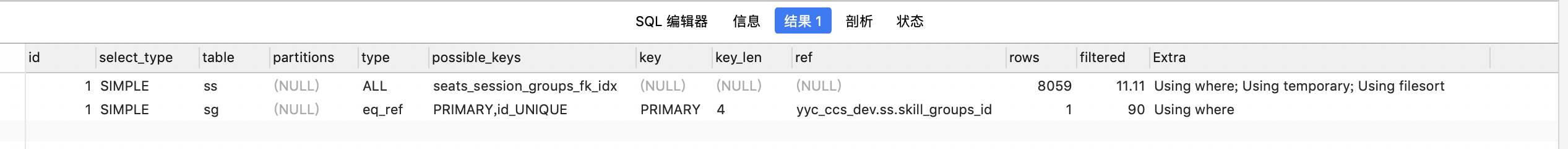
Mysql explain 执行计划每个字段含义



**id**： MySQL Query Optimizer 选定的执行计划中查询的序列号。表示查询中执行 select 子句或操作表的顺序,id值越大优先级越高,越先被执行。id 相同,执行顺序由上至下。

**select\_type**：查询类型，SIMPLE、PRIMARY、UNION、DEPENDENT UNION等。

**table**：显示这一行的数据是关于哪张表的

**type**：这是重要的列，显示连接使用了何种类型。从最好到最差的连接类型为const、eq\_reg、ref、range、indexhe和all

**possible\_keys**：显示可能应用在这张表中的索引。如果为空，没有可能的索引。可以为相关的域从where语句中选择一个合适的语句

**key**： 实际使用的索引。如果为null，则没有使用索引。很少的情况下，mysql会选择优化不足的索引。这种情况下，可以在select语句中使用use index（indexname）来强制使用一个索引或者用ignore index（indexname）来强制mysql忽略索引

**key\_len**：使用的索引的长度。在不损失精确性的情况下，长度越短越好

**ref**：显示索引的哪一列被使用了，如果可能的话，是一个常数

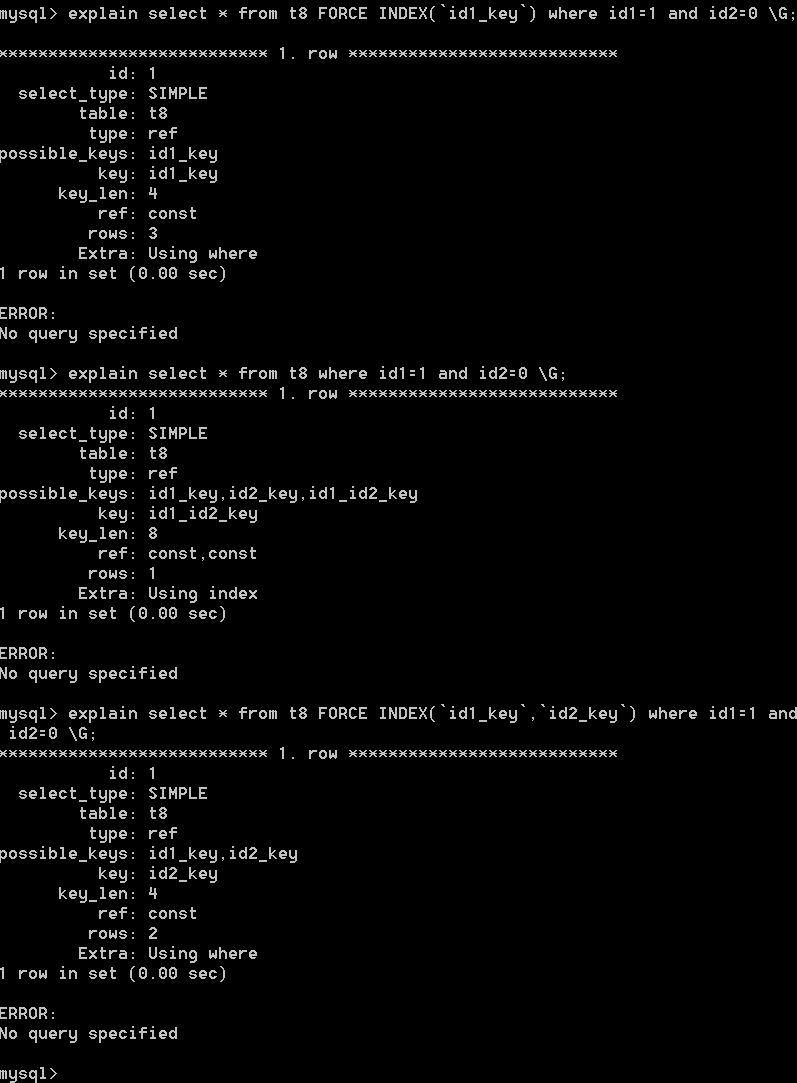
**rows**：mysql认为必须检查的用来返回请求数据的行数   
**extra**：关于mysql如何解析查询的额外信息。

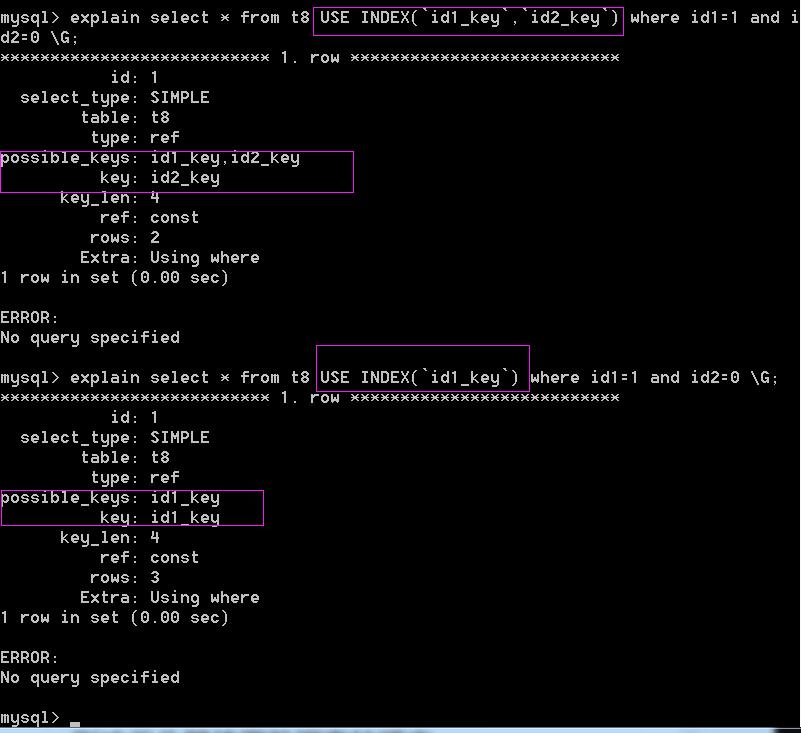
**调节MySQL优化器的优化**

**影响索引的选择**

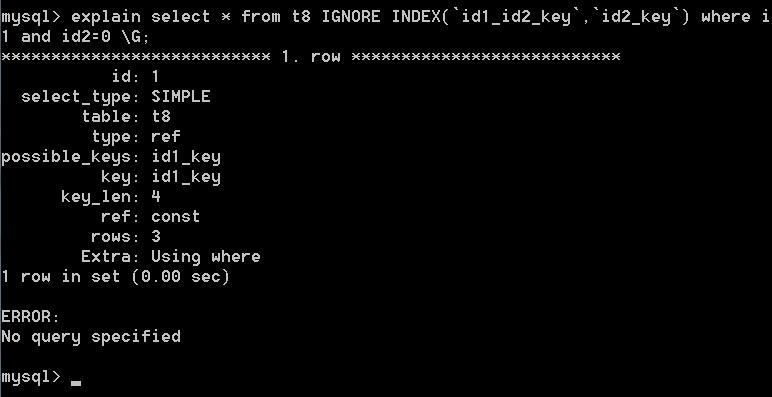
当我们在执行select \* from t8 where id1=1 and id2=0; 语句的时候，优化器会id1\_id2\_key索引，但我们可以通过IGNORE INDEX、 IGNORE INDEX来影响索引的选择

**强制索引**

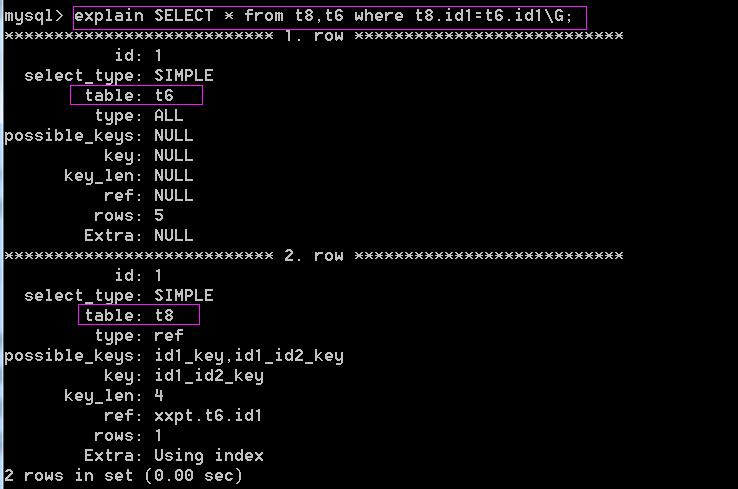
通过FORCE INDEX(索引1[，索引2])或者使用USE INDEX(索引1[，索引2])，来指定使用哪个索引，也可以指定多个索引，让优化器从中挑选。   


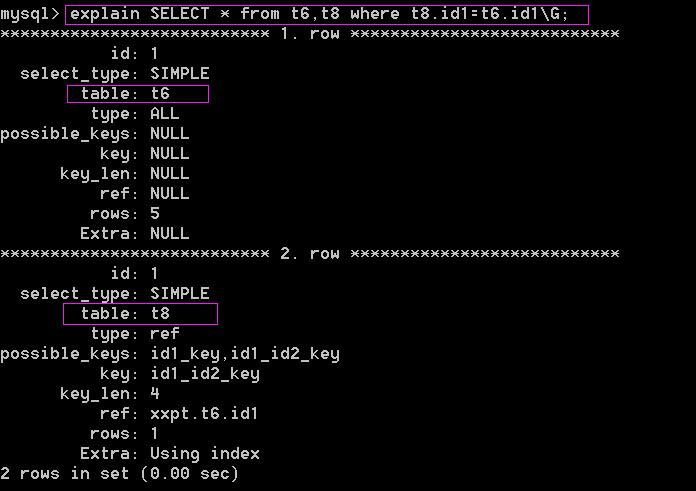
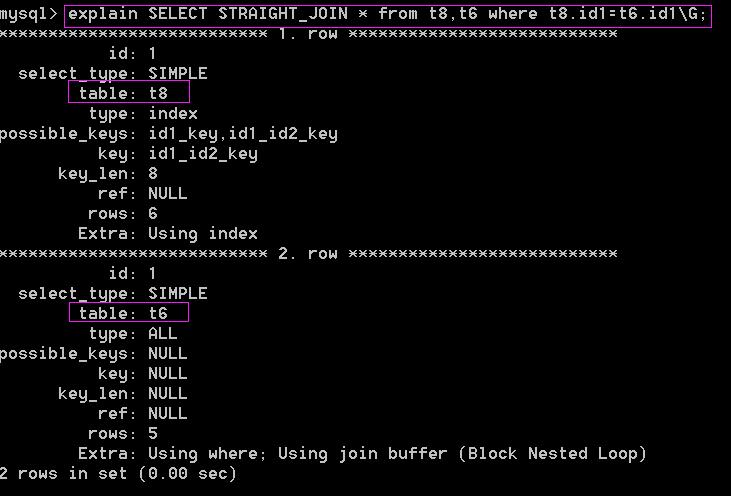
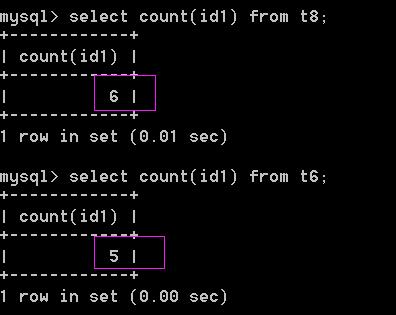
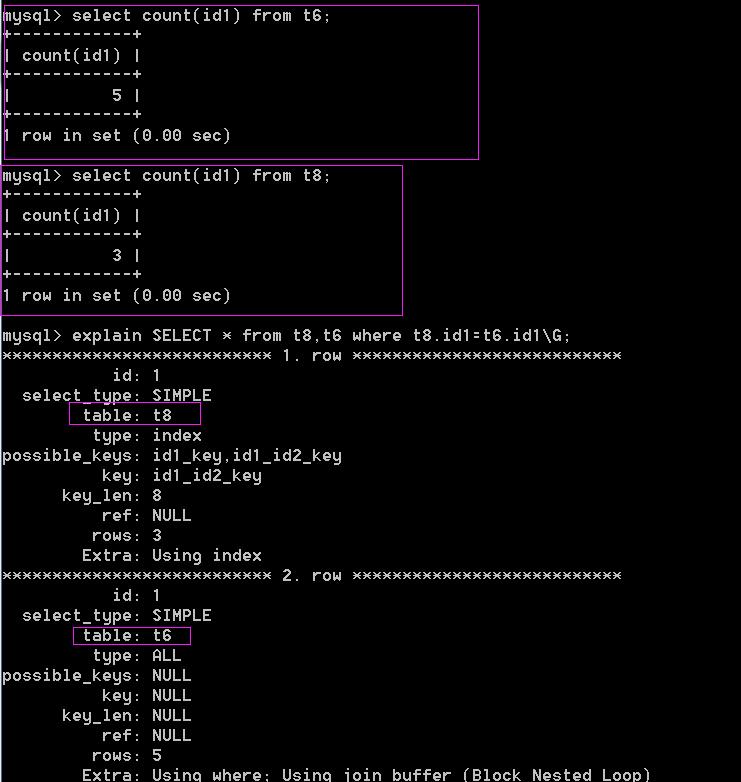


**忽略索引**

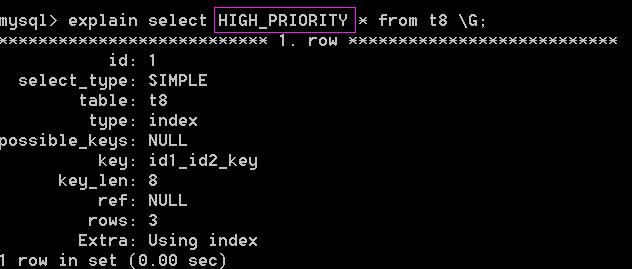
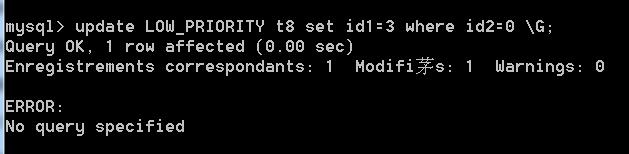
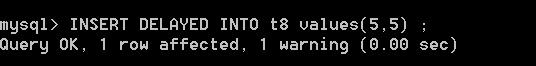
可以使用IGNORE INDEX(索引1[，索引2])来忽略一些索引，这样优化器，就不会考虑使用这些所有，减少优化器优化时间。   


**影响优化器使用数据表的顺序**

一般情况下，MySQL优化器会自行决定按照哪种顺序扫描数据表才能最快地检索出数据，但是我们可以通过STRAGHT\_JOIN强制优化器按特定的顺序使用数据表，毕竟优化器做的判断不一定都是最优的。使用原则是，让限制最强的选取操作最先执行。STRAIGHT\_JOIN可以放在SELECT后面，也可以放在FROM子句中。   
如下图   


   
可以看出，无论from t8,t6还是from t6，t8，都是先检索t6中的表。但是使用STRAIGHT\_JOIN的话，就会按照SQL中顺序。   
   
为什么优化器要选择先判断t6中的数据呢？一个主要的原因，因为t6中数据更少。   
   
如果将t8中数据删除几行后，很明显MySQL优化器选择顺序数据表的顺序就会发生变化。   


**控制SQL语句的优先权**

在高并发的网站中，因为MySQL默认的是写优先，有可能导致一些读操作有效时间内得不到执行机会，HIGH\_PRIORITY可以使用在**select**和**insert**操作中，让MYSQL知道，这个操作优先进行。   
   
LOW\_PRIORITY可以使用在**insert**和**update**操作中，让mysql知道，这个操作将优先权将降低。   
   
INSERT DELAYED告诉MySQL，这个操作将会延时插入。   
INSERT DELAYED INTO，是客户端提交数据给MySQL，MySQL返回OK状态给客户端。而这是并不是已经将数据插入表，而是存储在内存里面等待排队。当mysql有空余时，再插入。另一个重要的好处是，来自许多客户端的插入被集中在一起，并被编写入一个块。这比执行许多独立的插入要快很多，因为它较少了I/O操作。坏处是，不能返回自动递增的ID，以及系统崩溃时，MySQL还没有来得及插入数据的话，这些数据将会丢失。   


**控制查询缓冲**

在实际开发中，一些数据对实时性要求特别高，或者并不经常使用（可能几天就执行一次或两次）,这样就需要把缓冲关了,不管这条SQL语句是否被执行过，服务器都不会在缓冲区中查找该数据，每次都会从磁盘中读取。因为如果实时性要求特别高，缓存中数据可能和磁盘中的就不同步，如果数据不经常使用，被缓存起来，就会占用内存。   
在my.ini中的query\_cache\_type，使用来控制表缓存的。这个变量有三个取值：0,1,2，分别代表了off、on、demand。   
0：表示query cache 是关闭。   
1：表示查询总是先到查询缓存中查找，即使使用了sql\_no\_cache仍然查询缓存，因为sql\_no\_cache只是不缓存查询结果，而不是不使用查询结果。   
2：表示只有在使用了SQL\_CACHE后，才先从缓冲中查询数据，仍然将查询结果缓存起来。   
我本地缓存是关闭的，，如下图。   
