧表 ;

第一种只会copy表结构，不会copy索引，主键等

表已经有了，字段一样，copy数据

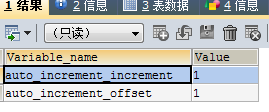
INSERT INTO 新表 SELECT \* FROM 旧表

表已经有了，字段不一样，copy数据

INSERT INTO 新表(字段1,字段2,.......) SELECT 字段1,字段2,...... FROM 旧表

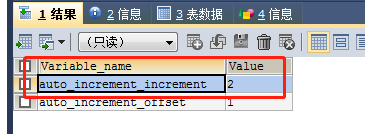
查询当前回话的自增长信息，起始值，步进

SHOW VARIABLES LIKE '%auto\_increment%'



设置回话的自增长步进

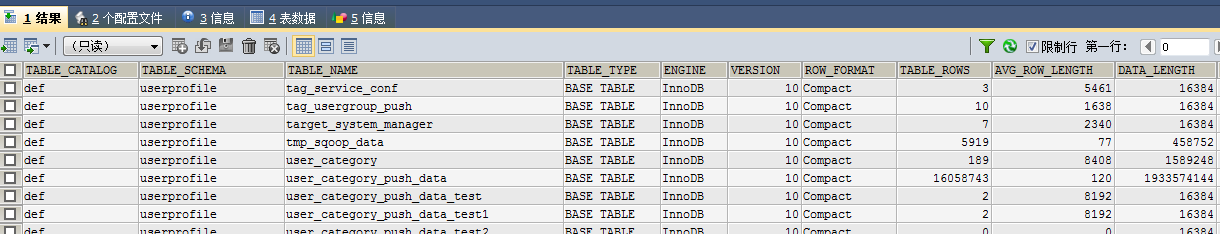
SET auto\_increment\_increment = 2;

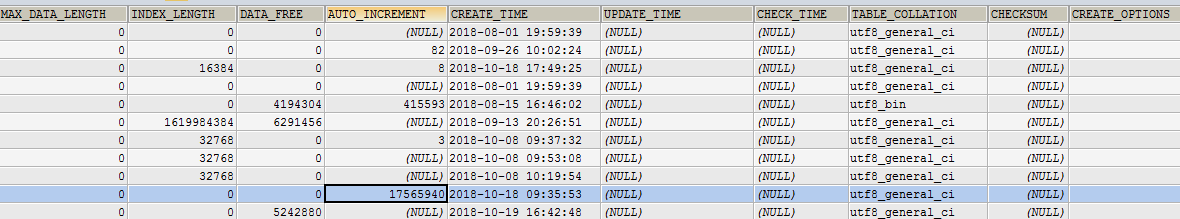


自增长修改是对整个数据库进行修改而不是单张表，且修改是会话级别，关闭并重新打开链接，变量恢复默认值。

查看表的信息

SELECT \* FROM information\_schema.`TABLES`其中就包括自增长的偏移量就是起始值到哪儿了。







聚合查询——查询字段的最大取值

SELECT MAX(id) FROM user\_category\_push\_data\_test3

查询字段的最小取值

SELECT MIN(id) FROM user\_category\_push\_data\_test3

保证自增长字段连续自增，下面两条语句一起执行

alter table emp AUTO\_INCREMENT=1;

INSERT INTO emp(ename,job) VALUES('yangssss11ssss','laoshi') ;

自增长如果是设计到字段改变，必须先删除自增长，后增加(一张表只能有一个自增长)；

修改当前自增长已存在的值，修改只能比最大值大，不能小(小不生效)。

auto\_increment：当对应的字段，不给值 || 给默认值 || 给null 值，会自动的被系统触发。

系统会从当前字段中已有的最大值进行+1操作，得到一个新的不同的字段，作为下一次插入数据是自增长字段的值。若手动修改了auto\_increment，且为向上修改，那么下一次值为auto\_increment！

自增长通常跟主键搭配。

自增长特定：

①任何一个字段要做自增长前提必须是一个索引；

②自增长字段必须是整型数字；

显示数据操作列表

show processlist

查看事物执行情况——权限不够

SELECT command denied to user 'usertag'@'172.31.23.131' for table 'INNODB\_TRX'

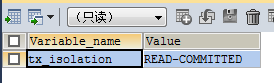
SELECT \* FROM information\_schem.INNODB\_TRX

查看mysql版本

SELECT VERSION()

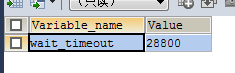
查看mysql 的隔离级别

SHOW VARIABLES LIKE '%iso%';



查看mysql连接超时时间

SHOW GLOBAL VARIABLES LIKE 'wait\_timeout';



显示mysql服务器的状态

SHOW STATUS

删除自增长属性

altertable my\_auto modify id intprimarykey;

--错误，会被认为又定义一个主键;

altertable my\_auto modify id int ;

--正确，且不会丢失掉 id 的主键

唯一键默认值允许自动为空，而且可以多个不同字段为空—空字段不参与唯一性比较。

建表时添加唯一索引

--第一种，字段增加

create table my\_unique(

name varchar(4) unique key,

number varchar(20) unique key

)charset utf8;

--第二种，字段末尾增加

create table my\_unique(

name varchar(4) ,

number varchar(20) ,

unique key name(name),--指定唯一约束名字

unique key(number)--使用默认名字

)charset utf8;

建表后，添加唯一索引

--第一种，modify column

alter table my\_unique modify column name varchar(5) UNIQUE KEY;

--第二种 add constraint unique key

alter TABLE my\_unique add CONSTRAINT num\_uk UNIQUE KEY(number);

--指定唯一约束名字为num\_uk

删除唯一索引

alter table my\_unique drop index num\_uk;

--根据指定约束名字删除，若创建的时候未指定名字，默认使用列名作为唯一约束名字。

--增加复合约束

alter TABLE my\_unique add CONSTRAINT num\_na\_uk UNIQUE KEY(number,name);

删除默认名字复合约束

若不指定名字，那么复合约束名字为第一个字段名字

alter TABLE my\_unique add CONSTRAINT UNIQUE KEY(number,name);

--查看表创建语句

show create TABLE my\_unique;

CREATE TABLE `my\_unique` (

`name` varchar(4) NOT NULL,

`number` varchar(20) DEFAULT NULL,

UNIQUE KEY `number` (`number`,`name`)--这里，名字为number

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--删除复合约束

alter table my\_unique drop index number;

**另外，唯一键同样是索引的类型，故唯一键的添加删除可以如下：**

ALTER TABLE `weight\_num\_curr\_detail`

DROP INDEX `unque\_store\_brand` ;

ADD UNIQUE INDEX `unque\_store\_brand` (`store\_code`, `brand\_code`, `year\_num`) USING BTREE ;

--查看表结构

desc my\_unique;

更改id字段属性为自增属性，在数据库命令行输入下面指令：

alter table `node\_table` change id id int not null auto\_increment primary key;

用mysql查询一个表自增主键的下一个主键值

select auto\_increment from information\_schema.`TABLES` where table\_name=''tableName“ and TABLE\_SCHEMA='databaseName'

tableName为表名

databaseName为表所在的数据库名

mysql分区表

分区有四种类型：range分区、list分区、hash分区、key分区，他们有一些差异，操作也不太一样。range分区是连续的，list分区是不连续的，在插入数据时需要先添加分区。hash分区、key分群，指定分区的列和分区的个数，mysql服务会将列取值通过hash等映射到不同的分区上。删除分区也操作也不太一样。

参考文档：

开源中国https://my.oschina.net/u/1454868/blog/202486

csdn：https://blog.csdn.net/eric\_sunah/article/details/17384073

新建表时就指定为分区表：

|  |
| --- |
| CREATE TABLE employees (  id INT NOT NULL,  fname VARCHAR(30),  lname VARCHAR(30),  hired DATE NOT NULL DEFAULT '1970-01-01',  separated DATE NOT NULL DEFAULT '9999-12-31',  job\_code INT,  store\_id INT  )  PARTITION BY RANGE (YEAR(separated)) (  PARTITION p0 VALUES LESS THAN (1991),  PARTITION p1 VALUES LESS THAN (1996),  PARTITION p2 VALUES LESS THAN (2001),  PARTITION p3 VALUES LESS THAN MAXVALUE  )； |

在已经创建的表设置分区，增加分区字段，下面以list分区为例

ALTER TABLE user\_category\_push\_data\_partition\_test PARTITION BY LIST COLUMNS(data\_date)

增加分区

ALTER TABLE user\_category\_push\_data\_partition\_test ADD PARTITION(

PARTITION p20180919 VALUES IN ('20180919'),

PARTITION p20180913 VALUES IN ('20180913'),

PARTITION p20180912 VALUES IN ('20180912'),

PARTITION p20181017 VALUES IN ('20181017'),

PARTITION p20181018 VALUES IN ('20181018'),

PARTITION p20181019 VALUES IN ('20181019'),

PARTITION p20180911 VALUES IN ('20180911')

)

也可以写在一起

ALTER TABLE user\_category\_push\_data\_partition\_test PARTITION BY LIST COLUMNS(data\_date)(

PARTITION p20181019 VALUES IN ('20181019'),

PARTITION p20181022 VALUES IN ('20181022'))

查看表有几个分区

SELECT TABLE\_NAME, PARTITION\_NAME ,TABLE\_ROWS FROM information\_schema.PARTITIONS WHERE TABLE\_NAME='user\_category\_push\_data';

添加索引

ALTER TABLE user\_category\_push\_data\_partition\_test ADD INDEX CATEGORY\_ID\_DATA\_DATE\_ID(category\_id, data\_date, id)

删除list分区——好像只能一个一个分区删除，可以删除多个，用逗号隔开

ALTER TABLE user\_category\_push\_data\_partition\_test DROP PARTITION p20180912;

删除hash分区、key分区

Alter table history\_uint coalesce partition 511;

取消分区

ALTER TABLE user\_category\_push\_data\_test5 REMOVE PARTITIONING;

删除索引

ALTER TABLE user\_category\_push\_data DROP INDEX DATE\_INDEX

增加字段

ALTER TABLE user\_category\_push\_data ADD COLUMN sequ\_num BIGINT(20) NOT NULL COMMENT '数据序号'

删除字段

ALTER TABLE user\_category\_push\_data DROP COLUMN id;

添加主键

ALTER TABLE user\_category\_push\_data ADD PRIMARY KEY (category\_id, data\_date, sequ\_num);

添加索引

ALTER TABLE user\_category\_push\_data ADD INDEX

INDEX\_CATEGORY\_ID\_DATA\_DATE\_SEQU\_NUM (category\_id, data\_date, sequ\_num);

设置表位分区表，添加list分区

ALTER TABLE user\_category\_push\_data PARTITION BY LIST COLUMNS(data\_date)

(

PARTITION p20181023 VALUES IN ('20181023'),

PARTITION p20181022 VALUES IN ('20181022'))

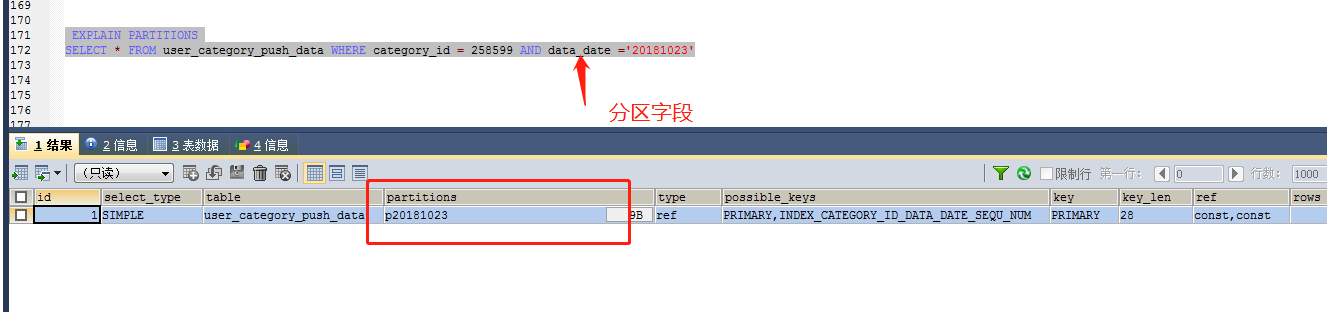
显示数据库当前连接

SHOW PROCESSLIST;

用分区字段作为条件查询

EXPLAIN PARTITIONS

SELECT \* FROM user\_category\_push\_data WHERE category\_id = 258599 AND data\_date ='20181023'



Mysql批量更新；存在就更新，不存在就插入。

insert into on duplicate key update实际例子

<insert id="batchUpdate">  
insert into recommend\_user\_loan\_product\_group\_rel(loan\_rank\_no, visible\_user\_layer\_no, visible\_user\_rate,  
 created\_by, last\_modified\_by) values  
<foreach collection="list" item="config" separator="), (" open="(" close=")">  
#{config.loanRankNo},#{config.visibleUserLayerNo},#{config.visibleUserRate},  
 #{config.createdBy},#{config.lastModifiedBy}  
</foreach>  
on duplicate key update visible\_user\_rate=values(visible\_user\_rate),  
 last\_modified\_by=values(last\_modified\_by), last\_modified\_date=now()  
</insert>

replace into 和insert into on duplicate key update的不同在于：

replace into　操作本质是对重复的记录先delete 后insert，如果更新的字段不全会将缺失的字段置为缺省值

insert into 则是只update重复记录，不会改变其它字段。

表要建立unique索引或者primarykey。语句根据他们来判断数据是否已经存在。

参考：

MYSQL：INSERT INTO .. ON DUPLICATE KEY更新多行记录，防止主键重复

<https://blog.csdn.net/qq_39806256/article/details/80238065>

Mysql中INSERT ... ON DUPLICATE KEY UPDATE的实践

<https://www.jianshu.com/p/78ea17c6d190>

【MySql】Update批量更新与批量更新多条记录的不同值实现方法

<https://www.cnblogs.com/mqxs/p/6794725.html>

MySql中4种批量更新的方法

<https://www.cnblogs.com/PatrickLiu/p/6385167.html>

同mysql服务不同库之间导数据带上库名.表明就好了

