Null和公式字段在过滤器中的最佳实践

在系统开发中常常因为空指针的错误引起流程上的报错，那么如何规避这种错误，该篇文章将详细说明。

- Null是结构化查询语言中用于表示数据库中不存在数据值的特殊标记。
- 在SQL和关系数据库中，NULL值是缺少数据
- NULL通常是一个保留关键字
- NULL不能等于零或任何值
- NULL等价与任何值的结果是未知的
- NULL = NULL为false

> 最近公司整体做了一次Re-org，将分公司拆分及建立独立省结构等。在这个项目中，需要从其他系统里同步到CRM的主数据部分的量级会成倍增长，这时对于接口的方案以及接口稳定性具有很大的挑战。在review了系统中大部分的接口代码后发现，在开发项目的过程中经常会忽略对Null值的处理，导致CRM在抓取数据时报错。在这篇文章中只讨论如何处理Null值的问题，我们的目标是提高代码中NULL处理的意识，以及它对事务性能的影响。我们还将提供减轻这些负面影响的方法。

过滤器为Null时对查询有什么有影响?

处理大数据量（LDV）时应尽量避免使用Null或者公式字段做过滤器。RDBMS（关系型数据库管理系统）的三值逻辑(3VL)中，Null值搜索意味着一个表的结果为不能被读取的索引。索引是以特定顺序存储数据的结构。因为NULL表示数据不存在，所以它不适合索引结构。

- Null： 默认情况下，底层的salesforce字段索引不包含Null值，因此，使用Null作为过滤器将会进行全表查询。
- 公式字段：默认情况下，公式字段没有底层索引，它为动态赋值，因此它需要完全扫描才能找到目标记录。

Null过滤器问题

Null过滤器问题在Salesforce环境中也很常见，主要来自代码(SOQL)在声明为Apex的一部分的列表中传递空值或空字符串。因此，在SOQL查询中包含空检查是良好编码实践的一部分。

1.案例一：通过聚合查询一个拥有500万条以上记录的对象，由于缺乏启用NULL的自定义索引，使用NULL搜索需要差不多7秒来读取整个记录表集。

```
SELECT count() FROM SAP_CONFIGURATION__c WHERE EQUNR__c != NULL
```

可以看到用这个过滤条件一共查询出5463113条数据，查询类型为全表查询，查询成本高于与selectivity的阈值高于1。

Cardinality	Fields	Leading Operation Type	Cost	sObject Cardinality	sObject Type
5463113		TableScan	4.789901...	5463113	SAP_CONFIGURATI...

2. 案例二：当筛选器是主对象上的字段时，在这种情况下，用户将获得所有没有Account映射到的Opportunity信息。系统最终会读取所有Opportunity记录和所有Account记录，然后过滤满足条件的记录。

```
SELECT count() FROM Opportunity WHERE account.id != NULL
```

Cardinality	Fields	Leading Operation Type	Cost	sObject Cardinality	sObject Type
1935		TableScan	2.784108...	2150	User

解决方案：

我们在项目开发之前就应尽量避免使用Null或者公式字段做过滤器，以免造成CPU超时和I/O使用量等错误。

如果是在平常系统运维中已经碰到了此类问题，最佳方案是建立一个自定义索引的文本公式字段，并使用文本代替Null值，如下代码示例：

```
RSGFormula__c = IF(ISBLANK(Responsibility_for_Sales_Group__c ), "blank",Responsibility_for_Sales_Group__c)
```

然后在采用`Responsibility_for_Sales_Group__c != 'blank'`来用于过滤条件，再将该字段申请建立一个索引，以确保过滤器可以搜索到索引。

经过优化之后的SOQL如下：

```
SELECT id, fax FROM user WHERE RSGFormula__c != 'blank'
```

除此之外， 在搜索结果为Null时，逻辑中仍然引用了查询之后的结果的情况下，系统将会出现空指针错误，如：

```
Account acc = [SELECT Id, Sap_Number__c FROM Account WHERE Sap_Number__c = '不存在'];
System.debug(acc.Sap_Number__c);  //运行时会出现空指针异常
```

修改方案：

```
List<Account> list_acc = [SELECT Id, Sap_Number__c FROM Account WHERE Sap_Number__c = '不存在'];
for(Account acc : list_acc) {
    //logic...
}
```

我们在代码逻辑对SOQL的处理上也应注意，如果需要直接把SOQL查询结果直接放在for循环上使用时，应注意此时过滤器变量可能的值。一但过滤器变量可以为空值运行时，将可能产生CPU超时和I/O使用量等错误，如下案例：

```
public static void getAccountDetail(String accId) {
    for(Account acc : [select id from contact where account.Id =: accId]) {
        //logic...
    }
}
```

修复方案：

```
public static void getAccountDetail(String accId) {
    if(accId != null) {
        for(Account acc : [select id from contact where account.Id =: accId]) {
            //logic...
        }
    }
}
```

总结：

1. NULL处理不当所产生的负面影响

- 运行Apex代码需要更长的执行时间
- Apex执行缓慢导致并发Apex请求限制错误
- 当Apex中的soql超过non-selective查询的200K对象大小限制时，它将会触发错误
- 高CPU资源消耗导致对请求进行节流或阻塞

2. 最佳实践方案

- 添加NULL字符串检查代码，不应该运行Null值
- 如果NULL一定要作为筛选器并且支持索引，那么创建一个带有NULL值的自定义索引
- 提高上游集成系统的数据质量
- 强制用户界面选择中的强制字段