

# ・業務的one-to-oneという カージナリティ

@ClubDB2 2014XmasParty

久保 雅彦 (jflute)

# 私はだれ？

- 久保 雅彦
  - はてなブログ、Twitterでは [@jflute](#)
- オープンソースプログラマ
  - DBFluteメインコミッタ
- 株式会社レイハウオリ
  - Javaチーム全体教育
- 株式会社ビズリーチ・株式会社ルクサ
  - 新卒研修・Java/DB周りのフォローイング



# ClubDB2登壇経験者

第169回 2013-09-13

ClubDB2でDBFlute語り

<http://d.hatena.ne.jp/jflute/20130914/clubdb2>

# メインテーマ

業務的one-to-oneとは？



# 能書きたらたら

物理的にはone-to-manyだが、  
業務的にはone-to-oneとして  
扱うリレーション

たぶん

みんな見たことある



# 例えば

[会員] と [会員住所]

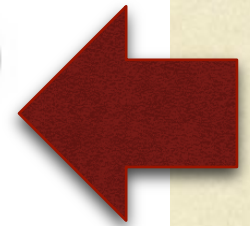
Aさん | 市原 2004年 - 2008年  
| 茂原 2009年 - 2012年

Bさん | 長柄 2007年 - 9999年

※住所が住んだ期間で積み上がる

# 有効期間カラム

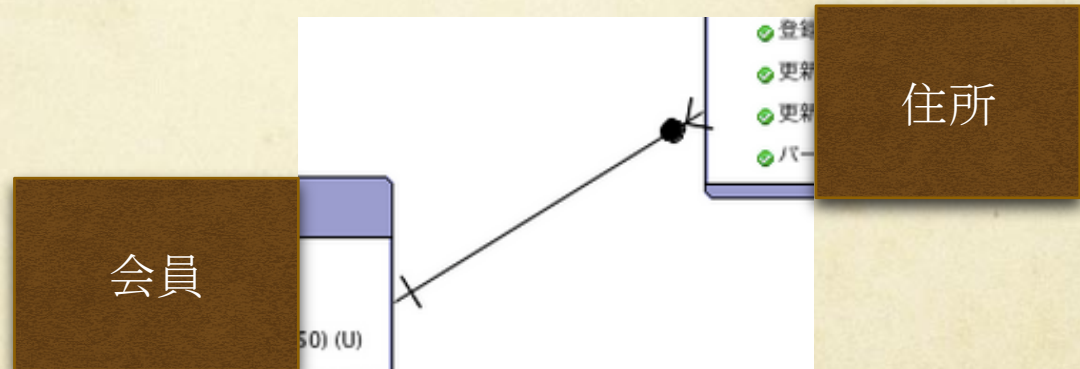
会員住所情報 / MEMBER_ADDRESS	
	会員住所ID/ MEMBER_ADDRESS_ID :INT
	会員ID/ MEMBER_ID :INT (U+)
	有効開始日/ VALID_BEGIN_DATE :DATE (U+)
	有効終了日/ VALID_END_DATE :DATE
	住所/ ADDRESS :VARCHAR(200)





だから

物理的には one-to-many



でも

普段は、  
過去の住所に興味はない



業務的にやりたいのは

「会員」と「今の住所」

を検索したい

だから

業務的にはone-to-one



なので

業務的one-to-one

えいっ！

select ... from 会員 mb

left outer join 住所 adr

on mb.会員ID = adr.会員ID

and adr.有効開始日 <= [現在日時]

and adr.有効終了日 >= [現在日時]

結合条件追加





# RDB的にはジレンマ

詳しくは、

漢のコンピュータ道さんの

スライドにて // 履歴データのここ

「データベース設計徹底指南」

<http://www.slideshare.net/nippondanji/db-engineerstudyanim>

でも現場では

自然切り替えができて便利

どうしてもまあ使う



でも実装でもジレンマ

いろいろな画面で

みんながこれ書く

ぜったい一人くらい

and adr.有効開始日  $\leq$  [現在日時]

and adr.有効終了日  $>$  [現在日時]



ああっ



こ う い う の

and adr.有効開始日 <= [現在日時]

and adr.有効終了日 <= [現在日時]



あああっ

やっちゃう

and adr.有効開始日 <= [現在日時]



ああああっ



というか

DB変更で条件増えたら  
ジ・エンド

※影響範囲ありすぎ...

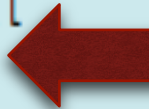
DBFluteなら

業務的one-to-oneの  
条件を再利用




# なんか設定ファイルに...

```
; FK_MEMBER_MEMBER_ADDRESS_AS_VALID = map:{  
  ; localTableName = MEMBER      ; foreignTableName = MEMBER_ADDRESS  
  ; localColumnName = MEMBER_ID ; foreignColumnName = MEMBER_ID  
  ; fixedCondition =  
    $$foreignAlias$$.VALID_BEGIN_DATE <= /*targetDate(Date)*/null  
and $$foreignAlias$$.VALID_END_DATE >= /*targetDate(Date)*/null  
  ; fixedSuffix = AsValid  
  ; comment = relation of valid address for the specified target date  
}
```



# するとJavaでは...

```
MemberCB cb = new MemberCB();  
cb.setupSelect_MemberAddressAsValid(targetDate);  
List<Member> memberList = memberBhv.selectList(cb);  
for (Member member : memberList) {
```



SQLでは、さっきの結合条件が展開される

結合条件の再利用



なので

“誰か一人くらいバグ”に強い！

条件変更にも強い！

アプリ側で

そういうツールを使えば、

DB設計者も安心



あとね、名前大事なんです

DB設計のパターンに名前があることで...

- 会話がしやすい
- 概念として学びやすい

さあ声に出して

業務的one-to-one！



もう一度っ

業務的one-to-one！

おしまい

ご清聴

ありがとうございました