NDBからRDBへのスキーマ変換方式の評価

6R−3

望月谷州子，加藤宣弘，関戸一紀
(株)東芝　情報処理・機器技術研究所

1　はじめに

従来、ビジネス分野におけるデータの管理では、ネットワークデータベース(NDB)が多く使用されてきた。NDBはデータ構造が、そのまま物理的状態に対応するところが多く、データ操作が手続き的であり、ユーザにとって使いにくい面が多い。それに対して、リレーショナルデータベース(RDB)は、データの独立性、モデルのわかりやすさ、データ操作の非手続き性などで、NDBに勝っている[1]が、これまで、性能面でNDBに及ばないとされてきた。しかし、近年、RDBの性能が向上し、ビジネス分野でも用途を限定して利用する傾向にある。そのため、既にNDBで構築されたDBをRDBに移植する検討も行なわれている[2]。

NDBからRDBへのスキーマ変換は、移植後のアプリケーションの生産性、メンテナンスの容易性、移植後の拡張性などに優れた方式を採用する必要がある。一方、移植による性能低下にも十分配慮する必要がある。本稿では、2つのスキーマ変換方式による、変換後のスキーマの構造と実アプリケーションによる性能評価から、より移植に適したスキーマ変換方式を示す。

2　スキーマ変換方式

NDBからRDBへのスキーマ変換で困難なのは、データの対応関係を示すセットの表現方法である。ここでは、セットの表現方法の異なる2つの一般的なスキーマ変換方式を示す。図1(a)は変換前のスキーマで、図1(b)はそれぞれのレコード型の項目である。下線のある項目はカルクキーである。図1(c)はオカーレンスレベルの表現で、セットによるレコードの対応関係を示している。

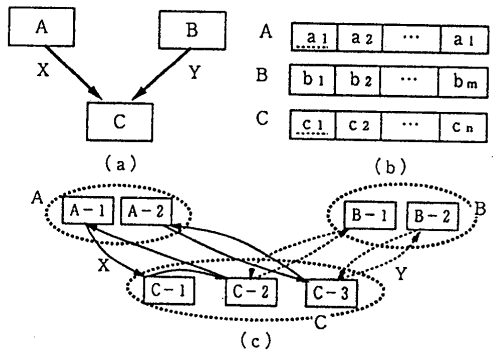

図1: NDBによるスキーマ構造

Evaluation of Schema Translation Method
from NDB to RDB
Yasuko MOCHIZUKI, Nobuhiro KATO, Kazunori SEKIDO
Information Systems Engineering Lab. TOSHIBA Corp.

2.1　方式1: メンバに対応するオーナの情報を付加

レコードはそれぞれタプルに対応させる。セットは、メンバレコードを表現するテーブルに、対応するオーナレコードの情報を付加する(図2)。オーナレコードの情報としては、オーナレコードを一意に識別できる属性を使用し、オーナレコードにカルクキーがある場合は、カルクキーの項目をメンバレコードを表現したテーブルの属性として追加する。カルクキーがない場合は、オーナレコードを表現したテーブルにタプル識別子(ID)を付加し、タプルIDをメンバレコードを表現したテーブルの属性に追加する。図では、Aのカルクキーa_1と、BのIDであるbidをテーブルCの属性として追加し、Cの各タプル(NDBでの各レコード)において、セット関係のあるオーナレコードを表現する。このようにセットをレコードを表現したテーブルの中で表現するには、オーナレコードを表現したテーブルに、対応するメンバレコードの情報を列挙する方式もあるが、対応するメンバレコードの数が不定で、フィールド数が決定できないため、不適当である。

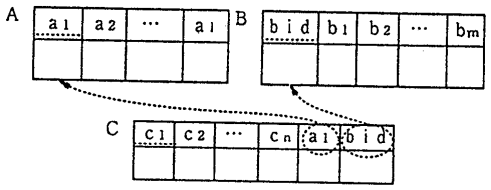

図2: 方式1

2.2　方式2: セットを独立したテーブルに展開

レコードを表現するテーブルに加え、セットを表現するテーブルを作成する。セットを表現するテーブルは、オーナレコードを表現する属性と、メンバレコードを表現する属性で、その対応関係を示す(図3)。それぞれのレコードを表現する属性は、レコードを一意に識別できる属性を使用し、レコードにカルクキーがある場合は、カルクキーの項目をセットを表現したテーブルの属性とする。カルクキーがない場合は、レコードを表現したテーブルにタプル識別子(ID)を付加し、タプルIDをセットを表現したテーブルの属性とする。図では、セットX,Yを表現するテーブルX,Yを作成し、XはA,Cのカルクキーであるa_1,c_1を属性として、AとCの対応関係を表現し、YはBのタプル識別子bidとCのカルクキーc_1を属性としてBとCの対応関係を表現する。

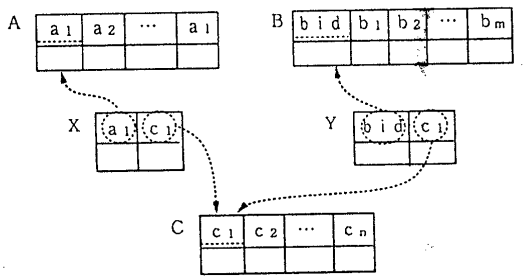

図 3: 方式 2

3 スキーマ変換方式の評価

3.1 変換後のスキーマによる評価

本実験では、実際に稼働しているシステムで使用しているNDBを上記の2方式でRDBに移植した。このNDBは2つのスキーマからなり、合計でレコード型数が87、セット型数が92である。

方式1においては、メンバレコードとセットが同じテーブルで表現されるため、その対応関係がわかりにくく、データの表現力が乏しいといった欠点があった。また、移植後の拡張性も低い。

方式2においては、レコードとセットが別のテーブルで表現され、それらがE-Rモデルの実体と関連にほぼ対応している。よって、概念モデルからRDBを直接設計して、実体と関連のテーブルに展開した[3]場合と、同様の形となっている。このことは、実世界に対応したスキーマを用いた表現であり、移植後の拡張性も十分高いことを意味する。

一方、方式2は、方式1に比べ、テーブル数が増え、セットに関係するデータ操作に原則としてJOINが入るため、データアクセス時間が長くなる可能性がある。そこで、実際のアプリケーションを用いて、性能を評価する。

3.2 性能評価方法

上記システムで実行時間の占める割合が高いアプリケーションをベンチマークに選んで評価を行なった。このアプリケーションでアクセスするスキーマを図4に示す。

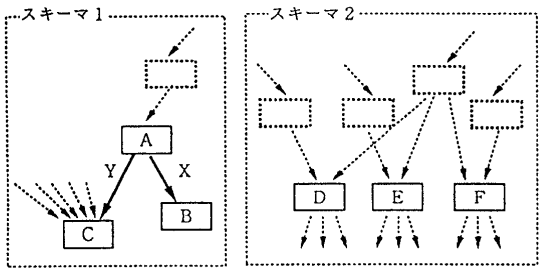

図 4: データ構造

また、このアプリケーションの流れは、まず、ある値をキーとして、Aからレコードを取り出す。XをたどってBからレコード群を取り出す。さらにYをたどってCからレコードを取り出す。この時、Cのレコードを1つ取り出すごとに、D,E,Fのいずれかのレコード型のレコードを1つ取り出す。なお、アクセスする6つのレコード型はすべてカルクキーを持っている。

このアプリケーションをRDBで実行すると、それぞれ次の流れとなる。ここで、4の実行は3で取り出されたタプルの数だけ繰り返し行なわれる。また、方式2において、Aを取り出すキーの項目は既にBも持っているため、Xをアクセスせずにを取り出せる。

- 方式1
  1. テーブルAのSELECT
  2. テーブルBのSELECT
  3. テーブルCのSELECT
  4. テーブルD,E,Fのいずれかの SELECT
- 方式2
  1. テーブルAのSELECT
  2. テーブルBのSELECT
  3. テーブルY,CをJOINしてSELECT
  4. テーブルD,E,Fのいずれかの SELECT

このアプリケーションを用いて、キーの入力から結果の出力までの応答時間を測定した。なお、入力キーの値によってテーブルCでSELECTされるレコード数が異なるので、そのレコード数が比較的平均的な値である8個の場合と16個の場合の両方について実験した。また、検索結果を標準出力に出力した場合の処理時間と、出力処理を除いた純粋のDBの処理時間を測った。

4 結果と考察

性能測定の結果の一部を表1に示す。

	出力あり		出力なし	
レコード数	8	16	8	16
方式1	1.60	2.63	1.41	2.29
方式2	1.64	2.72	1.47	2.39

表 1: 測定結果 (単位: 秒)

方式2ではJOINが1回あるため、応答時間は方式1に劣るが、全体の実行時間に対してわずかである。また、それぞれ、実際の業務に十分対応できることがわかる。このような性能に加え、データ表現力や拡張性などを考慮すると方式2の方が優れている。

5 おわりに

本稿では、NDBからRDBへのスキーマ変換方式を示し、実システムのDBを移植した。さらに、その上で動作するアプリケーションの応答時間から、スキーマ変換方式の評価を行なった。

今後は、本結果を参考にし、データの蓄積効率などを踏まえた各変換方式の有用性、DBやアプリケーションの性質による各変換方式の適用範囲等を検討していく。

参考文献

[1] 増永良文:「リレーショナルデータベース入門」
[2] 木戸他 (NTT):「CODASYL-DBからRDBへの移行時のデータ格納率評価法の検討」情報処理学会第43回全国大会 Oct.1991
[3] Ullman:「データベース・システムの原理」