

## SQL 进阶与实战作业\_week6

Action1:针对自己的业务场景，构建数据表

- (1)、数据表名，相应的字段
- (2)、列出至少 3 个常用的业务查询

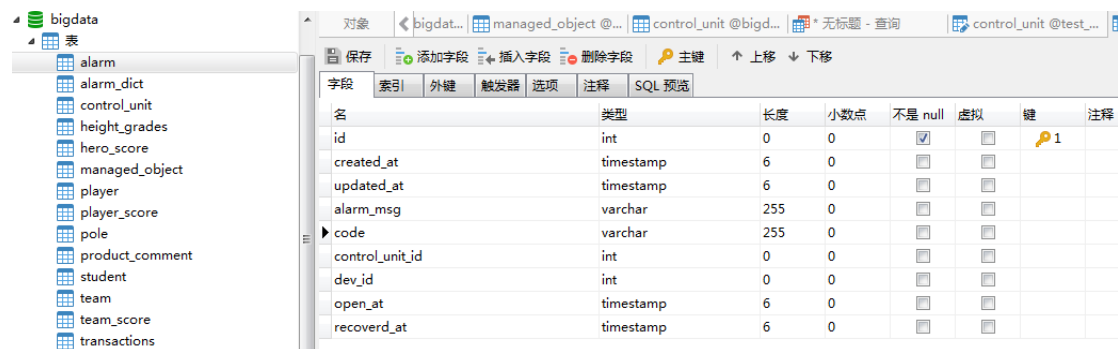
可以是单表，或多表联查，多种统计特征，排名，连续几天出现的情况等

- (3)、说明数据表的索引设计

答：业务属于路灯照明，主要在灯杆中安装了单灯控制器（只能对自身路灯进行控制），在相应配电箱上装有远程监控终端（配电箱，配电箱可以批量开关多条道路的路灯），通过路灯照明平台实现路灯的控制、报警处理、数据采集与故障分析，针对具体业务场景，抽取其中的报警处理部分，建立了四张表，分别为 alarm、alarm\_dict、control\_unit、managed\_object(主表)。

### (1)、一、alarm 表

```
CREATE TABLE `alarm` (`id` int not null, -- 报警 id
`created_at` timestamp(6), -- 创建时间
`updated_at` timestamp(6), -- 更新时间
`alarm_msg` varchar(255), -- 报警信息描述
`code` varchar(255), -- 报警类型编码 (alarm_dict 中有详细描述)
`control_unit_id` int, -- 控制表 id
`dev_id` int, -- 设备 id
`open_at` timestamp(6), -- 报警产生时间
`recoverd_at` timestamp(6), -- 报警恢复时间
PRIMARY KEY (`id`));
```

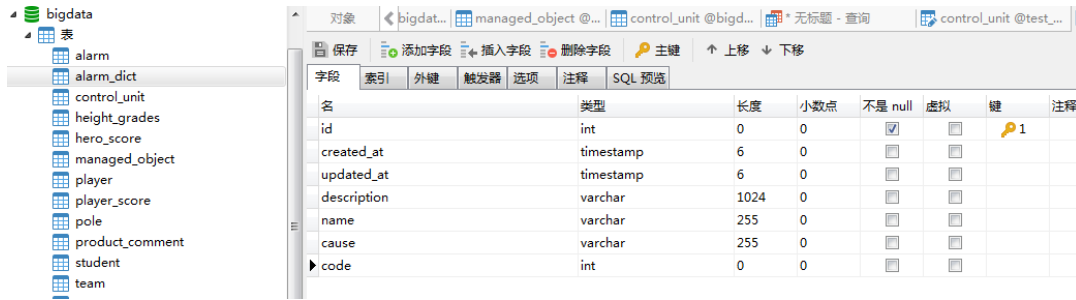


字段	索引	外键	触发器	选项	注释	SQL 预览
名						
id						int 0 0 不是 null 虚拟 键 1
created_at						timestamp 6 0
updated_at						timestamp 6 0
alarm_msg						varchar 255 0
code						varchar 255 0
control_unit_id						int 0 0
dev_id						int 0 0
open_at						timestamp 6 0
recoverd_at						timestamp 6 0

图 1 alarm 表

### 二、alarm\_dict 表

```
CREATE TABLE `alarm_dict` (`id` int not null, -- 报警详情 id
`created_at` timestamp(6), -- 创建时间
`updated_at` timestamp(6), -- 更新时间
`description` varchar(1024), -- 报警名称
`name` varchar(255), -- 报警 name
`cause` varchar(255), -- 报警产生原因
`code` int, -- 报警代码
PRIMARY KEY (`id`));
```

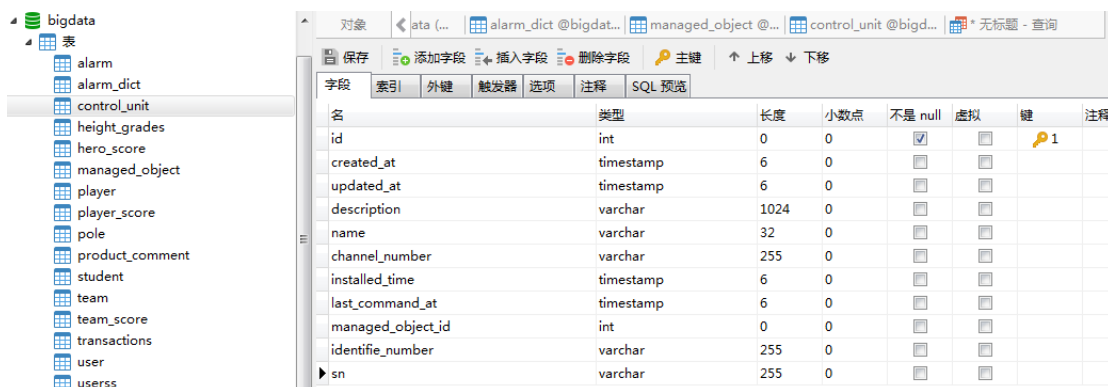


字段	索引	外键	触发器	选项	注释	SQL 预览
名						
id					int	0 0 不是 null 虚拟 键 1
created_at					timestamp	6 0
updated_at					timestamp	6 0
description					varchar	1024 0
name					varchar	255 0
cause					varchar	255 0
code					int	0 0

图 2 alarm\_dict 表

### 三、control\_unit 表

```
CREATE TABLE `control_unit` (`id` int not null, -- id
`created_at` timestamp(6), -- 创建时间
`updated_at` timestamp(6), -- 更新时间
`description` varchar(1024), -- 安装地点信息描述
`name` varchar(32), -- 配电箱名称
`channel_number` varchar(255), -- 创建者-由哪个用户创建
`installed_time` timestamp(6), -- 安装时间
`last_command_at` timestamp(6), -- 命令下发时间
`managed_object_id` int, -- 对应 managed_object 表
`identifie_number` varchar(255), -- SIM 卡识别码
`sn` varchar(255), -- 模组编号
PRIMARY KEY (`id`));
```



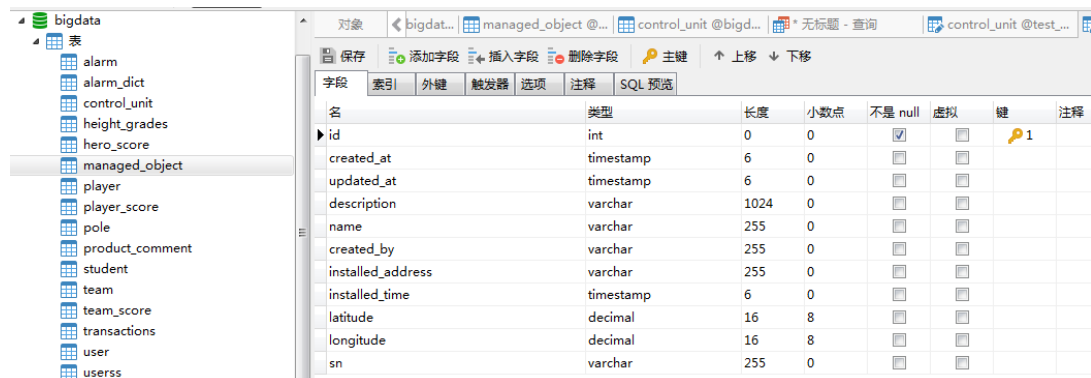
字段	索引	外键	触发器	选项	注释	SQL 预览
名						
id					int	0 0 不是 null 虚拟 键 1
created_at					timestamp	6 0
updated_at					timestamp	6 0
description					varchar	1024 0
name					varchar	32 0
channel_number					varchar	255 0
installed_time					timestamp	6 0
last_command_at					timestamp	6 0
managed_object_id					int	0 0
identifie_number					varchar	255 0
sn					varchar	255 0

图 3 control\_unit 表

### 四、managed\_object 表

```
-- 主表
CREATE TABLE `managed_object` (`id` int not null, -- id
`created_at` timestamp(6), -- 创建时间
`updated_at` timestamp(6), -- 更新时间
`description` varchar(1024), -- 安装地点信息描述
`name` varchar(255), -- 配电箱名称
`created_by` varchar(255), -- 创建者-由哪个用户创建
`installed_address` varchar(255), -- 设备安装地址
`installed_time` timestamp(6), -- 设备安装时间
`latitude` decimal(16,8), -- 安装位置的纬度
```

```
`longitude` decimal(16,8), -- 安装位置的经度
`sn` int, -- 通讯SIM卡卡号
PRIMARY KEY (`id`));
```



名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
created_at	timestamp	6	0	<input type="checkbox"/>	<input type="checkbox"/>		
updated_at	timestamp	6	0	<input type="checkbox"/>	<input type="checkbox"/>		
description	varchar	1024	0	<input type="checkbox"/>	<input type="checkbox"/>		
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		
created_by	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		
installed_address	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		
installed_time	timestamp	6	0	<input type="checkbox"/>	<input type="checkbox"/>		
latitude	decimal	16	8	<input type="checkbox"/>	<input type="checkbox"/>		
longitude	decimal	16	8	<input type="checkbox"/>	<input type="checkbox"/>		
sn	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		

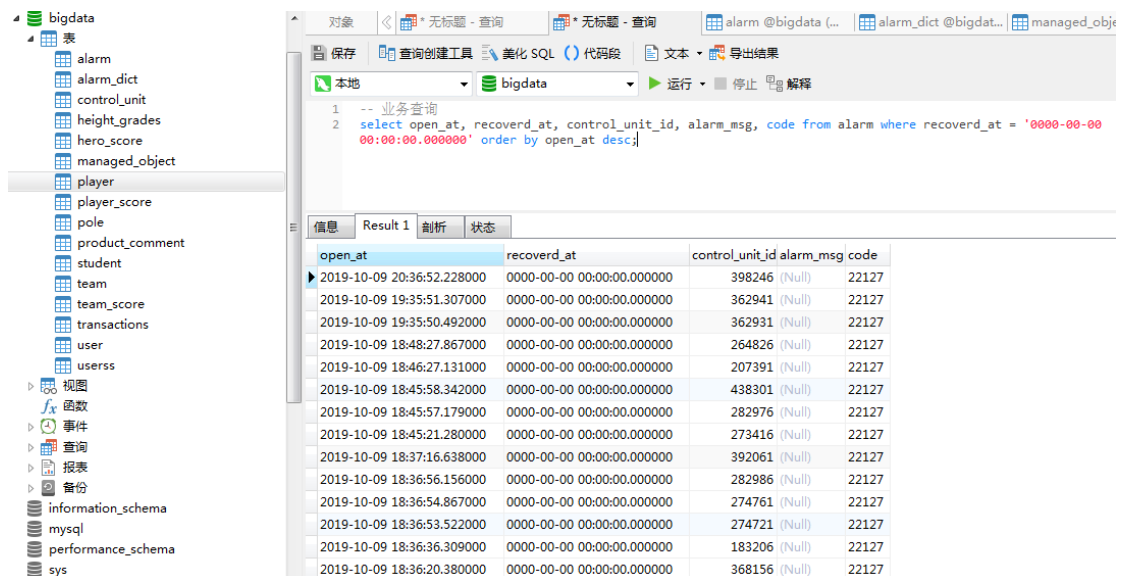
图 4 managed\_object 表

## (2)、业务查询

由于是根据实际项目产生的报警信息，直接利用了数据库中数据进行查询任务。

1、查询 alarm 表中 recoverd\_at 字段为 '0000-00-00 00:00:00.000000'（即报警未修复）的，并根据 open\_at（报警产生时间）字段的时间从大到小排列，查询结果由 open\_at、recoverd\_at、control\_unit\_id、alarm\_msg 和 code 字段组成。

```
select open_at, recoverd_at, control_unit_id, alarm_msg, code from alarm where
recoverd_at = '0000-00-00 00:00:00.000000' order by open_at desc;
```

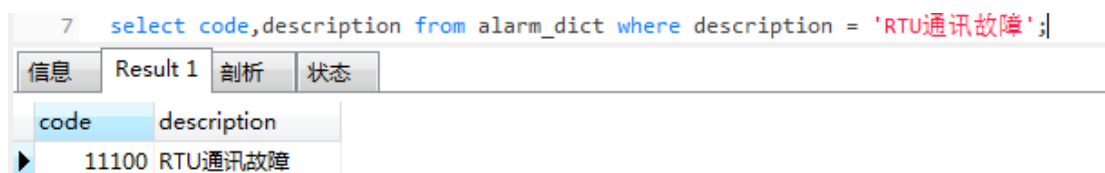


open_at	recoverd_at	control_unit_id	alarm_msg	code
2019-10-09 20:36:52.228000	0000-00-00 00:00:00.000000	398246	(Null)	22127
2019-10-09 19:35:51.307000	0000-00-00 00:00:00.000000	362941	(Null)	22127
2019-10-09 19:35:50.492000	0000-00-00 00:00:00.000000	362931	(Null)	22127
2019-10-09 18:48:27.867000	0000-00-00 00:00:00.000000	264826	(Null)	22127
2019-10-09 18:46:27.131000	0000-00-00 00:00:00.000000	207391	(Null)	22127
2019-10-09 18:45:58.342000	0000-00-00 00:00:00.000000	438301	(Null)	22127
2019-10-09 18:45:57.179000	0000-00-00 00:00:00.000000	282976	(Null)	22127
2019-10-09 18:45:21.280000	0000-00-00 00:00:00.000000	273416	(Null)	22127
2019-10-09 18:37:16.638000	0000-00-00 00:00:00.000000	392061	(Null)	22127
2019-10-09 18:36:56.156000	0000-00-00 00:00:00.000000	282986	(Null)	22127
2019-10-09 18:36:54.867000	0000-00-00 00:00:00.000000	274761	(Null)	22127
2019-10-09 18:36:53.522000	0000-00-00 00:00:00.000000	274721	(Null)	22127
2019-10-09 18:36:36.309000	0000-00-00 00:00:00.000000	183206	(Null)	22127
2019-10-09 18:36:20.380000	0000-00-00 00:00:00.000000	368156	(Null)	22127

图 5 查询-alarm

2、从 alarm\_dict 中查询 'RTU 通讯故障' 对应的 code;

```
select code,description from alarm_dict where description = 'RTU 通讯故障';
```



code	description
11100	RTU通讯故障

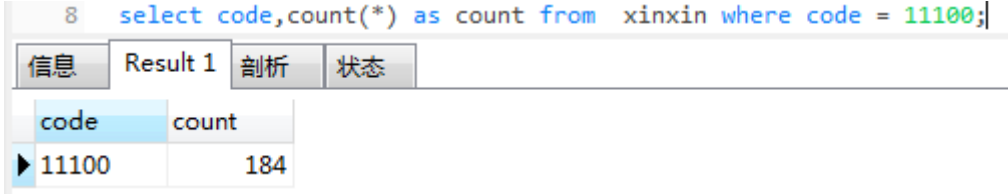
图 6 查询-alarm\_dict

将 1 中的查询结果生成一个新的 table:

```
create table xinxin (select open_at, recoverd_at, control_unit_id, alarm_msg, code
from alarm where recoverd_at = '0000-00-00 00:00:00.000000' order by open_at desc);
```

从新生成的表中统计' RTU 通讯故障' 出现的次数:

```
select code,count(*) as count from xinxin where code = 11100;
```



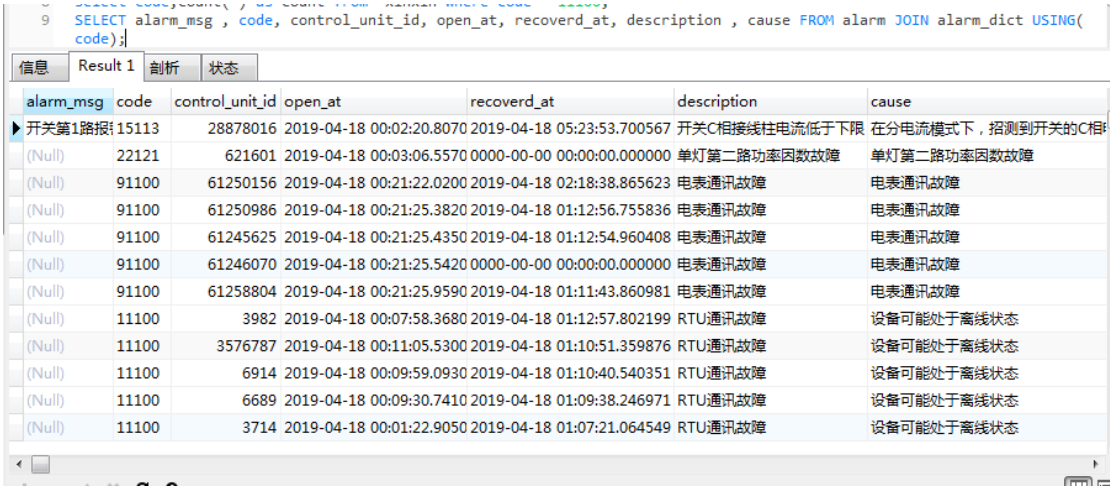
8 select code,count(*) as count from xinxin where code = 11100;	
信息	Result 1
剖析	状态
code	count
11100	184

图 7 count of code

由于 alarm 表中报警信息不够详细, 通过与 alarm\_dict 进行两表连接, 查询结果便于维护人员更好的掌握设备现状, 更好的解决故障。

选取 alarm\_msg , code, control\_unit\_id, open\_at, recoverd\_at, description , cause 字段, 由于两张表中存在相同字段 code, 通过 USING 进行等值连接:

```
SELECT alarm_msg , code, control_unit_id, open_at, recoverd_at, description , cause
FROM alarm JOIN alarm_dict USING(code);
```



9 SELECT alarm_msg , code, control_unit_id, open_at, recoverd_at, description , cause FROM alarm JOIN alarm_dict USING( code);						
信息	Result 1	剖析	状态			
alarm_msg	code	control_unit_id	open_at	recoverd_at	description	cause
开关第1路报	15113	28878016	2019-04-18 00:02:20.8070	2019-04-18 05:23:53.700567	开关C相接线柱电流低于下限	在分电流模式下, 检测到开关的C相
(Null)	22121	621601	2019-04-18 00:03:06.5570	0000-00-00 00:00:00.000000	单灯第二路功率因数故障	单灯第二路功率因数故障
(Null)	91100	61250156	2019-04-18 00:21:22.0200	2019-04-18 02:18:38.865623	电表通讯故障	电表通讯故障
(Null)	91100	61250986	2019-04-18 00:21:25.3820	2019-04-18 01:12:56.755836	电表通讯故障	电表通讯故障
(Null)	91100	61245625	2019-04-18 00:21:25.4350	2019-04-18 01:12:54.960408	电表通讯故障	电表通讯故障
(Null)	91100	61246070	2019-04-18 00:21:25.5420	0000-00-00 00:00:00.000000	电表通讯故障	电表通讯故障
(Null)	91100	61258804	2019-04-18 00:21:25.9590	2019-04-18 01:11:43.860981	电表通讯故障	电表通讯故障
(Null)	11100	3982	2019-04-18 00:07:58.3680	2019-04-18 01:12:57.802199	RTU通讯故障	设备可能处于离线状态
(Null)	11100	3576787	2019-04-18 00:11:05.5300	2019-04-18 01:10:51.359876	RTU通讯故障	设备可能处于离线状态
(Null)	11100	6914	2019-04-18 00:09:59.0930	2019-04-18 01:10:40.540351	RTU通讯故障	设备可能处于离线状态
(Null)	11100	6689	2019-04-18 00:09:30.7410	2019-04-18 01:09:38.246971	RTU通讯故障	设备可能处于离线状态
(Null)	11100	3714	2019-04-18 00:01:22.9050	2019-04-18 01:07:21.064549	RTU通讯故障	设备可能处于离线状态

图 8 USING code

筛选出报警未回复的部分, 即 recoverd\_at = '0000-00-00 00:00:00.000000', 根据 open\_at 进行排序:

```
SELECT alarm_msg , code, control_unit_id, open_at, recoverd_at, description , cause
FROM alarm JOIN alarm_dict USING(code) where recoverd_at = '0000-00-00
00:00:00.000000' order by open_at desc;
```

10	SELECT alarm_msg , code, control_unit_id, open_at, recoverd_at, description , cause FROM alarm JOIN alarm_dict USING( code) where recoverd_at = '0000-00-00 00:00:00.000000' order by open_at desc;					
信息	Result 1	剖析	状态			
alarm_msg	code	control_unit_id	open_at	recoverd_at	description	cause
(Null)	22127	398246	2019-10-09 20:36:52.2280 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	362941	2019-10-09 19:35:51.3070 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	362931	2019-10-09 19:35:50.4920 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	264826	2019-10-09 18:48:27.8670 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	207391	2019-10-09 18:46:27.1310 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	438301	2019-10-09 18:45:58.3420 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	282976	2019-10-09 18:45:57.1790 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	273416	2019-10-09 18:45:21.2800 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	392061	2019-10-09 18:37:16.6380 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	282986	2019-10-09 18:36:56.1560 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	274761	2019-10-09 18:36:54.8670 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障
(Null)	22127	274721	2019-10-09 18:36:53.5220 0000-00-00 00:00:00.000000		单灯第二路低电压故障	单灯第二路低电压故障

图 9 故障排序

有些设备可能存在多个故障，可以通过选取 control\_unit\_id 来进行排序，可以发现有多故障的点位：

SELECT alarm\_msg , code, control\_unit\_id, open\_at, recoverd\_at, description , cause  
FROM alarm JOIN alarm\_dict USING(code) where recoverd\_at = '0000-00-00  
00:00:00.000000' order by control\_unit\_id desc;

11	SELECT alarm_msg , code, control_unit_id, open_at, recoverd_at, description , cause FROM alarm JOIN alarm_dict USING( code) where recoverd_at = '0000-00-00 00:00:00.000000' order by control_unit_id desc;					
信息	Result 1	剖析	状态			
alarm_msg	code	control_unit_id	open_at	recoverd_at	description	cause
(Null)	11100	304235884	2019-06-19 17:00:39.9200 0000-00-00 00:00:00.000000		RTU通讯故障	设备可能处于离线状态
配电箱报警	12109	304235884	2019-05-14 11:20:46.3210 0000-00-00 00:00:00.000000		配电箱断电	三相电压均小于电压报警阈值
(Null)	22100	300937731	2019-05-31 10:04:51.6780 0000-00-00 00:00:00.000000		单灯通讯故障	单灯控制器通讯故障
(Null)	22100	300847405	2019-05-31 10:04:47.6970 0000-00-00 00:00:00.000000		单灯通讯故障	单灯控制器通讯故障
配电箱报警	12109	298367212	2019-06-12 17:00:40.9370 0000-00-00 00:00:00.000000		配电箱断电	三相电压均小于电压报警阈值
配电箱报警	12109	298366854	2019-06-19 17:00:08.7570 0000-00-00 00:00:00.000000		配电箱断电	二相电压均小于电压报警阈值
(Null)	11100	298366854	2019-06-20 03:17:40.2790 0000-00-00 00:00:00.000000		RTU通讯故障	设备可能处于离线状态
配电箱报警	12109	298364517	2019-06-19 17:00:38.8840 0000-00-00 00:00:00.000000		配电箱断电	三相电压均小于电压报警阈值
(Null)	11100	298364517	2019-06-20 03:50:40.5350 0000-00-00 00:00:00.000000		RTU通讯故障	设备可能处于离线状态
配电箱报警	12109	298364075	2019-06-19 17:00:42.3210 0000-00-00 00:00:00.000000		配电箱断电	三相电压均小于电压报警阈值
(Null)	11100	298364075	2019-06-20 03:39:40.4220 0000-00-00 00:00:00.000000		RTU通讯故障	设备可能处于离线状态
配电箱报警	12109	298361739	2019-06-19 17:00:45.3850 0000-00-00 00:00:00.000000		配电箱断电	三相电压均小于电压报警阈值

图 10 点位故障统计

create table xinguang (SELECT alarm\_msg , code, control\_unit\_id, open\_at, recoverd\_at, description , cause FROM alarm JOIN alarm\_dict USING(code) where recoverd\_at = '0000-00-00 00:00:00.000000' order by control\_unit\_id desc);

把查询结果创建一张新表存储。

3、虽然上面的业务查询得出了部分结果，但是并没有把点位信息查询出来，维护人员无法确定点位信息，无法到现场进行设备维护，所以需要结合 control\_unit 和 managed\_object 表来获取更详细的查询信息。control\_unit 表中的 control\_unit\_id 可以对应一个 managed\_object\_id,根据这个 managed\_object\_id 可以在 managed\_object 表中找到对应的设备安装及位置等信息了：

select managed\_object.updated\_at,control\_unit.id, control\_unit.managed\_object\_id, managed\_object.description,managed\_object.name,managed\_object.installed\_address ,control\_unit.sn from managed\_object left join control\_unit on managed\_object.id = control\_unit.managed\_object\_id where managed\_object.updated\_at BETWEEN '2020-04-01 00:00:00.000000' AND '2020-04-30 00:00:00.000000' order by managed\_object.updated\_at asc;

选取 managed\_object 的 updated\_at、description、name、installed\_address 字段，control\_unit 的 id、managed\_object\_id、sn 字段，并按照 updated\_at 进行升序排列（选取了 2020 年的 4 月 1 日到 4 月 30 日）。

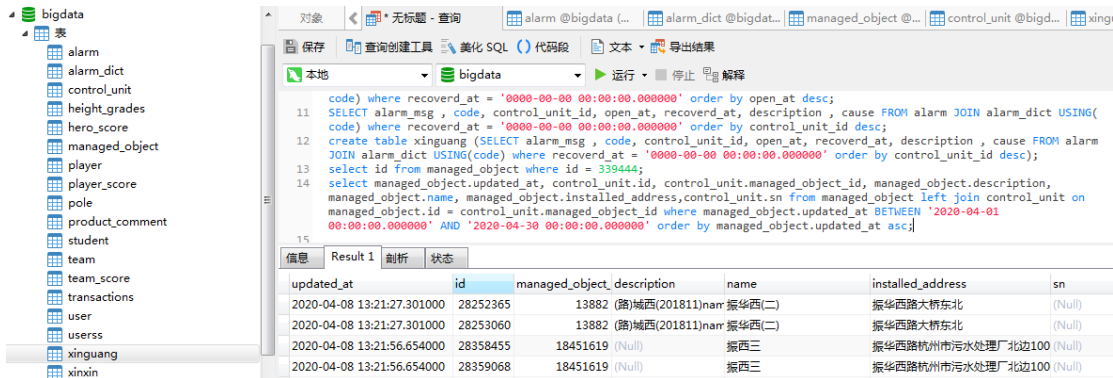
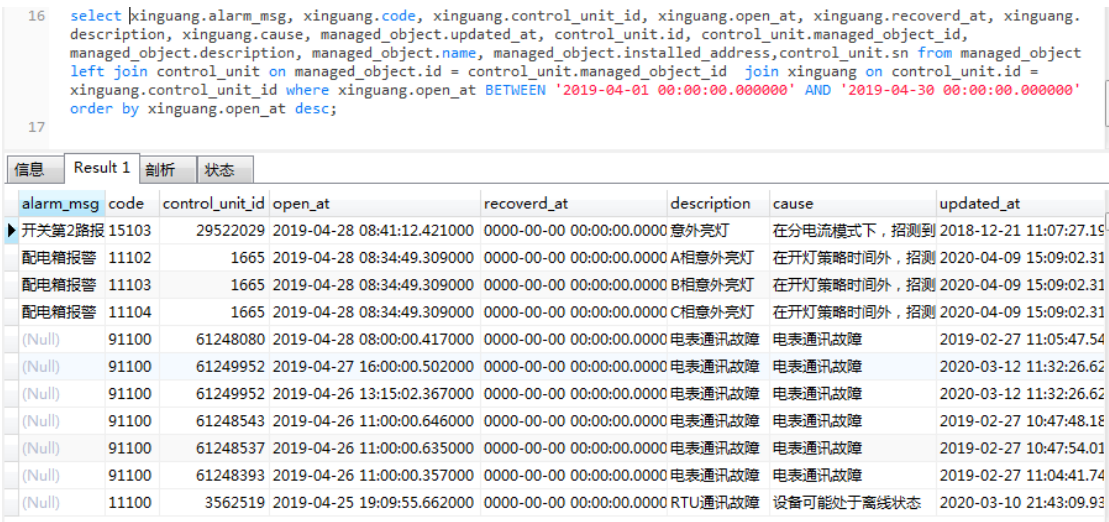


图 11 managed\_object 与 control\_unit 联查

为了更好的查看查询结果，结合 2 中生成的 xinguang 表，将 managed\_object、control\_unit 和 xinguang 三个表进行三表联查，并根据报警产生时间 open\_at，筛选出 2019 年 4 月份的进行降序排列：

```
select
xinguang. alarm_msg, xinguang. code, xinguang. control_unit_id, xinguang. open_at,
xinguang. recoverd_at, xinguang. description, xinguang. cause, managed_object. updated
_at, control_unit. id, control_unit. managed_object_id, managed_object. description, m
anaged_object. name, managed_object. installed_address, control_unit. sn
from
managed_object left join control_unit on managed_object.id =
control_unit.managed_object_id join xinguang on control_unit.id =
xinguang.control_unit_id where xinguang.open_at BETWEEN '2019-04-01
00:00:00.000000' AND '2019-04-30 00:00:00.000000' order by xinguang.open_at desc;
```





```

16 select xinguang.alarm_msg, xinguang.code, xinguang.control_unit_id, xinguang.open_at, xinguang.recoverd_at, xinguang.
description, xinguang.cause, managed_object.updated_at, control_unit.id, control_unit.managed_object_id,
managed_object.description, managed_object.name, managed_object.installed_address, control_unit.sn from managed_object
left join control_unit on managed_object.id = control_unit.managed_object_id join xinguang on control_unit.id =
xinguang.control_unit_id where xinguang.open_at BETWEEN '2019-04-01 00:00:00.000000' AND '2019-04-30 00:00:00.000000'
order by xinguang.open_at desc;
17

```

updated_at	id	managed_object_id	description(1)	name	installed_address	sn
2018-12-21 11:07:27.1990	29522029	9961	(路)城北(20181: 丰都	东风港路, 华鹤街	(Null)	
2020-04-09 15:09:02.3150	1665	1664	西湖景区养护	三潭印月3号	三潭印月西北角	13454475904
2020-04-09 15:09:02.3150	1665	1664	西湖景区养护	三潭印月3号	三潭印月西北角	13454475904
2020-04-09 15:09:02.3150	1665	1664	西湖景区养护	三潭印月3号	三潭印月西北角	13454475904
2019-02-27 11:05:47.5490	61248080	61248079	(Null)	之梦	留泗路090号灯杆座	865352031788549
2020-03-12 11:32:26.6250	61249952	61249951	(Null)	梅竺(三)	德胜路红普路高架#	865352031799843
2020-03-12 11:32:26.6250	61249952	61249951	(Null)	梅竺(三)	德胜路红普路高架#	865352031799843
2019-02-27 10:47:48.1860	61248543	61248542	(Null)	石祥高架(四)	石祥路与上塘路交	865352031799090
2019-02-27 10:47:54.0130	61248537	61248536	(Null)	石上	石祥路上塘高架交	865352031762601
2019-02-27 11:04:41.7440	61248393	61248392	(Null)	丽紫开关	丽水路与大浒路交	865352031760266
2020-03-10 21:43:09.9350	3562519	3562517	市管一标养护(20	城东桥东北角	城东桥东北角 (机	15024473130

图 12 managed\_object、control\_unit 与 xinguang 三表联查

### (3)、数据表索引设计

一、managed\_object 中 sn、latitude 和 longitude 设置为索引，可以提高查询效率，如：  
UPDATE managed\_object SET name = '120720 新增百田 022\_0'  
WHERE sn = '17816854562';

```

17 UPDATE managed_object SET name = '120720新增百田022_0'
18 WHERE sn = '17816854562';
19 select id from managed_object where id = 980321;
20

```

信息	剖析	状态
UPDATE managed_object SET name = '120720新增百田022_0' WHERE sn = '17816854562' > Affected rows: 3 > 时间: 0.085s		

图 13 未加索引

```

17 UPDATE managed_object SET name = '120720新增百田022_0'
18 WHERE sn = '17816854562';
19 select id from managed_object where id = 980321;
20

```

信息	剖析	状态
UPDATE managed_object SET name = '120720新增百田022_0' WHERE sn = '17816854562' > Affected rows: 3 > 时间: 0.003s		

对象	索引	外键	触发器	选项	注释	SQL 预览
name	字段					
sn	字段					
					索引类型	索引方法
					NORMAL	BTREE

图 14 添加索引后

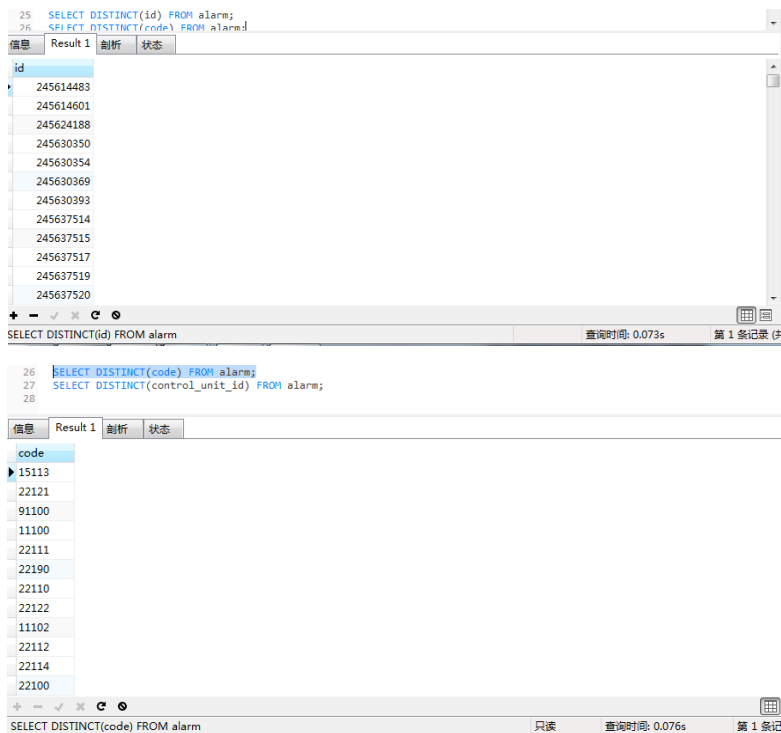


图 15 latitude 和 longitude 添加索引前后

通过运行时间可以看出，添加索引后，时间明显变少了。

## 二、alarm 表

添加 code、id 和 control\_unit\_id 索引：





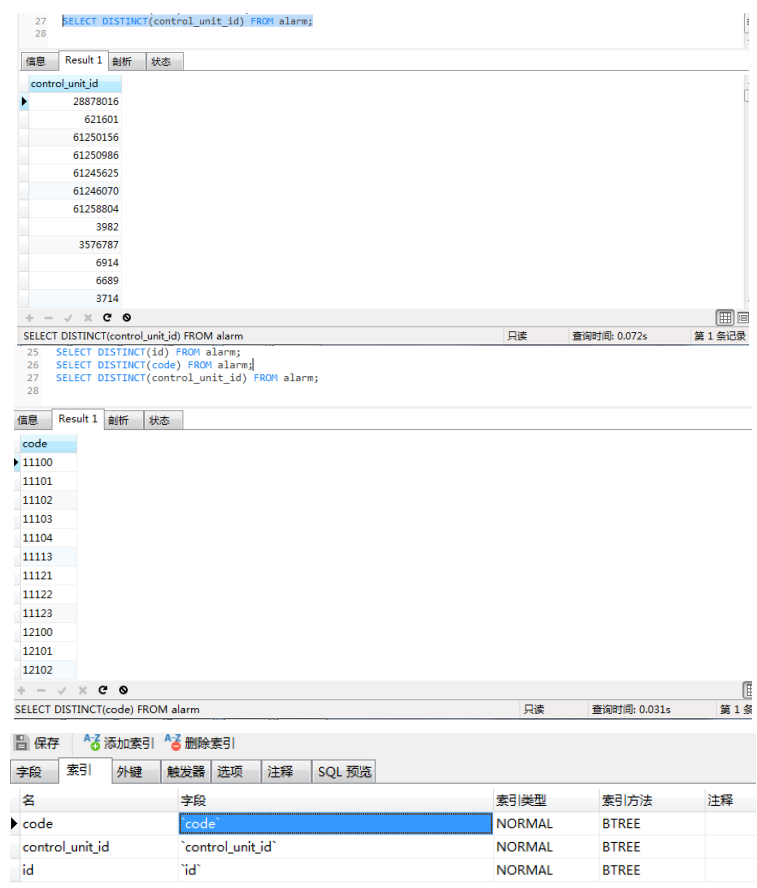
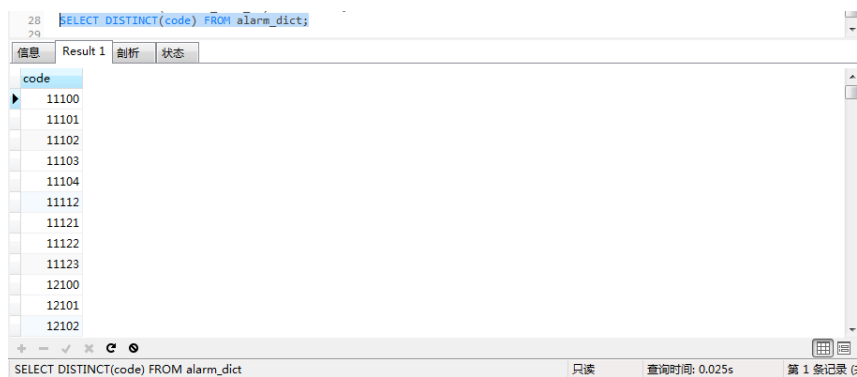


图 16 添加索引前后

alarm 表中经常需要查询报警 code 和对应 control\_unit\_id, 以及自身 id, 由于数据量不是很庞大, 效果不算显著, code 由之前的 0.076 变为 0.031, 效果还是有的。

### 三、alarm\_dict 表

由于此表中数据只有一百多个, 添加 code 索引, 效果不是很明显:



```

28 SELECT DISTINCT(code) FROM alarm_dict;
29

```

code
11100
11101
11102
11103
11104
11111
11112
11113
11121
11122
11123
11140

SELECT DISTINCT(code) FROM alarm\_dict 只读 查询时间: 0.024s 第 1 条

图 17 code 索引

根据上面查询时间可看，alarm\_dict 表中索引 code 可加可不加。

#### 四、control\_unit 表

```

28 SELECT DISTINCT(managed_object_id) FROM control_unit;
29

```

managed_object_id
209
227
427
488
553
559
587
693
751
830
1000
1040

SELECT DISTINCT(managed\_object\_id) FROM control\_unit 只读 查询时间: 0.110s 第 1 条

```

27 SELECT DISTINCT(id) FROM control_unit;
28 SELECT DISTINCT(managed_object_id) FROM control_unit;
29

```

id
210
228
428
489
554
560
588
694
752
831
1001
1041

SELECT DISTINCT(id) FROM control\_unit 查询时间: 0.126s 第 1 条记录 (共 12 条)

```

28 SELECT DISTINCT(managed_object_id) FROM control_unit;
29

```

managed_object_id
(Null)
5
11
19
25
31
37
43
49
55
61
68

SELECT DISTINCT(managed\_object\_id) FROM control\_unit 只读 查询时间: 0.067s 第 1 条

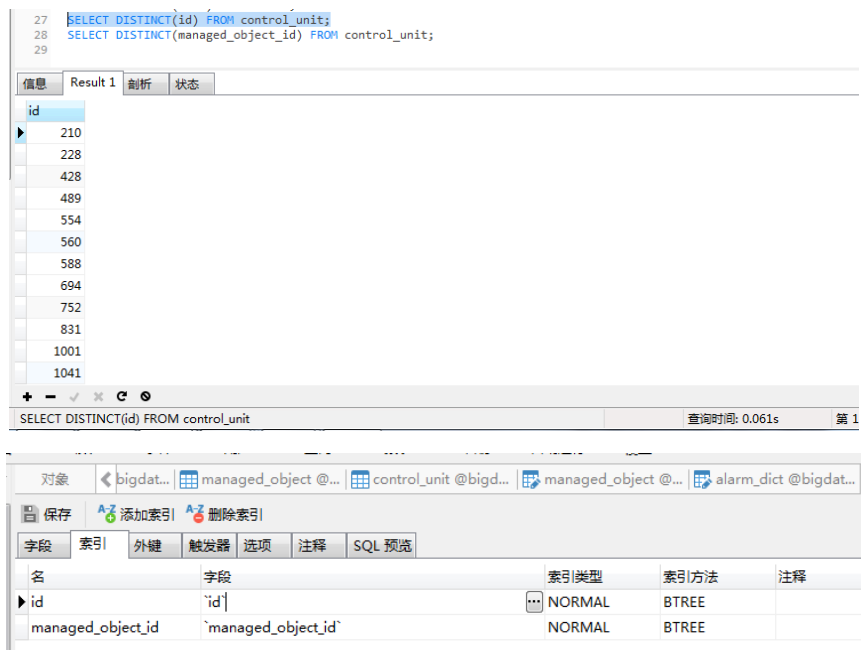


图 18 id 和 control\_unit\_id 索引

通过上面学习可以看出，需要经常进行查询和唯一标识的字段，需要通过添加索引来提高查询效率，更好更快的获取数据，当然，数据比较少的时候，可以不用添加索引。

本节课作业很好的学习了如何进行建表、数据查询、排序、多表联查、索引设计等相关内容，更好的理解了数据表的使用，使我在以后的学习中可以更好的学好新知识，利用好 SQL。