mysql索引以及执行计划

# 执行计划

mysql如何执行一条sql语句，包括sql查询的顺序、是否使用索引、以及使用的索引信息等内容。(<https://juejin.im/post/5a52386d51882573443c852a>)

## 基本语法

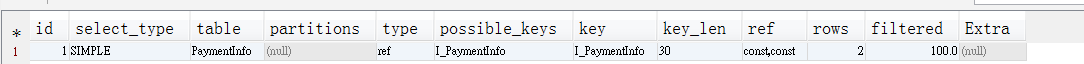
--基本语法 explain sql语句

explain select \* from PaymentInfo;

--其他

explain extended select \* from PaymentInfo where Status = 0 and CreatTime='20190324';

执行计划信息结构



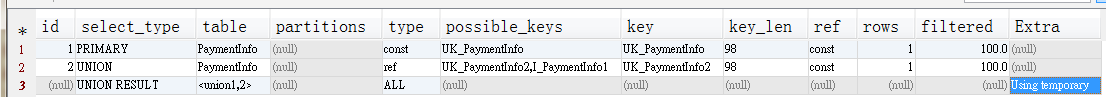
1. Id

查询中各个子查询的执行顺序；

id相同执行顺序由上至下;

id不同，id值越大优先级越高，越先被执行；

id为null时表示一个结果集，不需要使用它查询，常出现在包含union等查询语句中；



1. select\_type

查询类型，主要包含以下类型：

a)SIMPLE : 不包含任何子查询或union等查询;

b)PRIMARY : 包含子查询最外层查询就显示为 PRIMARY;

c)SUBQUERY: 在select或 where字句中包含的查询;

d)DERIVED: from字句中包含的查询;

e)UNION: 出现在union后的查询语句中;

f)UNION RESULT: 从UNION中获取结果集;

(3)table

查询的数据表。

(4)partitions

表分区、表创建的时候可以指定通过那个列进行表分区。

(5)type

访问类型。

a) ALL 扫描全表数据

b)index 遍历索引

c)range 索引范围查找

d)index\_subquery 在子查询中使用 ref

e)unique\_subquery 在子查询中使用 eq\_ref

f)ref\_or\_null 对Null进行索引的优化的 ref

g)fulltext 使用全文索引

h)ref 使用非唯一索引查找数据

i)eq\_ref 在join查询中使用PRIMARY KEYorUNIQUE NOT NULL索引关联

j) const 使用主键或者唯一索引，且匹配的结果只有一条记录。

k)system const 连接类型的特例，查询的表为系统表。

(6)poossible\_keys

可能使用的索引，但是不一定会使用。

(7)key

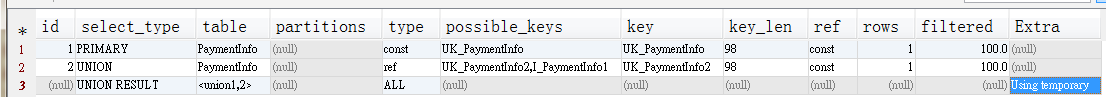
显示MySQL在查询中实际使用的索引，若没有使用索引，显示为NULL。

(8)key\_length

索引长度。

计算公式：(Character Set：utf8mb4=4,utf8=3,gbk=2,latin1=1) \* 列长度 + 1(允许null) + 2(变长列)。

例如：



Id=1或2时，key\_length=98，即32\*3+0+2 = 98；

(9)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。

(10)rows

返回估算的结果集数目，并不是一个准确的值。

(11)filtered

存储引擎返回的数据在server层过滤后，剩下多少满足查询的记录数量的比例，注意是百分比，不是具体记录数。

(12)Extra

辅助信息。

a)distinct：在select部分使用了distinc关键字;

b)no tables used：不带from字句的查询或者From dual查询;

c)使用not in()形式子查询或not exists运算符的连接查询，这种叫做反连接。即，一般连接查询是先查询内表，再查询外表，反连接就是先查询外表，再查询内表;

d)using filesort：排序时无法使用到索引时，就会出现这个。常见于order by和group by语句中;

e)using index：查询时不需要回表查询，直接通过索引就可以获取查询的数;

f)using join buffer（block nested loop），using join buffer（batched key accss）：5.6.x之后的版本优化关联查询的BNL，BKA特性。主要是减少内表的循环数量以及比较顺序地扫描查询;

g)using sort\_union，using\_union，using intersect，using sort\_intersection：

using intersect：表示使用and的各个索引的条件时，该信息表示是从处理结果获取交;

using union：表示使用or连接各个使用索引的条件时，该信息表示从处理结果获取并集;

using sort\_union和using sort\_intersection：与前面两个对应的类似，只是他们是出现在用and和or查询信息量大时，先查询主键，然后进行排序合并后，才能读取记录并返回;

h)using temporary：表示使用了临时表存储中间结果。临时表可以是内存临时表和磁盘临时表，执行计划中看不出来，需要查看status变量，used\_tmp\_table，used\_tmp\_disk\_table才能看出来;

i)using where：表示存储引擎返回的记录并不是所有的都满足查询条件，需要在server层进行过滤。查询条件中分为限制条件和检查条件，5.6之前，存储引擎只能根据限制条件扫描数据并返回，然后server层根据检查条件进行过滤再返回真正符合查询的数据。5.6.x之后支持ICP特性，可以把检查条件也下推到存储引擎层，不符合检查条件和限制条件的数据，直接不读取，这样就大大减少了存储引擎扫描的记录数量。extra列显示using index condition;

j)firstmatch(tb\_name)：5.6.x开始引入的优化子查询的新特性之一，常见于where字句含有in()类型的子查询。如果内表的数据量比较大，就可能出现这个;  
K：loosescan(m..n)：5.6.x之后引入的优化子查询的新特性之一，在in()类型的子查询中，子查询返回的可能有重复记录时，就可能出现这个;

# 索引

索引是一种特殊的查询表，可以被数据库搜索引擎用来加速数据的检索。

注：索引能够提高 SELECT 查询和 WHERE 子句的速度，但是却降低了包含 UPDATE 语句或 INSERT 语句的数据输入过程的速度。

索引创建语法: create index 索引名称 on 表名 (字段名….)。

若创建唯一索引需在create后加上unique。

常用索引的分类：单列索引、唯一索引、组合索引。

1. 单列索引

仅包含一个字段名的索引，称单列索引。例如：

/\*==============================================================\*/

/\* Index: I\_PaymentInfo1 \*/

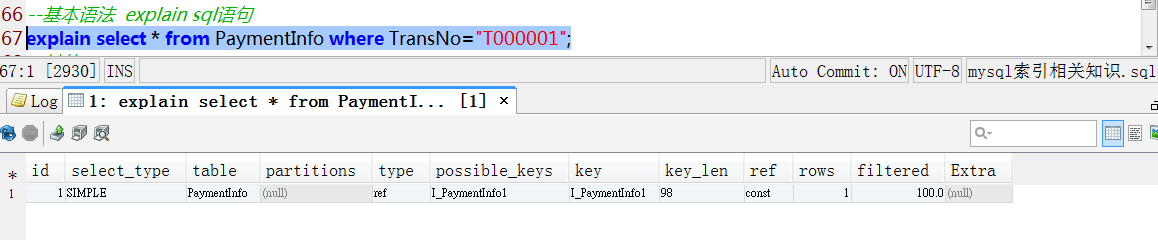
/\*==============================================================\*/

create index I\_PaymentInfo1 on PaymentInfo

(

TransNo

);



1. 唯一索引

唯一索引也是一种特殊的索引。即不允许被索引的数据列包含重复的值。

/\*==============================================================\*/

/\* Index: UK\_PaymentInfo2 \*/

/\*==============================================================\*/

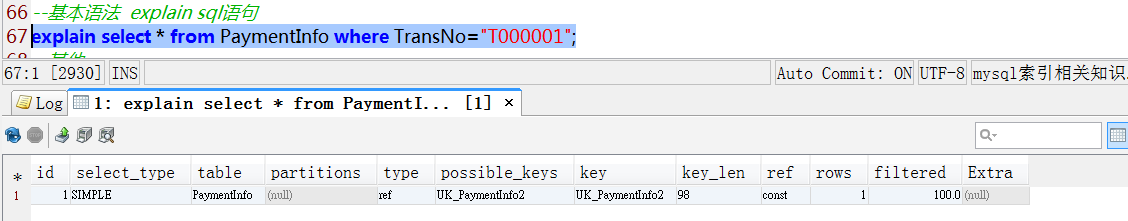
create unique index UK\_PaymentInfo2 on PaymentInfo

(

TransNo,

Status

);



在表中，若插入表中的TransNo和Status值在数据库中已存在时，此时报错，无法插入。

1. 组合索引

即索引中的字段至少两个或两个以上。例如：

/\*==============================================================\*/

/\* Index: I\_PaymentInfo \*/

/\*==============================================================\*/

create index I\_PaymentInfo on PaymentInfo

(

TransNo,

Status

);

## 最左前缀原则

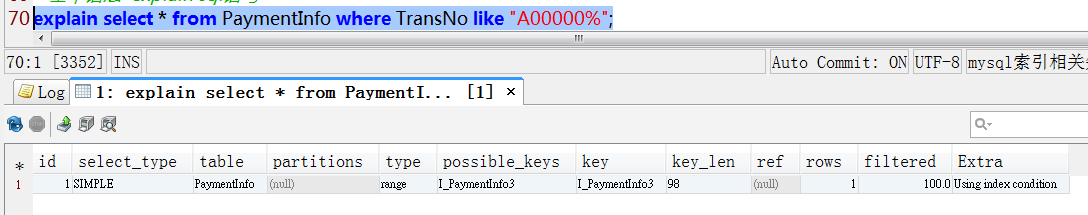
顾名思义，最左优先原则。例如已建立组合索引UK\_PaymentInfo2 index(TransNo, Status){注:index(cloumname,…)，表示建立索引}两个字段。其实已经包含了index(TransNo)，和index(TransNo, Status)两个索引。

若建立索引index(TransNo, Status,WorkDate),就已经包含了index (TransNo){单列索引}, index(TransNo, Status) {组合索引}, index(TransNo, Status,WorkDate) {组合索引}三个索引。

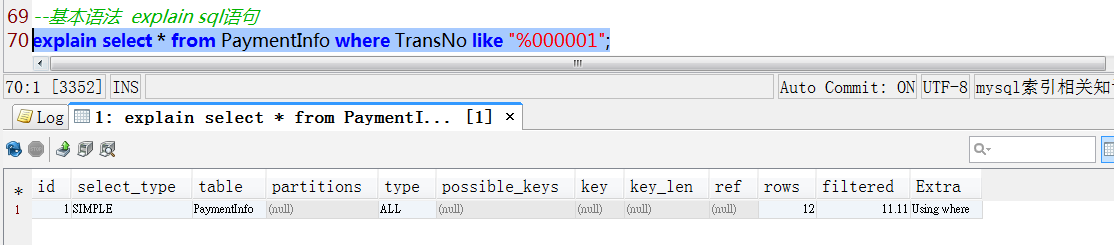
隐含的最左前缀：对于索引的第一个字段，用like时左边必须是固定值，通配符只能出现在右边；字段前加了函数，则索引会被抑制。

eg:

(1)使用like通配符时能够使用索引的情况：



1. 使用like通配符时不能够使用到索引的情况：



select \* from PaymentInfo where Status = 1 and CreatTime="20190324" ; //该语句不使用索引，因为不符合最左前缀原则。

## 避免使用索引情况

1. 表中数据量特别少的时候，仅需全表查询即可；
2. 频繁进行大批量的更新或者插入的表；
3. 若表中的某一列的值为几个特定常量值或者为null,该字段比适合创建索引，比如，table中status仅包含0,1,3三个状态枚举。若status创建index, status=0的数据量特别大，数据库还需进行二次查询，影响效率；
4. 频繁操作的列不适合创建索引，因为每次更新的同时会更新索引数据；

# 例子

首先建立表 缴费信息表 PaymentInfo

drop index I\_PaymentInfo1 on PaymentInfo;

drop index I\_PaymentInfo on PaymentInfo;

drop index UK\_PaymentInfo on PaymentInfo;

drop index UK\_PaymentInfo2 on PaymentInfo;

drop table if exists PaymentInfo;

/\*==============================================================\*/

/\* Table: PaymentInfo \*/

/\*==============================================================\*/

create table PaymentInfo

(

Id int not null auto\_increment,

AcceptId varchar(32) not null,

TransNo varchar(32) not null,

Status int not null,

CreatTime varchar(8) not null,

Extent varchar(32),

primary key (Id)

);

/\*==============================================================\*/

/\* Index: UK\_PaymentInfo \*/

/\*==============================================================\*/

create unique index UK\_PaymentInfo on PaymentInfo

(

AcceptId

);

/\*==============================================================\*/

/\* Index: UK\_PaymentInfo2 \*/

/\*==============================================================\*/

create unique index UK\_PaymentInfo2 on PaymentInfo

(

TransNo,

Status

);

/\*==============================================================\*/

/\* Index: I\_PaymentInfo \*/

/\*==============================================================\*/

create index I\_PaymentInfo on PaymentInfo

(

Status,

CreatTime

);

/\*==============================================================\*/

/\* Index: I\_PaymentInfo1 \*/

/\*==============================================================\*/

create index I\_PaymentInfo1 on PaymentInfo

(

TransNo

);

---插入几条测试记录

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000001", "T000001", 0, "20190323");

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000002", "T000002", 0, "20190323");

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000003", "T000003", 0, "20190323");

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000011", "T000011", 0, "20190323");

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000012", "T000012", 0, "20190324");

insert into PaymentInfo (AcceptId,TransNo,Status,CreatTime) values ("A000013", "T000013", 0, "20190324");