# \*\*\*\*表号和编号 在各个规则中是唯一的。换而言之，只能在一个规则中。

# 上下限判读规则

## 表名和表结构说明

##### 上下限 rule\_interval

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ---------------- | -------- | ---- |

| id | 主键（自动递增） | int | 11 |

| task\_id | 任务号 | varchar | 64 |

| table\_number | 参数表号 | int | 11 |

| param\_id | 参数主键 | int | 11 |

| param\_name | 参数名 | varchar | 64 |

| param\_nickname | 参数代号 | varchar | 64 |

| min | 最小值 | double | 16,4 |

| max | 最大值 | double | 16,4 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 数据结构

typedef struct st\_UpLowDataRule\_

{

float min;//最小值

float max;//最大值

bool flag;//标志，true进行判读，false不判读

st\_UpLowDataRule\_()

{

min = 0.0;

max = 0.0;

flag = true;

}

}st\_UpLowDataRule;

//map关键字为 table\_number参数表表号 + param\_id参数表主键

//eg table\_number=1081 ; param\_id = 1 key = 18011

typedef QMap<int, st\_UpLowDataRule> st\_UpLowDataRule\_Map;

## 算法说明

1. 获取表rule\_interval的所有记录
2. 每次数据到来，根据Key匹配，判读数据
3. if(realData > maxData || realData < minData)

return false

Else

Return ture

4,如果判读错误 存储数据库 发送到web服务端。

# 差值判读

## 表名和表结构说明

##### 差值 rule\_diff

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ---------------- | -------- | ---- |

| id | 主键（自动递增） | int | 11 |

| task\_id | 任务号 | varchar | 64 |

| table\_number | 参数表号 | int | 11 |

| param\_id | 参数主键 | int | 11 |

| param\_name | 参数名 | varchar | 64 |

| param\_nickname | 参数代号 | varchar | 64 |

| min | 最小值 | double | 16,4 |

| max | 最大值 | double | 16,4 |

| interval | 间隔 | int | 11 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 数据结构

//##### 差值规则表对应的数据结构

typedef struct st\_DiffDataRule\_

{

int interval;//间隔

float min;//最小值

float max;//最大值

bool flag;//标志，true进行判读，false不判读

st\_DiffDataRule\_()

{

interval =0;

min = 0.0;

max = 0.0;

flag = true;

}

}st\_DiffDataRule;

//map关键字为 table\_number参数表表号 + param\_id参数表主键

//eg table\_number=1081 ; param\_id = 1 key = 18011

typedef QMap<int, st\_DiffDataRule> st\_DiffDataRule\_Map;

## 算法说明

1. 获取表rule\_diff的所有记录
2. 每次数据到来，根据Key匹配，判读数据dy
3. 记录每次上报的值，上一次上报的值 与 下一次上报值 之间做差值判读

d\_value = d2.data - d1.data;

if((d\_value <= dy.para\_max) && (dy.para\_min <= d\_value))

Return ture

Else

Return false

4,如果判读错误 存储数据库 发送到web服务端。

5，需要维护一个map，存放上一次记录的值。（如果记录为新记录，插入。如果为已有的，则更新。以便下次作为判读依据）

备注： 上一次上报的值 与 下一次上报值 之间做差值判读

# 分时分区间的规则表

## 表结构说明

##### 分时间区间（主表） rule\_time

| 列名 | 含义 | 类型 | 长度 |

| ------------------ | ---------------------- | -------- | ------ |

| id | 主键（自动递增） | int | 11 |

| task\_id | 任务号 | varchar | 64 |

| start\_table\_number | 开始参数表号 | int | 11 |

| start\_param\_id | 开始参数主键 | int | 11 |

| start\_param\_name | 开始参数名 | varchar | 64 |

| end\_table\_number | 结束参数表号 | int | 11 |

| end\_param\_id | 结束参数主键 | int | 11 |

| end\_param\_name | 结束参数名 | varchar | 64 |

| ~~detail\_id~~ | ~~详情ID~~ | ~~int~~ | ~~11~~ |

| type | 类型：0 差值，1 上下限 | tinyint | 1 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 分时间分区间（上下限）规则

##### 分时间区间（上下限）rule\_time\_interval

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ---------------- | -------- | ---- |

| id | 主键（自动递增） | int | 11 |

| rule\_time\_id | 分时间区间主表ID | int | 11 |

| table\_number | 判读参数表号 | int | 11 |

| param\_id | 判读参数主键 | int | 11 |

| param\_name | 判读参数名 | varchar | 64 |

| param\_nickname | 判读参数代号 | varchar | 64 |

| min | 最小值 | double | 16,4 |

| max | 最大值 | double | 16,4 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 分时间分区间（差值）规则

##### 分时间区间（差值）rule\_time\_diff

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ---------------------------------- | -------- | ---- |

| id | 主键（参数表表号+主键拼接后转int） | int | 11 |

| rule\_time\_id | 分时间区间主表ID | int | 11 |

| table\_number | 判读参数表号 | int | 11 |

| param\_id | 判读参数主键 | int | 11 |

| param\_name | 判读参数名 | varchar | 64 |

| param\_nickname | 判读参数代号 | varchar | 64 |

| min | 最小值 | double | 16,4 |

| max | 最大值 | double | 16,4 |

| interval | 间隔 | int | 11 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 算法说明

//map关键之为开始编号 或者 结束编号 value 为关联的编号

typedef QMap<int, int> st\_tableParam\_Map;

### 上下限：

输出1， st\_tableParam\_Map开始map

输出2 st\_tableParam\_Map结束map

输出3 st\_UpLowDataRule\_Map上下限 初始值：flag = false

1, 根据表rule\_time 和 表rule\_time\_interval 查询，

输出1 st\_tableParam\_Map开始map

输出2 st\_tableParam\_Map结束map

1. 根据表rule\_time\_interval查询出所有记录

输出3 st\_UpLowDataRule\_Map上下限 初始值：flag = false

1. 根据收到的记录，在st\_tableParam\_Map，st\_tableParam\_Map。中查询。

如果st\_tableParam\_Map开始map中存在，则致st\_UpLowDataRule\_Map中关联的数据为true.

如果st\_tableParam\_Map结束map中存在，则致st\_UpLowDataRule\_Map中关联的数据为false.

4,剩下的就是和上下限判读一样的操作。

### 差值

输出1， st\_tableParam\_Map开始map

输出2 st\_tableParam\_Map结束map

输出3 st\_DiffDataRule\_Map差值 初始值：flag = false

1, 根据表rule\_time 和 表rule\_time\_interval 查询，

输出1 st\_tableParam\_Map开始map

输出2 st\_tableParam\_Map结束map

1. 根据表rule\_time\_interval查询出所有记录

输出3 st\_DiffDataRule\_Map差值 初始值：flag = false

1. 根据收到的记录，在st\_tableParam\_Map，st\_tableParam\_Map。中查询。

如果st\_tableParam\_Map开始map中存在，则致st\_DiffDataRule\_Map中关联的数据为true.

如果st\_tableParam\_Map结束map中存在，则致st\_DiffDataRule\_Map中关联的数据为false.

4,剩下的就是和差值判读一样的操作。

# 飞行时序规则

## ##### 飞行时序主表 rule\_sequence

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ------------------------ | -------- | ---- |

| id | 主键（自动递增） | int | 11 |

| task\_id | 任务号 | varchar | 64 |

| table\_number | 参数表表号 | int | 11 |

| param\_id | 参数表主键 | int | 11 |

| param\_name | 参数名 | varchar | 64 |

| param\_nickname | 参数代号 | varchar | 64 |

| sort | 参数出现先后顺序 | int | 11 |

| type | 参数类型：0 无线，1 有线 | tinyint | 1 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

//1,根据参数类型分类，2，根据参数出现先后顺序排序

## ##### 飞行时序详情表 rule\_sequence\_detail

| 列名 | 含义 | 类型 | 长度 |

| -------------- | ---------------- | -------- | ---- |

| id | 主键（自动递增） | int | 11 |

| sequence\_id | 飞行时序主表ID | int | 11 |

| table\_number | 参数表表号 | int | 11 |

| param\_id | 参数表主键 | int | 11 |

| param\_name | 参数名 | varchar | 64 |

| param\_nickname | 参数代号 | varchar | 64 |

| theory | 理论值 | double | 16,4 |

| error | 误差范围 | double | 16,4 |

| remark | 备注 | text | 0 |

| create\_time | 创建时间 | datetime | 0 |

## 数据结构

//##### 飞行时序规则表对应的数据结构

typedef struct st\_SequenceDataRule\_

{

int interval;//间隔

float theory;//理论值

float error;//误差范围

st\_SequenceDataRule\_()

{

interval =0;

theory = 0.0;

error = 0.0;

}

}st\_SequenceDataRule;

Q\_DECLARE\_METATYPE(st\_SequenceDataRule)//供一个默认的拷贝构造函数

//map关键字为 table\_number参数表表号 + param\_id参数表主键

//eg table\_number=1081 ; param\_id = 1 key = 18011

typedef QVector<int, st\_SequenceDataRule> st\_SequenceDataRule\_Vector;

//第一个参数为，主点编号

typedef QVector<int,st\_SequenceDataRule\_Vector> st\_SequenceDataRuleAll\_Vector;

## 算法说明

以“无线”为例子。

1, 读取主表rule\_sequence，根据参数类型分类筛选出无线飞行时序记录，

2，根据参数出现先后顺序排序，

1. 根据主表ID从rule\_sequence\_detail表中，获取主点的从点表。
2. 初始化当前参数信息为主表第一节点。
3. 当收到的“表号+编号”为当前主点，更新时间信息，作为计算标准。

以顺序遍历完从表，再遍历主表。

无论有线、无线测量参数，均按:

偏差值=实测值－理论值进行计算