JPA criteria with complicated data structure strategy

top-down or bottom-up

* 通常我們使用JPA Criteria

某部分來說 會有些方便 例如….

結構欄位名稱都相同的table

可以直接copy之前寫好的拿來用

但缺點就是

Criteria 有些語法 相對模糊

不像自己寫native sql 可以很確定 這個連結關係是否一定存在

* 問題點在於

如果今天結構比較複雜 master=>detail=>event

下面還有可能還有東西

然後某些欄位可能是不存在的連結情況下

如果我們就直接用master.detail.event.xxxxx屬性

<<<<問題容易發生在資料非連結很強的情況下>>>>

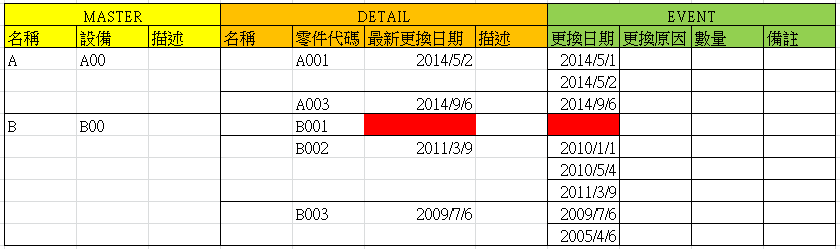
基本上 假設 有可能是不存在的情況下

OUTER JOIN

以oracle 來說就是 (+)

有可能會有非預期的結果產出

例如



這種情況下 如果是用

由上到下串下去的寫法

容易產生問題如下

1. 後面是用正常的join 因此可能有缺漏的資料不會出現?

EX.event沒有

1. 部分缺漏的資料是否 撈得出來?

EX.detail 部分資料有缺情況下 是否會影響到master

* 另外有種情況下

不是去撈某一層的某個屬性

像order by

Master.detail.event.change\_date

後面就會隱含join

也就會導致某些資料有可能非預期會撈不出來

* <<<分析與比較>>>

如果關係有可能不存在 以下寫法可能會有問題

Master=>detail=>detail的某個屬性

<<<可能會有問題>>>

Predicate p1 = cb.greaterThanOrEqualTo(//

root.get("oilSettingDetailList").get("modifyNoticeDate").as(Date.class)//

, startDate//

);

<<<可以改成>>>

我們可以額外去寫 找detail or event的方法

透過某個屬性

可以先找出清單 List<Event> or List<Detail>

這時再透過entity連結 可以回推List<Master>有哪些

最後在master façade裡面

我只要找id in xxxx 即可

//根據條件找出有符合的細項

List<OilSettingDetail> detailMatchList = this.oilSettingDetailFacade.findByCriterial(queryConditionX);

//根據細項把主項目有哪些給找出來 id list

List<Long> idList = new ArrayList<Long>();

for (OilSettingDetail detail : detailMatchList) {

idList.add(detail.getMasterId().getId());

//最後用in idList

Predicate p = root.get("id").as(Long.class).in(idList);

predicateList.add(p);

* <<<思考方式的改變 可以避免一些問題>>>

1. 正常連結很強的情況下 幾乎大部分撈東西 程式都可以寫在master的façade

至於下面的detail / event façade等等 都可以很虛 幾乎都沒有東西

1. 但如果關聯性有可能不存在

這時候master有些寫法 不要再用由上往下去問

而是將部分撈資料比例 分攤到detail 或event façade

這時候要撈某些東西 反而是由下往上

然後master只要關心 有哪些 master\_id 是有可能符合的

* <<<額外建議>>>

使用criteria 如果結構複雜

1. 改用native sql or jpql
2. 或是多測試 或印最後轉出的sql 出來看 連結關係是否如預期