4H−10

データベースアクセスインタフェース高度化の検討*

吉田 勝彦　加藤保夫†
NTT ソフトウエア研究所‡

1 はじめに

現在業務APにおいて、RDBMSが使われ始めている。APの機能拡張や他システムとの統合と共にAPの開発規模が増大し、DBの規模も増大しつつある。この開発規模の増大とともにAP開発の効率化が望まれている。RDBを利用したAPを開発する場合、DBのデータ構造がAPの詳細設計に大きな影響を与えている。また、DBの構造は性能条件から最適化が行なわれ業務内容からイメージしやすい構造には必ずしもなっていない。そこで、DBアクセスのインタフェースを実のデータ構造と分離し高度化する方式を検討した。本論文では、E-Rモデルでのデータ操作をSQLに変換する方式について報告する。

2 背景

現状のRDBは次のような問題を抱えている。

- データ構造の問題
  既存のDBシステムのデータ構造はAPに特化した構造になっており、必ずしも業務内容からイメージできる構造になっていない。また、データの持つ意味から汎用性のあるデータ構造とした場合、性能の劣化や、データ構造が複雑になることがある。
- データ構造の変更の問題
  設計時のDBのデータ構造は性能条件から構造が変わることがありAPの設計変更を余儀なくされることがある。

このような問題がAPの開発効率をあげることの出来ない理由となっている。DBアクセスインタフェースを高度化し、実DBのデータ構造とAPを分離することにより、ユーザの負担を減らすことが出来る。

3 アプローチ

ユーザに見えるデータ構造をユーザビューとして定義し、ユーザビューを通して実DBを操作することによって、ユーザには実DBのデータ構造を見せないようにする。これにより、ユーザの操作を実DBの操作と分離し、DBアクセスインタフェースの高度化を狙う。ユーザビューは、データの持つ意味から定義される概念構造とみなすことが出来る。検討の1例として、社内の開発現場でRDBの概念構造の設計を行う場合によく用いられているE-Rモデルを利用し、E-Rモデルのデータ構造をそのまま操作する方法を検討する。この言語をユーザビュー操作言語、データ構造変換に必要な情報を定義する言語をDB構造変換言語と呼ぶこととする。システムのイメージを図1に示す。

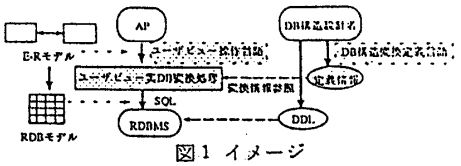

図1 イメージ

4 ユーザビュー

E-Rモデルの記述方法は幾つか提案されているが、ここでは、以下の省略された記述方法を用いる。例を図2に示す。

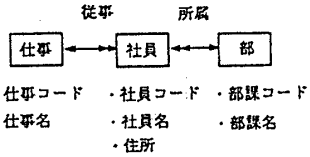

図2 E-Rモデル

四角で囲まれたものはエンティティ、エンティティとエンティティを結ぶ線はリレーション、エンティティ内には幾つかのアトリビュートがある。E-RモデルとRDBのデータ構造の関係は、エンティティがRDBのテーブル、アトリビュートがRDBのカラムに相当させている。RDBではE-Rモデルのリレーションに相当するテーブル間の意味づけは陽にはなく、ユーザが意識する必要がある。図2に相当するRDBのデータ構造は図3の様になる。

bu

bcode	bname	syacode

syain

syacode	syaname	addr	scode	sname

図3 RDBのテーブル構造

図3では、仕事と社員が一つのテーブルになっている。このように、論理設計、物理設計では、エンティティとテーブルが1対1ではなく、複数のエンティティに対し一つのテーブルが対応するようになることがある。

5 ユーザビュー操作言語

5.1 基本方針

E-Rモデルの問い合わせは、エンティティ、アトリビュート、リレーション、を使って作ることになるが、次の3つの方針にしたがいユーザに便宜をはかる。

- ユーザにRDBのテーブル構造は見せない。
- 問い合わせの形式はSQLと互換性を意識する。
- E-Rモデルをなるべく自然な形で利用する。

5.2 問題点

エンティティ(E)はテーブル(T)、アトリビュート(A)はカラム(C)に対応させることが出来るが、SQLでは、E-Rモデルでいうリレーション(R)に相当する考え方はない。そこで、リレーションの考え方を拡張する。従来検討されているSQLに近い言語仕様の記述方法と本論文で検討したリレーションを定義する方法2つとを比較する。

*A Study of High Level Human Interface for Data Base Access
†Katuhiko Yoshida and Yasuo Kato
‡NTT Software Laboratories

- LAMBDA[1]
  SQL に互換性を持たせながら、隣合うエンティティの関係を用いた方法。
- ２項間関係定義方法
  一組のエンティティを使用する場合、そのエンティティ同士を結ぶリレーションを指定する方法。
- 経路指定方法
  幾つかの条件を与えた場合、複数のエンティティによる接続条件の経路が出来る。経路を指定しておきそれを使用し検索条件を指定する方法。

表１リレーション指定法の評価

	2項間関係定義方法	経路指定方法	LAMBDA
E-Rに則した考え方	○	△	○
読みやすさ	×	○	○
詳細な記述	○	△	△
構文解析の容易さ	×	○	○
SQLとの互換性	×	○	△
モデルの整合性	○	○	△

表１より情報研の提唱するモデルの整合性と SQL との互換性を考え経路指定定義方法を採用する。

5.3 仕様

select 選択リスト
where 検索条件
linkaged by リレーション関係

選択リスト、検索条件：エンティティをテーブル、アトリビュートをカラムとしたSQL と同様の表記。
リレーション関係：エンティティとリレーションを順番に記述する。

6 構造変換定義言語

6.1 テーブルとエンティティの関係

次の５つが考えられる。

- エンティティとテーブルが１対１
  基本的な対応。
- エンティティとテーブルが１対ｎ
  １つのエンティティ内でテーブルが構造を持っている。
- エンティティとテーブルがｎ対１
  論理設計時の基本的最適化結果。
- エンティティとテーブルがｎ対ｍ
  モデルとして不自然であるので考慮しない。
- リレーションにテーブルを持つ
  エンティティ同士がｎ：ｍの関係の場合起こる。

6.2 仕様

E-R モデルから SQL へ変換するための定義は、E-R モデルの構造を定義する定義情報と、エンティティとテーブル、アトリビュートとカラム、リレーションとテーブル同士を関係付けるカラムに変換する定義情報からなる。

E-R モデルの構造を定義する定義情報は、ユーザビュー操作言語の構文解析時に使用し、E-R モデルと RDB の対応の定義情報はユーザビューから SQL の変換する時に使用する。

- E-R モデルの構造を定義する定義情報

```
structure E  { A, ···,
               linkage E by R(n:m) }
```

- E-R モデルと RDB の対応の定義情報

```
entity E = T{ attribute A = C,
              relation R = C direct T.C,
              relation R = implicit },
        T{ attribute A = C,
           relation R = C direct T.C }
```

7 考察

構造変換定義言語から、エンティティはテーブル名に、アトリビュートはカラム名に、単純に変換することが出来る。リレーション情報は構造変換定義言語の relation 項で定義されているカラムを "=" で結び and 条件で SQL の where 句に付加することで、目的とする結果が得られる。以上の操作は、単純な情報検索変換により行なえるので、処理は軽いと考えられる。 例として図２のユーザビューと図３のテーブル構造において次のような操作を行ないたい場合を示す。

【ユーザビュー構造定義】

```
structure 社員 { 社員コード,
                 社員名,
                 住所,
                 linkage 仕事 by 従事 (n:1)
                 linkage 部 by 所属 (n:1) }

structure仕事 { 仕事コード,
                仕事名,
                linkage 社員 by 従事 (1:n) }
structure部 { 部課コード,
              部課名,
              linkage 社員 by 所属 (1:n) }
```

【ユーザビュー実 DB 操作変換定義】

```
entity 仕事 = syain{ attribute 仕事コード = scode,
                     attribute 仕事名     = sname,
                     relation 従事        = implicit }
entity社員 = syain{ attribute 社員コード = syacode,
                    attribute 社員名     = syaname,
                    attribute 住所       = addr,
                    relation 従事        = implicit,
                    relation 所属        = syacode direct bu.syacode }
entity 部 = bu{ attribute 部課コード = bcode,
                atribute 部課名      = bname,
                reration 所属        = syacode direct syain.syacode }
```

【操作】
部課コードが 999 の社員名と仕事名を検索する。
【ユーザビュー操作言語】

```
select 社員.社員名, 仕事.仕事名
where 部.部課コード = 999
linkaged by 仕事 of 従事 of 社員 of 所属 of 部
```

【ＳＱＬ】

```
select syain.syacode, syain.scode
from bu, syain
where bu.bcode =.999 and syain.syacode = bu.syacode
```

8 まとめ

ユーザビューからデータを操作することによって、ユーザは RDB のテーブル構造を意識することなく、業務モデルを理解していることでデータを操作することが出来る様になる。さらに、データの概念構造と、物理構造が分離されるので、性能条件によるデータ構造の変更によって、AP 自体の変更をせずに済む。

9 今後の予定

今後は、更新系、登録系についても検討し、仕様の充実を図り、プロトタイプの作成を行い、記述実験を行う。また、RDB のデータ構造の変更の容易性を検討する。

参考文献

[1] F. Velez:"LAMBDA:AN ENTITY-RELATIONSHIP BASED QUERY LANGUAGE FOR THE RETRIEVAL OF STRUCTURED DOCUMENTS", Proc. of Int. Conf. on Entity-Relationship Approach, pp.82-99(1985).