create table T (

ID int primary key,

username VARCHAR(45) DEFAULT NULL,

index username\_inx(username)

)

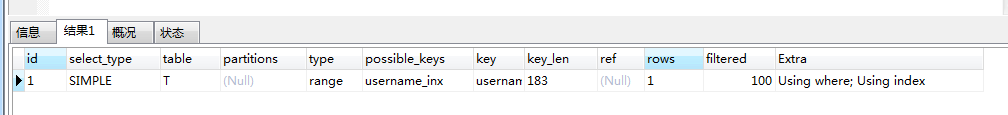
engine=InnoDB;

创建一个表包含两个字段，一个是主键，一个是普通索引

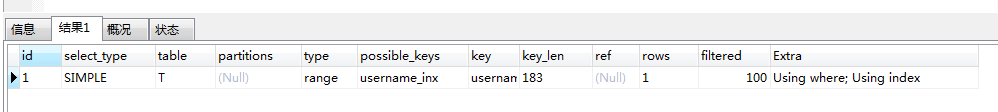
INSERT into T (id, username) values (1, 'test1'),(2, 'test2'),(3, 'test3'),(4, 'test4'),(5, 'test5')

4种情况的执行计划如下：

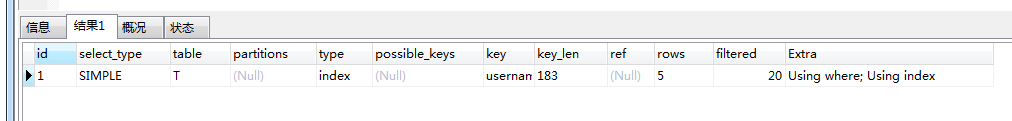
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j'



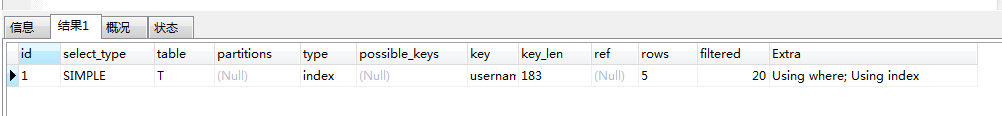
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j%'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j%'



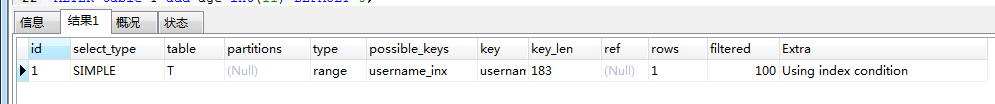
可以看到上面四种情况都使用到了索引，这是因为要查询的字段根据索引就可以全部找到，自然就使用到了索引，后面两种是基于全索引扫描

如果我们新增一个age字段，没有索引

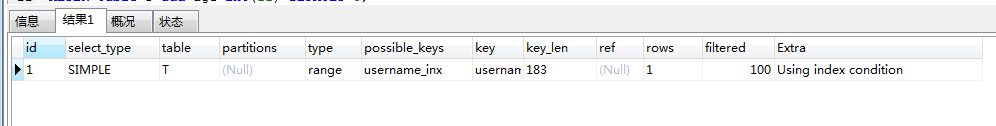
ALTER table T add age int(11) DEFAULT 0;

我们再看4种情况的执行计划

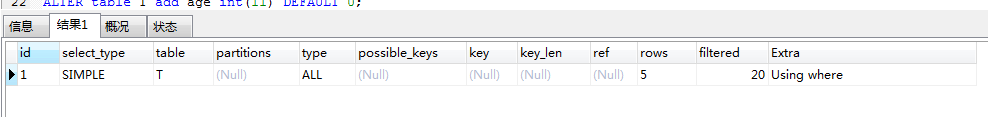
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j'



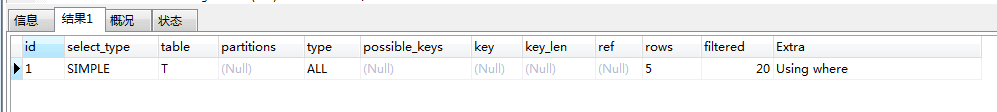
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j%'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j%'



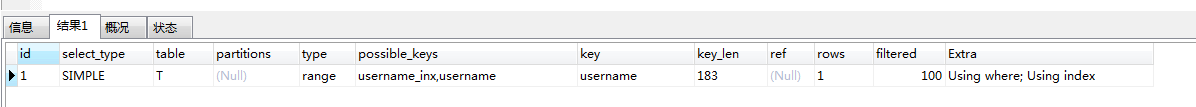
可以看到下面两种情况就没有使用索引了，符合索引左前缀原则

如果给username,age 字段建立联合索引

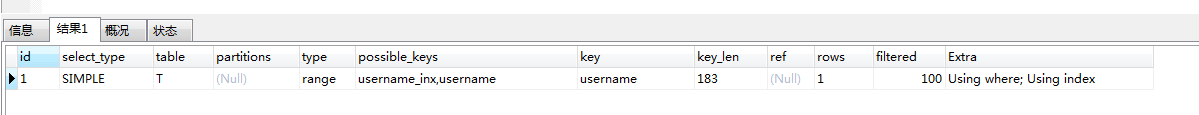
ALTER table T add index(username, age)

我们再看4种情况的执行计划

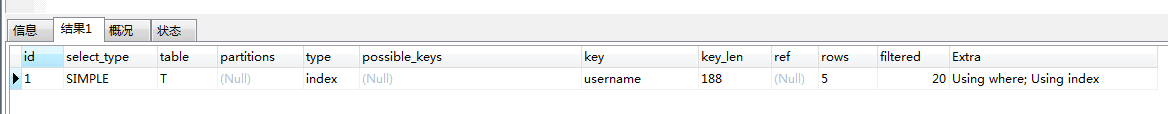
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j'



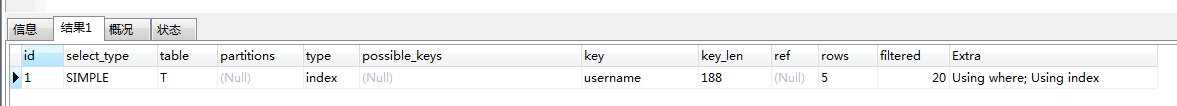
EXPLAIN SELECT \* FROM T WHERE username LIKE 'j%'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j'



EXPLAIN SELECT \* FROM T WHERE username LIKE '%j%'



上面4种情况都使用到了索引