背景：

*为了提高数据库效率，建索引是家常便饭；那么当查询条件为2个及以上时，我们是创建**多个单列索引**还是创建**一个联合索引**好呢？他[illegible]的区别是率高呢？我在这里详细测试分析下。*

一、联合索引测试

注：Mysql版本为 5.7.20

创建测试表(表记录数为63188)：

```
CREATE TABLE `t_mobilesms_11` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `userId` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '' COMMENT '用户id，创建任务时的userid',
  `mobile` varchar(24) NOT NULL DEFAULT '' COMMENT '手机号码',
  `billMonth` varchar(32) DEFAULT NULL COMMENT '账单月',
  `time` varchar(32) DEFAULT NULL COMMENT '收/发短信时间',
  `peerNumber` varchar(64) NOT NULL COMMENT '对方号码',
  `location` varchar(64) DEFAULT NULL COMMENT '通信地(自己的)',
  `sendType` varchar(16) DEFAULT NULL COMMENT 'SEND-发送; RECEIVE-收取',
  `msgType` varchar(8) DEFAULT NULL COMMENT 'SMS-短信; MSS-彩信',
  `serviceName` varchar(256) DEFAULT NULL COMMENT '业务名称. e.g. 点对点(网内)',
  `fee` int(11) DEFAULT NULL COMMENT '通信费(单位分)',
  `createTime` datetime DEFAULT NULL COMMENT '创建时间',
  `lastModifyTime` datetime DEFAULT NULL COMMENT '最后修改时间',
  PRIMARY KEY (`id`),
  KEY `联合索引` (`userId`,`mobile`,`billMonth`)
) ENGINE=InnoDB AUTO_INCREMENT=71185 DEFAULT CHARSET=utf8 COMMENT='手机短信详情'
```

我们为 `userId`，`mobile`，`billMonth` 三个字段添加上联合索引！

我们选择 `explain` 查看执行计划来观察索引利用情况：

1.查询条件为 `userid`

```
EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222'
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	t_mobilesms_11	*(NULL)*	ref	联合索引	联合索引	767	const	1	100

可以通过 `key` 看到，联合索引**有效**

2.查询条件为 `mobile`

```
EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972'
```

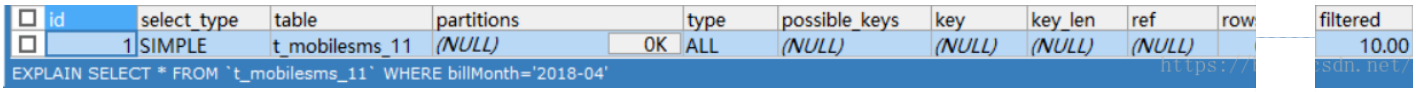
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	t_mobilesms_11	*(NULL)*	ALL	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*	62990	10.00

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972'

可以看到联合索引**无效**

3.查询条件为 `billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE billMonth='2018-04'
```



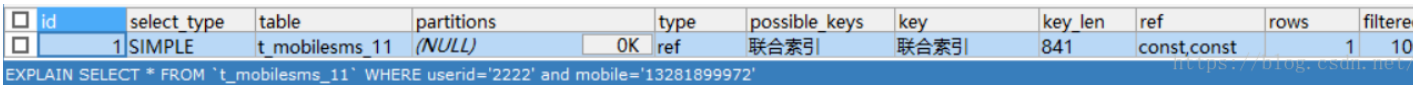
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	row	filtered
1	SIMPLE	t_mobilesms_11	*(NULL)* 0K	ALL	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*		10.00

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE billMonth='2018-04'

联合索引**无效**

4.查询条件为 `userid and mobile`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' AND mobile='13281899972'
```



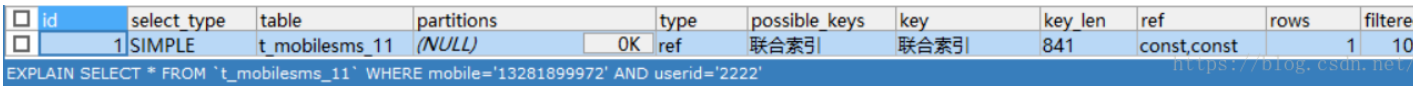
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtere
1	SIMPLE	t_mobilesms_11	*(NULL)* 0K	ref	联合索引	联合索引	841	const,const	1	10

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' and mobile='13281899972'

联合索引**有效**

5.查询条件为 `mobile and userid`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE  mobile='13281899972' AND userid='2222'
```



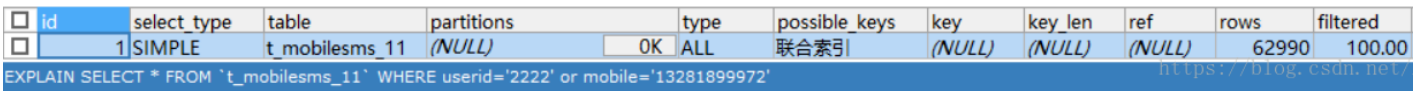
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtere
1	SIMPLE	t_mobilesms_11	*(NULL)* 0K	ref	联合索引	联合索引	841	const,const	1	10

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972' AND userid='2222'

在4的基础上调换了查询条件的顺序，发现联合索引依旧**有效**

6.查询条件为 `userid or mobile`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' OR mobile='13281899972'
```



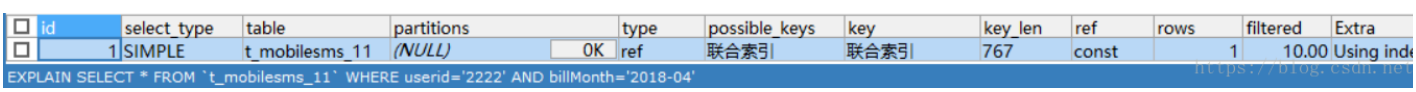
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	t_mobilesms_11	*(NULL)* 0K	ALL	联合索引	*(NULL)*	*(NULL)*	*(NULL)*	62990	100.00

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' or mobile='13281899972'

把 `and` 换成 `or`，发现联合所索引**无效**！

7.查询条件为 `userid and billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' AND billMonth='2018-04'
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t_mobilesms_11	*(NULL)* 0K	ref	联合索引	联合索引	767	const	1	10.00	Using inde

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' AND billMonth='2018-04'

这两个条件分别位于联合索引位置的**第一和第三**，测试联合索引依旧**有效**！

8.查询条件为 `mobile and billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972' AND billMonth='2018-04'
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	t_mobilesms_11	(NULL)	ALL	(NULL)	(NULL)	(NULL)	(NULL)		1.00

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972' AND billMonth='2018-04'

这两个条件分别位于联合索引位置的**第二和第三**，发现联合索引**无效**！

9.查询条件为 `userid and mobile and billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE  userid='2222' AND mobile='13281899972' AND billMonth='2018-
```

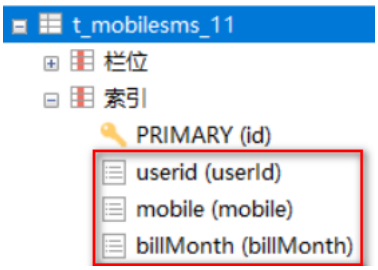
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	t_mobilesms_11	(NULL)	ref	联合索引	联合索引	940	const,const,const	1	1(

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' AND mobile='13281899972' AND billMonth='2018-04'

所有条件一起查询，联合索引**有效**！（当然，这才是最正统的用法啊！）

二、单列索引测试

创建三个单列索引：



1.查询条件为 `userid and mobile and billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE  userid='2222' AND mobile='13281899972' AND billMonth='2018-04'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t_mobilesms_11	ref	userid,mobile,billMonth	userid	767	const	1	Using

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' AND mobile='13281899972' AND billMonth='2018-04'

我们发现三个单列索引**只有 `userid` 有效（位置为查询条件第一个）**，其他两个都没有用上

2.查询条件为 `mobile and billMonth`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972' AND billMonth='2018-04'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t_mobilesms_11	ref	mobile,billMonth	mobile	74	const	1	Using where

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE mobile='13281899972' AND billMonth='2018-04'

我们发现此处两个查询条件**只有 `mobile` 有效（位置也为查询条件第一个）**，后面的无效

3.查询条件为 `userid or mobile`

```
1  EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE  userid='2222' OR mobile='13281899972'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t_mobilesms_11	index_merge	userid,mobile	userid,mobile	767,74	(NULL)	2	Using uni...id,mobile);

EXPLAIN SELECT * FROM `t_mobilesms_11` WHERE userid='2222' or mobile='13281899972'

这次把 `and` 换成 `or`，发现两个查询条件都用上索引了！

三、结论

通俗理解：

利用索引中的附加列，您可以缩小搜索的范围，但使用**一个具有两列的索引 不同于使用两个单独的索引**。复合索引的结构与**电话簿类**...名由姓和名簿首先按姓氏对进行排序，然后按名字对有相同姓氏的人进行排序。**如果您知道姓，电话簿将非常有用；如果您知道姓和名，电话簿则...有用，但如不姓，电话簿将没有用处。**

所以说创建复合索引时，应该仔细考虑**列的顺序**。对索引中的**所有列**执行搜索或仅对**前几列**执行搜索时，**复合索引非常有用；仅对后面的任意列**执行搜索**引则没有用处。**

重点：

多个单列索引在**多条件查询**时只会生效**第一个**索引！所以**多条件联合查询时最好建联合索引！**

最左前缀原则：

顾名思义是最左优先，以**最左边的为起点**任何连续的索引都能匹配上，
注：如果第一个字段是**范围查询需要单独建一个索引**
注：在创建联合索引时，要根据业务需求，where子句中**使用最频繁的一列放在最左边**。这样的话扩展性较好，比如 `userid` 经常需要作为查询条件不常常用，则需要把 `userid` 放在联合索引的**第一位置，即最左边**

同时存在联合索引和单列索引（字段有重复的），这个时候查询mysql会怎么用索引呢?

这个涉及到mysql本身的**查询优化器策略**了，当一个表有多条索引可走时, Mysql 根据**查询语句的成本**来选择走哪条索引；

有人说where查询是按照从左到右的顺序，所以筛选力度大的条件尽量放前面。网上百度过，很多都是这种说法，但是据我研究，**mysql执行优化器会...，当不考虑索引时，where条件顺序对效率没有影响，真正有影响的是是否用到了索引！**

联合索引本质：

当创建**(a,b,c)联合索引**时，相当于创建了**(a)单列索引**，**(a,b)联合索引**以及**(a,b,c)联合索引**
想要索引生效的话,只能使用 a和a,b和a,b,c三种组合；当然，我们上面测试过，**a,c组合也可以，但实际上只用到了a的索引，c并没有用到！**
注：这个可以结合上边的 *通俗理解* 来思考！

其他知识点：

1、需要加索引的字段，**要在where条件中**
2、**数据量少的字段不需要加索引**；因为**建索引有一定开销**，如果数据量小则没必要建索引（速度反而慢）
3、如果where条件中是**OR**关系，加索引**不起作用**
4、**联合索引比对每个列分别建索引更有优势**，因为索引建立得越多就越占磁盘空间，在更新数据的时候速度会更慢。另外建立多列索引时，顺序也是**应该将严格的索引放在前面，这样筛选的力度会更大，效率更高。**