管理系统-数据查询SQL语句优化建议

1.录音记录查询

场景

- 查询范围(部门)内相关用户的录音记录
  - 管理员管理多个部门(可达上万个部门)
  - 每个部门对应多个用户
  - 每个用户存在多条录音

数据结构

- BMS数据库： 管理员权限范围表

```
CREATE TABLE `bmsUserScope` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `userId` varchar(64) CHARACTER SET utf8mb4 NOT NULL COMMENT '用户ID',
  `tenementId` varchar(64) CHARACTER SET utf8mb4 NOT NULL COMMENT '企业ID',
  `departmentId` varchar(64) CHARACTER SET utf8mb4 NOT NULL COMMENT '部门ID',
  `descendantVisible` tinyint(3) unsigned NOT NULL DEFAULT '1' COMMENT '1表示包含子孙部门，0表示不包含',
  PRIMARY KEY (`id`),
  KEY `userId` (`userId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- EIM数据库： 部门用户表

```
CREATE TABLE `ucDepartmentUser` (
  `departmentId` varchar(64) NOT NULL,
  `username` varchar(64) NOT NULL,
  PRIMARY KEY (`departmentId`,`username`),
  KEY `ucDepartmentUser_username_idx` (`username`(20))
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- Record录音记录数据库：录音记录表(按月分表)

```
CREATE TABLE `recordInfo_201904` (
  `idx` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `cdrid` char(64) NOT NULL DEFAULT '0_0',
  `caller` char(64) NOT NULL,
  `callee` char(64) NOT NULL,
  `callDir` int(11) NOT NULL DEFAULT '0',
  `owner` char(64) NOT NULL,
```

```
  `ownerType` tinyint(4) NOT NULL DEFAULT '0',
  `ownerDepIDs` varchar(256) DEFAULT NULL,
  `recName` varchar(256) NOT NULL,
  `storageName` varchar(256) NOT NULL,
  `startTime` int(8) NOT NULL,
  `longTime` int(11) NOT NULL DEFAULT'0',
  `fileSize` int(11) NOT NULL DEFAULT'0',
  `TenantID` int(11) NOT NULL,
  `isConnected` tinyint(1) NOT NULL DEFAULT '0',
  `mark` tinyint(4) DEFAULT '0',
  `expiredTime` int(11) NOT NULL,
  `comment` varchar(256) DEFAULT NULL,
  PRIMARY KEY (`idx`),
  KEY `TenantID_idx` (`TenantID`),
  KEY `cdrid_dir_idx` (`cdrid`,`callDir`),
  KEY `Owner_timer_idx`(`owner`,`startTime`),
  KEY `StartTime_idx` (`startTime`),
  KEY `Caller_idx` (`caller`),
  KEY `Callee_idx` (`callee`),
  KEY `expiredTime_idx` (`expiredTime`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

查询语句

```
# 查询管理员可管理的部门列表 departmentIds
select departmentId from bms.bmsUserScope where userId = "u10001";

# 查询可管理的用户列表 userIds
select userId from eim.ucDepartmentUser where departmentId in (${departmentIds});

# 查询可见的录音记录 *${userIds}可能上万个*
SELECT
     idx, cdrid, caller, callee, callDir, owner, ownerType, recName, storageName, startTime, longTime, fileSize,
TenantID as tenantId, isConnected, mark, comment
FROM Record.RecordInfo_201903
WHERE
TenantID  =  880001  AND
isConnected = 1
AND startTime >= 1551369600
AND startTime <= 1554047999
   AND  (caller  IN(${userIds})  OR
   callee IN(${userIds}))
```

原始执行计划：

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	RecordInfo_201903	range	TenantID_idx,StartTime_ic	StartTime_idx	4	(Null)	2355128	Using where

执行速度:

手动创建10000个userId组成SQL，速度是22到40秒。

问题分析：

1. 数据在服务器和客户端流动多次；
2. In 列表的常量太多；

推荐优化方案:

1. 将列表 ${departmentIds}存入临时表。下面是例子：

```
Create temporary table userId_List(primary key userId_List_idx(username))
   select distinct username
from eim.ucDepartmentUser
where departmentId in
(select departmentId from bms.bmsUserScope where userId = "u10001");
```

2. 改写SQL，用临时表代替departmentIds列表字符串 （0.016秒）

```
SELECT
idx, cdrid, caller, callee, callDir, owner, ownerType, recName,
storageName, startTime, longTime, fileSize, TenantID as tenantId,
isConnected, mark, comment
FROM Record.RecordInfo_201905, userId_List
WHERE
TenantID = 880001 AND isConnected = 1
AND startTime >= 1556640204
AND startTime <= 1559232000
AND ((caller = username) or (callee = username))
order by startTime
```

```
limit 15;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	RecordInfo_201905	range	TenantID_idx,StartTime_idx	StartTime_idx	4	NULL	2785020	Using where
1	SIMPLE	userId_List	ALL	PRIMARY	NULL	NULL	NULL	10143	Range checked for each record (index map: 0x1)

3．删除临时表

```
drop temporary table userId_List;
```

优化结果分析：

1. 速度从30-40秒，提高到7-8秒（主要是创建临时表的时间）；

2.挂机短信查询

场景

查询管理范围内相关 “中继号码” 关联的 “短信内容”

- 每个管理员对应多部门(可达上万个部门)
- 每个部门关联多个中继号码
- 每个中继号码关联一条短信内容记录(如果有)

数据结构

部门中继号码关联表

```
CREATE TABLE `bms_departmentOutwireNumber` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `tenementId` varchar(64) NOT NULL COMMENT '企业ID',
  `tenantId` int(10) unsigned NOT NULL COMMENT 'VOS企业ID',
  `departmentId` varchar(64) NOT NULL COMMENT '部门ID',
  `outwireNumber` varchar(32) NOT NULL COMMENT '中继号码',
  `trunkId` varchar(64) NOT NULL COMMENT '中继',
  `inputTime` bigint(13) unsigned NOT NULL DEFAULT '0' COMMENT '添加时间',
  PRIMARY KEY (`id`),
  KEY `departmentId` (`departmentId`),
  KEY `outwireNumber` (`outwireNumber`),
  KEY `trunkId` (`trunkId`) USING BTREE,
  KEY `tenantId` (`tenantId`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='部门号码关系表';
```

挂机短信记录表

```sql
CREATE TABLE `bms_onHookMessage` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `tenementId` varchar(64) NOT NULL COMMENT '企业ID',
  `tenantId` int(10) unsigned NOT NULL COMMENT 'VOS企业ID',
  `outwireNumber` varchar(32) NOT NULL COMMENT '中继号码 ',
  `trunkId` varchar(64) NOT NULL COMMENT '中继ID',
  `callInMsg` text NOT NULL COMMENT '呼入短信内容',
  `callOutMsg` text NOT NULL COMMENT '呼出短信内容',
  `callInEnabled` tinyint(1) unsigned NOT NULL DEFAULT '0' COMMENT '1开启 0 关闭',
  `callOutEnabled` tinyint(1) unsigned NOT NULL DEFAULT '0' COMMENT '1开启 0 关闭',
  `inputTime` bigint(13) unsigned NOT NULL DEFAULT '0' COMMENT '添加时间',
  `modifyTime` bigint(13) unsigned NOT NULL DEFAULT '0' COMMENT '修改时间',
  PRIMARY KEY (`id`),
  KEY `tenantId_outnum_trunkId` (`tenantId`,`trunkId`,`outwireNumber`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='挂机短信';
```

查询语句

```
# 查询列表 ${departmentIds} 可达上万个
部门ID # callInMsg,callOutMsg需要支持
模糊搜索
SELECT
    a.tenantId,
    a.tenantId,
    a.departmentId,
    a.outwireNumber,
    a.trunkId,
    c.callInMsg,
    c.callOutMsg,
    c.callInEnabled,
    c.callOutEnabled,
    a.inputTime AS
    outwireInputTime,
    c.inputTime,
    c.modifyTime
FROM
    (
        SELECT
            *
        FROM
            bms_departmentOutwireNumbe
        r WHERE
            departmentId IN(${departmentIds})
    )a
INNER JOIN bms_onHookMessage c
    ON( c.tenantId =
    a.tenantId
    AND c.outwireNumber = a.outwireNumber
    AND c.trunkId = a.trunkId
    AND c.callInMsg LIKE CONCAT('%', '会计',
    '%') AND c.callOutMsg LIKE CONCAT('%',
    '98798', '%')
)
WHER
E
    a.tenantId =
'880001' ORDER BY
    a.outwireNumber
ASC LIMIT 0,15
```

原始执行计划：

```
+----+-------------+-----------------------------+------+-----------------------+-----------+---------+-------+------+----------------------------------------------+
| id | select_type | table                       | type | possible_keys         | key       | key_len | ref   | rows | Extra                                        |
+----+-------------+-----------------------------+------+-----------------------+-----------+---------+-------+------+----------------------------------------------+
|  1 | PRIMARY     | <derived2>                  | ref  | <auto_key0>           | <auto_key0> | 4       | const |   30 | Using temporary; Using filesort              |
|  1 | PRIMARY     | c                           | ALL  | tenantId_outnum_trunkId | NULL      | NULL    | NULL  |    5 | Using where; Using join buffer (Block Nested Loop) |
|  2 | DERIVED     | bms_departmentOutwireNumber | ALL  | departmentId          | NULL      | NULL    | NULL  | 9667 | Using where                                  |
+----+-------------+-----------------------------+------+-----------------------+-----------+---------+-------+------+----------------------------------------------+
```

执行速度:

基于人工随机产生的数据，原始SQL在3到4秒左右。

潜在的问题分析：

1. 数据在服务器和客户端流动多次；
2. In 列表的常量太多；

推荐优化方案（如果数据量变大，出现性能问题时尝试）：

1．将列表 ${departmentIds}存入临时表。下面是例子：

```
create temporary table departmentId_List(primary key
departmentId_idx(departmentId))
select distinct(departmentId) from bms_departmentOutwireNumber limit
10000;
```

2．改写SQL，用临时表代替departmentIds列表字符串，推入条件。

```
SELECT
a.tenantId, a.tenantId, a.departmentId, a.outwireNumber, a.trunkId,
c.callInMsg, c.callOutMsg, c.callInEnabled, c.callOutEnabled,
a.inputTime AS outwireInputTime, c.inputTime,
c.modifyTime FROM
(
SELECT
* FROM
bms_departmentOutwireNumber WHERE
departmentId IN (select departmentId from departmentId_List)
and tenantId = '880001'
)a
INNER JOIN bms_onHookMessage c force Index(tenantId_outnum_trunkId)
ON( c.tenantId = a.tenantId
AND c.outwireNumber = a.outwireNumber
AND c.trunkId = a.trunkId
AND c.callInMsg LIKE CONCAT('%', 'do', '%') AND c.callOutMsg LIKE
CONCAT('%', 'no', '%')
and c.tenantId = '880001'
) WHERE
a.tenantId = '880001' ORDER BY
a.outwireNumber ASC LIMIT 0, 15;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ref	<auto_key0>	<auto_key0>	4	const	187	Using where
1	PRIMARY	c	ref	tenantId_outnum_trunkId	tenantId_outnum_trunkId	392	const,a.trunkId,a.outwireNumber	1	Using index condition; Using where
2	DERIVED	departmentId_List	index	PRIMARY	PRIMARY	258	NULL	9364	Using index
2	DERIVED	bms_departmentOutwireNumber	ref	departmentId,tenantId	departmentId	258	DB_IPS_3000.departmentId_List.departmentId	2	Using where

3．删除临时表

```
drop temporary table departmentId_List;
```

优化结果分析：

1. 虽然改写后的查询（1.46秒）比原始SQL（3.8秒）快，考虑到创建临时表的时间，整体速度没有明显的提升；建议暂时不实施。

3来电彩铃查询

场景

查询管理范围相关 “中继号码” 的 “彩铃配置”

- 每个管理员对应多个部门(可达上万个部门)

- 每个部门对应该多个中继号码
- 每个中继号码对就一条来电彩铃配置记录(如果有)
    - 彩铃从VOS的彩铃资源中选择

数据结构

BMS彩铃配置表

```
CREATE TABLE `bms_callerColorBell` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `tenementId` varchar(64) NOT NULL COMMENT '企业ID',
  `tenantId` int(10) unsigned NOT NULL COMMENT 'VOS企业ID',
  `outwireNumber` varchar(32) NOT NULL COMMENT '号码',
  `trunkId` varchar(64) NOT NULL,
  `promptId` varchar(32) NOT NULL COMMENT '提示单(彩铃)ID',
  `inputTime` bigint(13) unsigned NOT NULL DEFAULT '0' COMMENT '导入时间',
  `modifyTime` bigint(13) unsigned NOT NULL DEFAULT '0' COMMENT '最后修改时间',
  PRIMARY KEY (`id`),
  KEY `outwireNumber`(`outwireNumber`),
  KEY `trunkId` (`trunkId`),
  KEY `promptId` (`promptId`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='来电彩铃';
```

VOS彩铃文件资源表

```
CREATE TABLE `IvrPrompt` (
  `PromptID` char(128) NOT NULL COMMENT '保存文件名',
  `TenantID` int(11) NOT NULL DEFAULT '0',
  `OriginalFileName` char(128) NOT NULL COMMENT '原始文件名',
  `PromptName` varchar(128) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '' COMMENT '名称',
  `content` varchar(512) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '' COMMENT '语音内容',
  `LastModifyTimestamp` int(8) NOT NULL COMMENT '最后修改时间戳',
  `AuditResult` int(1) NOT NULL DEFAULT '0' COMMENT '审计结果, 0 未审计 1 未通过 2 通过',
  `RefCount` int(11) NOT NULL DEFAULT '0' COMMENT '引用计数',
  PRIMARY KEY (`PromptID`,`TenantID`),
  UNIQUE KEY `Name` (`TenantID`,`PromptName`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

查询语句

```
# 查询 ${departmentIds} 可达上万个部门
```

```
# PromptName,OriginalFileName,content需要支持模糊搜索

# 查询 ${departmentIds} 可达上万个部门 # PromptName,OriginalFileName,content需要支持模糊搜索

 SELECT
  a.tenementId,
  a.tenantId,
  a.departmentId,
  a.outwireNumber,
  a.trunkId,
  a.inputTime AS outwireInputTime,
  d.OriginalFileName AS originFileName,
  d.PromptName AS promptName,
  d.content AS content,
  d.AuditResult AS auditResult,
  d.RefCount AS refCount,
  d.promptId,
  d.modifyTime
FROM
  (SELECT
      *
  FROM
      bms_departmentOutwireNumber
  WHERE
      departmentId IN (${departmentIds})) a
      INNER JOIN
  (SELECT
      c.tenantId,
          c.outwireNumber,
          c.trunkId,
          e.OriginalFileName,
          e.PromptName,
          e.content,
          e.AuditResult,
          e.RefCount,
          c.promptId,
          c.modifyTime
  FROM
      bms_callerColorBell c
  INNER JOIN IvrPrompt e ON (e.PromptID = c.promptId
      AND e.PromptName LIKE '%''897''%'
      AND e.OriginalFileName LIKE '%''陈奕迅''%'
      AND e.content LIKE '%''达''%')) d ON (d.tenantId = a.tenantId
      AND d.outwireNumber = a.outwireNumber
      AND d.trunkId = a.trunkId)
WHERE
  a.tenantId = '880001'
ORDER BY a.outwireNumber ASC
LIMIT 0 , 15;
```

原始执行计划：

执行速度： 手工产生10000个departmentId，组成In列表值，速度在3-4秒左右。

潜在的问题分析：

1. 数据在服务器和客户端流动多次；
2. In 列表的常量太多；

推荐优化方案（如果数据量变大，出现性能问题时尝试）：

1．将列表 ${departmentIds}存入临时表。下面是例子：

```
create temporary table departmentId_List(primary key
departmentId_idx(departmentId))
select distinct(departmentId) from 表;
```

2．改写SQL，用临时表代替departmentIds列表字符串,推入条件 （1.62秒）。

```
 SELECT
 a.tenementId,
 a.tenantId,a.outwireNumber,
 a.trunkId,
```

```
a.inputTime AS outwireInputTime, d.OriginalFileName AS
originFileName, d.PromptName AS promptName,
d.content AS content, d.AuditResult AS
auditResult, d.RefCount AS refCount, d.promptId,
d.modifyTime FROM
(     SELECT
* FROM
bms_departmentOutwireNumber
force Index(departmentId)
WHERE
departmentId IN(select departmentId from departmentId_List)
and a.tenantId ='880001'
)a
INNER JOIN(
SELECT
c.tenantId, c.outwireNumber, c.trunkId,
e.OriginalFileName, e.PromptName, e.content,
e.AuditResult, e.RefCount, c.promptId,
c.modifyTime
FROM
bms_callerColorBell c INNER JOIN IvrPrompt e
ON(
e.PromptID = c.promptId
and c.tenantId ='880001'
AND e.PromptName LIKE "%" '897' "%"
AND e.OriginalFileName LIKE "%" '陈奕迅' "%"
AND e.content LIKE "%" '达' "%"
)
)d ON(
d.tenantId = a.tenantId
and d.tenantId ='880001'
AND d.outwireNumber =a.outwireNumber
AND d.trunkId= a.trunkId
) WHERE
a.tenantId ='880001' ORDER BY
a.outwireNumber ASC LIMIT 0,15
```

3．删除临时表

```
drop temporary table departmentId_List;
```

优化结果分析：

1. 虽然改写后的查询（1.62秒）比原始SQL（3.25秒）快，考虑到创建临时表的时间，整体速度没有明显的提升；建议暂时不实施。