# 需求

1. 数据是否可以删除需要在前端控制，需要根据数据是否被引用来判断是否可以删除。
2. 数据被引用需要支持指定的业务字段不可以编辑。
3. 数据是否可以被编辑和删除需要支持通过数据安全进行配置，启用了数据安全的删除、修改控制，需要根据数据是否被引用来控制是否可以删除或修改。
4. 数据删除需要支持回收站功能，删除的数据不支持只根据ID删除，需要传入完整的数据进行删除。

# 解决方案

**存在问题：**

**第一次编辑或未被引用的数据，前端做删除和编辑的时候需要查询一下数据是否被引用，才可以控制数据是否允许被修改和删除。**

1. 采用查询的时候判断数据是否被引用，未被引用的数据查询一下数据库，该方法性能最差。查询多数据的时候需要逐行进行判断是否被引用未被，未被引用需要去查询该记录是否被引用。

**优点：**只需要在查询时更新数据是否被引用。

**缺点：**严重影响性能。

1. 采用前端执行修改或删除数据的时候判断对象是否存在数据标记字段，如果存在并且值不为1则异步查询数据库获取对应的引用关系，并且更新对应得引用关系。在前端根据对应的值和数据安全开关控制数据是否可以被编辑和修改。

**优点：**性能佳，用户体验好。

**缺点：**前端逻辑复杂，需要获取数据安全状态和引用状态等控制数据是否可以操作。

1. 采用后端判断数据是否可以修改和删除，修改的时候后端校验是否有只读字段，如果存在只读字段并且数据被引用则抛出异常。

**优点：**性能好，代码逻辑相对简单

**缺点：**用户体验不好。

**结论：**方案2和3的性能差别不大，采用方案2的方式实现，目前的设计只在前端进行控制，后端逻辑不进行控制。

## 添加数据来源的对象

|  |
| --- |
| **package** com.yss.framework.api.common.co;  **import** java.util.List;  /\*\*  \* 数据来源对象，提供被引用的表和字段  \* **@author** huangsq  \* **@version** 1.0, 20171206  \*/  **public** **class** DataMarkSource **extends** BasePojo {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **private** String c\_SOURCE\_TABLE;  **private** String c\_SOURCE\_FEILD;  **private** String c\_READONLY\_FEILDS;  **private** List<DataMarkTarget> dataMarkTargetList;  **public** String getC\_SOURCE\_TABLE() {  **return** c\_SOURCE\_TABLE;  }  **public** **void** setC\_SOURCE\_TABLE(String c\_SOURCE\_TABLE) {  **this**.c\_SOURCE\_TABLE = c\_SOURCE\_TABLE;  }  **public** String getC\_SOURCE\_FEILD() {  **return** c\_SOURCE\_FEILD;  }  **public** **void** setC\_SOURCE\_FEILD(String c\_SOURCE\_FEILD) {  **this**.c\_SOURCE\_FEILD = c\_SOURCE\_FEILD;  }  **public** String getC\_READONLY\_FEILDS() {  **return** c\_READONLY\_FEILDS;  }  **public** **void** setC\_READONLY\_FEILDS(String c\_READONLY\_FEILDS) {  **this**.c\_READONLY\_FEILDS = c\_READONLY\_FEILDS;  }  **public** List<DataMarkTarget> getDataMarkTargetList() {  **return** dataMarkTargetList;  }  **public** **void** setDataMarkTargetList(List<DataMarkTarget> dataMarkTargetList) {  **this**.dataMarkTargetList = dataMarkTargetList;  }  } |

## 添加数据被引用的对象

|  |
| --- |
| **package** com.yss.framework.api.common.co;  /\*\*  \* 数据被引用对象  \* **@author** huangsq  \* **@version** 1.0, 20171206  \*/  **public** **class** DataMarkTarget **extends** BasePojo {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **private** String c\_DMS\_ID;  **private** String c\_TARGET\_TABLE;  **private** String c\_TARGET\_FIELD;  **public** String getC\_DMS\_ID() {  **return** c\_DMS\_ID;  }  **public** **void** setC\_DMS\_ID(String c\_DMS\_ID) {  **this**.c\_DMS\_ID = c\_DMS\_ID;  }  **public** String getC\_TARGET\_TABLE() {  **return** c\_TARGET\_TABLE;  }  **public** **void** setC\_TARGET\_TABLE(String c\_TARGET\_TABLE) {  **this**.c\_TARGET\_TABLE = c\_TARGET\_TABLE;  }  **public** String getC\_TARGET\_FIELD() {  **return** c\_TARGET\_FIELD;  }  **public** **void** setC\_TARGET\_FIELD(String c\_FIELD) {  **this**.c\_TARGET\_FIELD = c\_FIELD;  }  } |

## 添加支持数据被引用标记的接口

需要控制数据是否被引用的Pojo对象，需要实现ISystemDataMark接口。

|  |
| --- |
| **package** com.yss.framework.api.common.co;  /\*\*  \* 数据被引用标记  \* **@author** huangsq  \* **@version** 1.0, 20171206  \*/  **public** **interface** ISystemDataMark {  /\*\*  \* 获取数据被引用标记，0或空未被引用  \* **@return**  \*/  **public** Integer getDataQuoteMark();  /\*\*  \* 设置数据被引用标记，0或空未被引用  \* **@param** dataQuoteMark  \*/  **public** **void** setDataQuoteMark(Integer dataQuoteMark);  /\*\*  \* 获取数据被引用只读的字段，多个字段用“,”分割  \* **@return**  \*/  **public** String getReadlonyFeilds();  /\*\*  \* 设置数据被引用只读的字段，多个字段用“,”分割  \* **@param** readlonyFeilds  \*/  **public** **void** setReadlonyFeilds(String readlonyFeilds);  /\*\*  \* 获取被引用的表名  \* **@return**  \*/  **public** String getDataMarkSourceTable();  /\*\*  \* 设置被引用的表名  \* **@param** dataMarkSourceTable 表名  \* **@return**  \*/  **public** String setDataMarkSourceTable(String dataMarkSourceTable);  } |

## 数据引用关系配置及注册

在各业务组件的META-INF/config/添加systemDataMark.xml配置文件，在组件启动的时候如果配置了该文件，将对应的配置文件注册到数据库中，其中source中的table和feild是复合主键，同一个source下的target的table和feild是唯一的。数据库中已经存在则不重新插入。

数据引用关系配置注册方法在BaseApplicationActivator类中实现，添加registerSystemDataMark方法，在启动组件的时候将注册信息写入数据库中。

systemDataMark.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <systemDataMarks>  <!--source table="被引用的表名" feild="被引用的字段" readonlyFeilds="被引用后只读的字段，多个字段用“,”分割" -->  <source table=*"T\_P\_AB\_PORT"* feild=*"C\_PORT\_CODE"* readonlyFeilds=*"C\_PORT\_CODE,C\_ASS\_CODE"*>  <!-- target table="引用的表名" feild="引用的字段" / -->  <target table=*"T\_PL\_PORT\_EXTEND"* feild=*"C\_PORT\_CODE"* />  <target table=*"T\_PL\_..."* feild=*"C\_PORT\_CODE1"* />  </source>  </systemDataMarks> |

## 添加数据被引用标记服务接口

提供获取数据源是否被引用和更新数据被引用状态的服务，通过调用 ISystemDataMarkService接口的服务。

|  |
| --- |
| **package** com.yss.framework.service;  **import** com.yss.framework.api.common.co.ISystemDataMark;  /\*\*  \* 数据被引用标记服务  \* **@author** huangsq  \* **@version** 1.0, 20171206  \*/  **public** **interface** ISystemDataMarkService {  /\*\*  \* 更新数据被引用的标识  \* **@param** dataMark 数据对象  \*/  **public** **void** updateDataMark(ISystemDataMark dataMark) ;  /\*\*  \* 查询被引用状态，1为被引用、0未被引用  \* **@param** dataMark 数据对象  \* **@return**  \*/  **public** **int** queryDataMark(ISystemDataMark dataMark) ;  } |

## 删除核心流程图



## 修改核心流程图



## 数据库表

### 数据引用源表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **表 名** | T\_S\_DATA\_MARK\_SOURCE | | **说 明** | | 数据引用源表 |
| **约束、索引** |  | | | | |
| **字段** | **名称** | **数据类型** | **可空** | **默认值** | **备注说明** |
| C\_IDEN | ID | VARCHAR2(30) | N |  | 主键 |
| C\_SOURCE\_TABLE | 源表名 | VARCHAR2(30) | N |  | 被引用表名 |
| C\_SOURCE\_FEILD | 源表名称 | VARCHAR2(50) | N |  | 被引用的字段名 |
| C\_READONLY\_FEILDS | 引用表字段 | VARCHAR2(1000) | N |  | 被引用后只读的字段 |

### 数据引用目标表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **表 名** | T\_S\_DATA\_MARK\_TARGET | | **说 明** | | 数据引用目标表 |
| **约束、索引** |  | | | | |
| **字段** | **名称** | **数据类型** | **可空** | **默认值** | **备注说明** |
| C\_IDEN | ID | VARCHAR2(30) | N |  | 主键 |
| C\_DMS\_ID | 源表ID | VARCHAR2(30) | N |  | 被引用的来源表的ID |
| C\_TARGET\_TABLE | 引用表名 | VARCHAR2(50) | N |  | 引用表的表名 |
| C\_TARGET\_FEILD | 引用表字段 | VARCHAR2(50) | N |  | 引用表的字段 |