データベースの設計における概念モデルの検討

6P−1

田中　久志,加藤　哲朗
ＮＴＴデータ通信株式会社

1 はじめに

計算機上のシステム設計は、データベースのデータ構造と、データを処理する手続きの設計の２つに分けられる。さらに、オブジェクトデータベース（以下ODBと略）のオブジェクトには、データ構造とそのデータ構造に関連づけられる手続きとしてメソッドが含まれる。

ODBのデータ構造には入れ子構造が存在するが、このような構造を設計する手法は現在確立していない。また、データを操作するためのメソッドについても、明確な設計法が存在しないので設計は複雑である。

このように、ODBの設計では２つ（データ構造とメソッド）の問題点がある。本研究では、前者のデータ構造の設計法について検討する。

2 背景

2-1 オブジェクトデータモデルについて

従来のデータベースの弱点を補うデータベースとしてODBに大きな期待が寄せられている[1]。

オブジェクトデータモデルとして、Won Kimは、Core object-oriented modelを提案している[2]。このCore object-oriented modelは、リレーショナルデータベース（以下RDBと略）に比べて、表1のような特徴がある。

Core object-oriented modelではRDBとは異なり、データモデルにユーザ定義型や、クラス階層、メソッド等が拡張されている。

ここでは、このデータモデルを対象とし、データ構造の設計について考える。

表１　ODBの特徴

	ODB	RDB	ODBのメリット
クラス	ユーザがテーブルをデータ型として定義できる	システムが用意したinteger,float,string....のみ	データに入れ子構造が可能なので統一したデータ管理ができる
データ値	配列構造／リスト構造 } が可	単一の値のみ	集合の大きさが、関連のデータ毎に変化するデータを入れる事ができる。
リインヘリタンス	クラス間のインヘリタンスがある	概念なし	同類のクラスを定義することが容易である。
メソッド	データとメソッドをカプセル化	データへの手続きがない	データを扱うアプリケーションがデータの種類や性質を知っている必要がない。

2-2 概念モデル

複雑な構造を持った実世界から直接データベースのスキーマを設計することは大変困難である。概念モデルはこの複雑な実世界を抽象化して設計するためのモデルである。

この概念モデルには、構造化設計で用いられるERモデルや、オブジェクト指向設計で用いられる、オブジェクトモデルが存在する[3]。

各々の設計方法では、この概念モデルからデータ構造が生成される。

2-5 データ構造

ここでは図の様なn対mのERモデル（図1-a）を例にとってこのモデルから生成されるデータ構造を示す。

ここでは、データ構造として、リレーションによる表現、配列表現、リスト表現を取り上げる。

2-5-1 リレーション（図1-b）

この概念モデルをRDBのテーブルで構成すると、実体のためのテーブル２つと、関連を示すためのテーブル１つの組となる。

2-5-2 配列構造（図1-c）

Core object-oriented modelではクラスの中に集合の属性を定義することで、n対mの構造を定義できる。配列構造はこの集合属性の一つであり、ページ内部で連続して配置されている。

2-5-3 リスト構造（図1-d）

リスト構造も集合属性の一つである。しかし、リスト構造は配列構造と違って、データが各々ポインタを用いてつながっている。

3研究の対象

3-1 問題点

現在、ERモデルに対応するRDBのスキーマを生成する技術は確立されており、商品化されている。

ODBは、RDBに比べデータ構造を表現する自由度が高く、概念モデルから複数のスキーマを生成することができる。そのため、現存の概念モデルからだけではスキーマを一意に定めることはできない。このモデル

A study of Conseptual model on database design
Hisashi TANAKA, Akio KATO
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION.

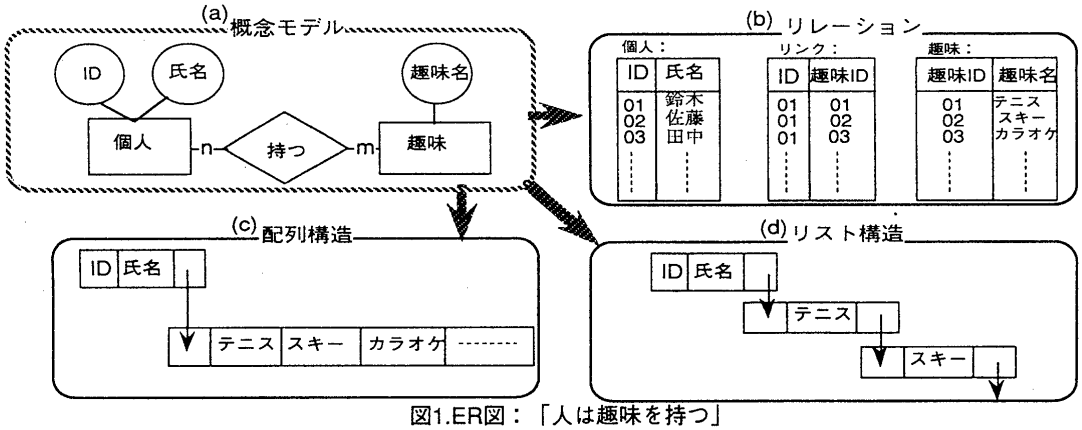

図1.ER図：「人は趣味を持つ」

から、最適なものを一つ選び出すことが問題である。

3-2 性能の要求

データベースはディスクアクセスを前提としている。従って、概念モデルにおける設計でディスクアクセスを最適化するデータ構造が決定できれば高性能なシステム設計が容易となる。

本研究では、性能という観点から、概念モデルからスキーマを生成するために必要な条件について考える。

4 考察

例として前述のモデル（図1）から出力されるデータ構造を考える。

4-1 検索例

図1で、「氏名」からその人の「趣味名」を検索する問い合わせを行なう。

リレーションの場合、テーブル「個人」から「氏名」に対応するIDを検索し、のテーブル「リンク」から、趣味IDを検索する。そしてその趣味ID一つ一つに対して、テーブル：趣味から「趣味名」を検索する。これは、結合演算となり、一般に性能が低いことが知られている[4]。

集合属性を表現する配列構造やリスト構造では、データ間をポインタで結んでいるため、「氏名」に合うオブジェクトが検索できるとあとは、ポインタをたどるだけである。このため検索時間は短い。

表2.データ構造とデータ操作（○：優、△：劣）

	テーブル	配列構造	リスト構造
趣味名の挿入	△	△	○
趣味名の複合検索	△	○	△
趣味名から氏名検索	○	△	△

この集合属性の中のリスト構造では、いろいろなページにデータが分散している場合があり、ディスクアクセスが起こりやすく、検索時間が長くなる。

一方、配列構造では、データがページに集中して入っているためディスクアクセスを最小に押えることができ、検索時間が短い。

これ以外の検索について各々のデータ構造における特性を表2に示す。

この結果、検索の種類によって最適なデータ構造は異なることがわかる。

5 まとめ

以上の考察は、色々な検索を考えたときの性能を基にして、データ構造を決定できることが解る。

しかし、ここでは、概念モデルとして多対多を一例として示しただけであり、これ以外の概念モデル（1対多、1対1）がどのようにCore object-oriented Modelに対応しているか示されていない。

また、これらの要因をデータベースの設計者が常に意識して設計するのは困難である。どのような種類のデータ操作が多いかの傾向を、概念モデルに盛り込む事ができれば、最適化したスキーマを自動生成できる可能性がある。

概念モデルの他のパターンを調べることや、各々のデータ操作の性能がシステム全体にどのように影響するかを調査することが今後の課題である。

○参考文献

[1]穂鷹　良介：オブジェクト指向VS関係データモデル,情報処理学会第44回全国大会,1992
[2]Won Kim：Introduction to Object-Oriented Databases,The MIT Press,1990
[3]原田　実：CASEのすべて,オーム社,1991
[4]滝沢　誠：データベースシステム入門技術解説～基礎から詳細技術まで～,SRCハンドブック,1971