话单查询慢优化

场景：计费话单查询

计费话单条件查询

```
2019 - 02 - 22T09 : 32 : 16.777 + 0800 I COMMAND[conn163]command gzbsms.ucCallRecord command : find {
    find : "ucCallRecord",
    filter : {
        $and : [{
                tenementId : {
                    $in : ["880001"]}}, {calling : /^.*290727.*$/}, {called : /^.*1369.*$/}, {direction : "in"}, {trunkId : "SIPT_4_880001"},
                    {wireNumber : /^.*0551.*$/}, {beginTime : {$gte : 1548000000000}}, {beginTime : {$lte : 1550764799000}}]
    },
    sort : {
        beginTime : -1
    },
    limit : 15
}
planSummary : IXSCAN {
    direction : 1
}
keysExamined : 3511273 docsExamined : 3511273 hasSortStage : 1 cursorExhausted : 1 numYields : 27508 nreturned : 0 reslen : 107 locks : {
    Global : {acquireCount : {r : 55018},acquireWaitCount : {r : 114},timeAcquiringMicros : {r : 136482}},
    Database : {acquireCount : {r : 27509}},
    Collection : {acquireCount : {r : 27509}
    }
}
protocol : op_query 20453ms
```

如上图所示:

- 查询条件: 企业ID: **880001**, 主叫：**290727**，被叫：**1369**，入局，中继：**cop**, 外线号：**0551**，时间范围：**2019-01-21 -- 2019-2-21**
- 分页数量为 **15**条
- 扫描数据量为**3511273** 个文档，**3511273** 个索引
- **查询到的数据为 0 条**
- **耗时 20.45秒**

查询慢的原因

- 查询条件多，需要过滤的文档多，导致扫描数据大
- 没有命中排序索引，需要排序的数据量大 (**模糊查询导致**)
- 条件索引多，选举耗时的耗长
- 内存不够

优化思路:

- 分词：把模糊查询优化成，分词匹配，确保索引能够精确命中（模糊查询会扫描所有的索引记录）
- 保证内存空间：确保索引都在内存中扫描
- 分表/分库： 尽可能地使扫描数据记录减少

模糊优化方案

- 最左前缀匹配
- 模糊查询字段分词匹配

环境:

数据： 2700W条

分词查询环境: 192.168.76.82

模糊查询环境: 192.168.76.84

左前缀查询环境: 192.168.76.84

192.168.76.82：数据库占用内存: 4.1G, 虚拟内存: 4.5 G ucCallRecord: 数据大小 68.9G 压缩文件大小：33.5G 索引大小：57.6G

192.168.76.84：数据库占用内存: 7.1G, 虚拟内存: 7.4G ucCallRecord: 数据大小 21.5G 压缩文件大小：6.5G 索引大小：2.9G,

查询条件

模糊查询

```
db.ucCallRecord.find({
    tenementId : {
        $in : ["880001"]
    },
    calling : {
        $regex : /^.*290727.*$/
    },
    direction : "in",
    trunkId : "SIPT_4_880001",
    beginTime : {
        $gte : 1548000000000,
        $lte : 1550764799000
    }
}).sort({
    beginTime : -1
}).limit(15).explain('executionStats');
```

左前缀查询

```
db.ucCallRecord.find({
    tenementId : {
        $in : ["880001"]
    },
    calling : {
        $regex : /^290727.*$/
    },
    direction : "in",
    trunkId : "SIPT_4_880001",
    beginTime : {
        $gte : 1548000000000,
        $lte : 1550764799000
    }
}).sort({
    beginTime : -1
```

```
}).limit(15).explain('executionStats');
```

分词查询

```
db.ucCallRecord.find({
    tenementId : {
        $in : ["880001"]
    },
    callingarr : '290727',
    direction : "in",
    trunkId : "SIPT_4_880001",
    beginTime : {
        $gte : 1548000000000,
        $lte : 1550764799000
    }
}).sort({
    beginTime : -1
}).limit(15).explain('executionStats');
```

查询条件	模糊查询	左前缀	分词
主呼叫：290727 企业：880001 方向：入局 中继：SIPT_4_880001	索引扫描：653948， 文档扫描：653948，耗时：91625 ms, 返回：2	索引扫描：1537， 文档扫描：1541，耗时：177ms, 返回：0	索引扫描：2034， 文档扫描：2034， 耗时：65ms, 返回：2
主呼叫：290727 企业：880001 中继：SIPT_4_880001	索引扫描：8539， 文档扫描：8539，耗时：129ms, 返回：15	索引扫描：1537， 文档扫描：1541，耗时：51ms, 返回：15	索引扫描：16， 文档扫描：16，耗时：15ms, 返回：15
主呼叫：290727 中继：SIPT_4_880001	索引扫描：8539， 文档扫描：8539，耗时：81ms, 返回：15	索引扫描：1541， 文档扫描：1537，耗时：25ms, 返回：15	索引扫描：16， 文档扫描：16，耗时：0ms, 返回：15
主呼叫：290727	索引扫描：12669， 文档扫描：12669，耗时：73ms, 返回：15	索引扫描：1541， 文档扫描：1537，耗时：14ms, 返回：15	索引扫描：15， 文档扫描：15，耗时：0ms, 返回：15

由此可见条件越多查询，扫描文档或者索引越多，查询耗时越慢

查询条件	模糊查询	左前缀	分词
主呼叫：290727 企业：880001 方向：入局 中继： SIPT_4_880001	索引扫描：653948， 文档扫描：653948， 耗时：91625ms, 返回：2	索引扫描：1540， 文档扫描：1537， 耗时：472 ms, 返回：0	索引扫描：2034， 文档扫描：2034， 耗时：65ms, 返回：2
主呼叫：2907 企业：880001 方向：入局 中继： SIPT_4_880001	索引扫描：28159， 文档扫描：28159， 耗时：1311ms, 返回：15	索引扫描：36798， 文档扫描：36560， 耗时：5249ms, 返回：0	索引扫描：2377， 文档扫描：2377， 耗时：148 ms, 返回：15
主呼叫：290 企业：880001 方向：入局 中继： SIPT_4_880001	索引扫描：3707， 文档扫描：3707， 耗时：145ms, 返回：15	索引扫描：344784， 文档扫描：342444， 耗时：17502 ms, 返回：2	索引扫描：2570， 文档扫描：2570， 耗时：532ms, 返回：15
主呼叫：29 企业：880001 方向：入局 中继： SIPT_4_880001	索引扫描：36， 文档扫描：36， 耗时：23 ms, 返回：15	索引扫描：399454， 文档扫描：396748， 耗时：21831 ms, 返回：15	索引扫描：338， 文档扫描：338， 耗时：26 ms, 返回：15

测试小结：

左前缀匹配索引方式进行模糊查询

优点

- 无需做业务上的修改，后期开发灵活，只需要查询时使用正则表达式即可
- 在查询越精确的数据速度越快，若没有查询到匹配的记录会返回很快， 基本上都是毫秒级别
- 内存，存储压力较少

缺点

- 当查询匹配到的记录越多时，返回数据越慢（因为需要对查询到的记录进行排序）

分词方式进行模糊查询

优点

- 数据按照索引顺序，进行精确匹配，查询速度很快

缺点

- 空间换时间： 数据大小增加,所需要内存增大（若内存不足可能会很大程度上影响查询速度）
- 逻辑复杂: 需要对模糊匹配字段进行分词，同时需要添加数组索引， 在逻辑复杂度增加
- 操作不灵活： 当需求变更是，若添加/删除模糊字段，需要重新定义数据结构

- 若是有多个模糊查询需要命中到索引，则会扫描匹配分词数组，测试若分词数量大时，查询会相对的慢很多（**建议不要同时使用两个以上的模糊查询**）

针对查询方式优化建议：

- **查询条件尽可能简单（不建议条件组合太复杂）**
- **模糊查询并且数据量（1000w以上）时建议： 查询字段的值进行分词**
- **排序字段若是查询条件时建议选为必填字段**
- **尽可能减少两个以上的模糊查询字段同时查询**
    - 主叫，被叫，外线号码 三选其一
- **模糊查询字段长度需要控制字段值长度少于20个字符**