

MYSQL order by排序与索引关系总结

这里主要讨论一下InnoDB B-Tree索引的使用，不提设计，只管使用。B-Tree索引主要作用于WHERE和ORDER BY子句。这里讨论的均在MySQL-Server-5.1.42测试

```
CREATE TABLE `friends` ( `ID` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
`uid` bigint(20) UNSIGNED NOT NULL DEFAULT '0', `fuid` bigint(20) UNSIGNED
NOT NULL DEFAULT '0', `fname` varchar(50) NOT NULL DEFAULT '', `fpicture`
varchar(150) NOT NULL DEFAULT '', `fsex` tinyint(1) NOT NULL DEFAULT '0',
`status` tinyint(1) NOT NULL DEFAULT '0',PRIMARY KEY (`ID`) )
ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8; ALTER TABLE`friends`
ADD INDEX uid_fuid (uid, fuid);
```

1.如果索引了多列，要遵守最左前缀法则。所谓最左前列，指的是查询从索引的最左前列开始，并且不跳过索引中的列。

mysql> explain select uid,fuid,fname from friends where uid = 1936840739;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	friends	ref	uid_fuid	uid_fuid	8	const	251	

1 row in set (0.01 sec)

mysql> explain select uid,fuid,fname from friends where fuid = 1872611129;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	friends	ALL	NULL	NULL	NULL	NULL	4117818	using where

1 row in set (0.00 sec)

mysql> explain select uid,fuid,fname from friends where uid = 1936840739 and fuid = 1872611129;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	friends	ref	uid_fuid	uid_fuid	16	const,const	1	

1 row in set (0.00 sec)

第2条语句,从索引的第二列开始查找，使用索引失败，导致MySQL采用ALL访问策略，即全表查询.在开发中，应该尽量避免全表查询。

2.当MySQL一旦估计检查的行数可能会”太多”，范围查找优化将不会被使用。

mysql> explain select uid,fuid,fname from friends where uid > 1936840738;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	friends	range	uid_fuid	uid_fuid	8	NULL	251	using where

1 row in set (0.01 sec)

mysql> explain select uid,fuid,fname from friends where uid > 21676103;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	friends	ALL	uid_fuid	NULL	NULL	NULL	4117818	using where

1 row in set (0.00 sec)

第2条语句使用了全表查询，它与第1条语句唯一的区别在于需要检查的行数远远多于第1条语句。在应用中，可能不会碰到这么大的查询，但是应该避免这样的查询出现: `select uid from users where registered < 1295001384`

3.索引列不应该作为表达式的一部分，即也不能在索引列上使用函数

```
mysql> explain select uid,fuid,fname from friends where uid = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ref | uid_fuid | uid_fuid | 8 | const | 251 | |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain select uid,fuid,fname from friends where abs(uid) = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ALL | NULL | NULL | NULL | NULL | 4117818 | Using where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> explain select uid,fuid,fname from friends where uid/1 = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ALL | NULL | NULL | NULL | NULL | 4117818 | Using where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

第2和3条语句都有使用表达式，索引派不上用场。

4.尽量借用覆盖索引，减少select * from ...语句使用

```
mysql> explain select uid,fuid from friends where uid = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ref | uid_fuid | uid_fuid | 8 | const | 251 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain select uid,fuid,fsex from friends where uid = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ref | uid_fuid | uid_fuid | 8 | const | 251 | |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain select uid,fuid,id from friends where uid = 1936840739;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ref | uid_fuid | uid_fuid | 8 | const | 251 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

第1句Extra中使用了Using index表示使用了覆盖索引。第3句也使用了覆盖索引,虽然ID不在索引uid_fuid索引列中，但是InnoDB二次索引(second index)叶子页的值就是PK值，不同于MyISAM。Extra部分的Using index表示应用了索引，不要跟type中的index混淆。第2句没有使用覆盖索引，因为fsex不在索引中。

5.ORDER BY子句，尽量使用Index方式排序,避免使用FileSort方式排序
MySQL支持二种方式的排序，FileSort和Index，后者效率高，它指MySQL扫描索引本身完成排序。FileSort方式效率较低。ORDER BY满足以下情况，会使用Index方式排序：

- a)ORDER BY 语句使用索引最左前列。参见第1句

b)使用Where子句与Order BY子句条件列组合满足索引最左前列。参见第2句。

以下情况，会使用FileSort方式的查询

a)检查的行数过多，且没有使用覆盖索引。第3句，虽然跟第2句一样，order by使用了索引最左前列uid，但依然使用了filesort方式排序，因为status并不在索引中，所以没办法只扫描索引。

b)使用了不同的索引，MySQL每回只采用一个索引。第4句，order by出现二个索引，分别是uid_fuid和聚集索引(pk)

c)对索引列同时使用了ASC和DESC。通过where语句将order by中索引列转为常量，则除外。第5句，和第6句在order by子句中，都出现了ASC和DESC排序，但是第5句却使用了filesort方式排序，是因为第6句where uid取出排序需要的数据，MySQL将其转为常量，它的ref列为const。

d)where语句与order by语句，使用了不同的索引。参见第7句。

e)where语句或者ORDER BY语句中索引列使用了表达式，包括函数表达式。参见第8，9句

f)where 语句与ORDER BY语句组合满足最左前缀，但where语句中使用了条件查询。查见第10句，虽然where与order by构成了索引最左有缀的条件，但是where子句中使用的是条件查询。

g)order by子句中加入了非索引列，且非索引列不在where子句中。

h)order by或者它与where组合没有满足索引最左前列。参见第11句和12句，where与order by组合，不满足索引最左前列。(uid, fsex)跳过了fuid

i)当使用left join，使用右边的表字段排序。参见第13句，尽管user.uid是pk，依然会使用filesort排序。

7.高选择性索引。 尽量使用高选择性的过引来过滤数据。高选择性指 Cardinality/#T越接近1，选择性越高，其中Cardinality指表中索引列不重复值(行)的总数。PK和唯一索引，具有最高的选择性，即1。推荐可选性达到20%以上。

```
mysql> explain select ID,uid,fuid from friends where uid = 7884 and fuid = 222294335 and ID = 5801383;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | const | PRIMARY,uid_fuid_fsex | PRIMARY | 4 | const | 1 | |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

这里有二个索引可供使用,而MySQL选择PRIMARY，是因为它具有更高的选择性。

8.谨防where子句中的OR。 where语句使用or，且没有使用覆盖索引,会进行全表扫描。应该尽量避免这样OR语句。尽量使用UNION代替OR

```
mysql> explain select fuid,fsex from friends where uid = 7884 or fuid = 2222943;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | index | uid_fuid_fsex | uid_fuid_fsex | 17 | NULL | 4122237 | Using where; Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain select fuid,status from friends where uid = 7884 or fuid = 2222943;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | friends | ALL | uid_fuid_fsex | NULL | NULL | NULL | 4122237 | Using where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

第1句虽然使用了索引,但是查行时间依然不可以恭维，mysql要检查的行很多,但是返回的行却很少.Extra中的using where表示需要通过where子句抛弃不需要的数据行。

9.LIMIT与覆盖索引 limit子句，使用覆盖索引时比没有使用覆盖索引会快很多

转自：<http://my.oschina.net/longniao/blog/110384?fromerr=dKzaJdu1>