

ソフトウェア設計支援システムMDL/MDA 5F-5 における複数仕様間操作の一方式

大槻 繁、柳 一美
(日立製作所 システム開発研究所)

1. はじめに

ソフトウェアのモジュール設計からロジック設計にいたる種々の設計仕様情報を一元管理し、設計作業を総合的に支援するシステムMDL/MDA (Module Design Language/Analyzer)を開発している。本システムは、編集、解析、文書作成等のツール群から構成されており、これ等のツール群の扱う設計仕様情報は、P. Chenの提案したEntity-Relationship Model (ERM) [1]に基づく仕様データベース管理システムによって蓄積、管理されている。

設計効率を向上させるための有効な一手段に、既存の仕様情報を再利用する方法がある。ある設計場面において、他の部分仕様を再利用するには、付随した情報をも含め一括して複写を行なうとともに、不要なものを削除したり、複写先で矛盾が起こらないような機構が必要である。本稿では、これを実現するための仕様データベースにおけるデータ操作方式について述べる。

2. データモデル

設計仕様データベースは、冗長性を排し、概念レベルの仕様構造を忠実に反映するために、ERMを採用している。設計仕様を扱うために、純粋のERMを、特殊化あるいは拡張を行い[2]、以下の特長を持たせている。

- (1) バリュースセット (Vセット) として、プログラミング言語の基本型を想定し、整数、実数、文字列の他に、スカラーや自然語記述用のテキストを導入したこと。
- (2) リレーションシップセット (Rセット) のロールには、複数のエンティティセット (Eセット) を対応させることが可能なこと。
- (3) Rセットのうち、エンティティの存在依存を規定する1:nの親子関係をウィークリレーションシップセット (Wセット) として明示的に導入したこと。
- (4) Wセットによる子エンティティは、順序付けられ、さらに、子エンティティの名称に、有効範囲を設けたこと。これにより、プログラムとモジュール、あるいは、ファイルのレコードとフィールドといった設計仕様特有の包含関係を簡潔に表現することができる。

3. データ操作の対象

仕様DBは、第2節で述べたデータモデルによるスキーマ S_i と、これに従ったオカレンスデータ D_{ij} から成る。 D_{ij} 内のデータ操作には、生成、削除、更新、参照等が完備な操作体系として既に用意されている。ここでは、より広範囲に渡る再利用を考え、複数のオ

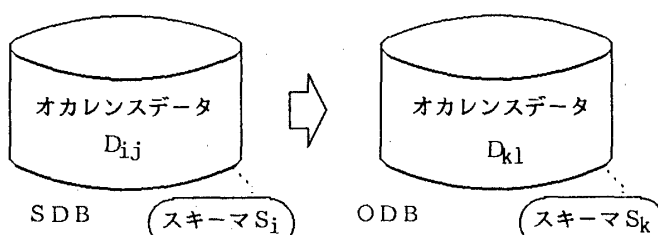


図1. 複数データベース操作

カレンスデータ間の操作、特に、複写操作について述べる。

データベース間複写操作は、図1に示すように複写元 D_{ij} から複写先 D_{kl} への《部分データ》複写操作である。前者をソースDB (SDB)、後者をオブジェクトDB (ODB)と呼ぶ。SDBとODBのスキーマは、異なってもよい。

部分データの単位として、Wセットによって構造化された部分木を考える。これは、例えば、あるプログラムを複写する場合に、これに含まれるモジュール、さらには、そのモジュールに含まれる局所変数も同時に複写しなくてはならないことを意味している。

部分木には、その基本構造を構成するエンティティとウィークリレーションシップの他に、部分木に含まれるエンティティ間のリレーションシップ、および、これらに付随したアトリビュート値をも含んで考えなくてはならない。

また、あるモジュールを再利用する場合は、これが呼出しているモジュールをも同時に複写しなくてはならない。この呼出し関係は、ウィークリレーションシップではない。このような状況を解決するためには、同時に、複数の部分木を操作する必要がある。この部分木の集合のことを《木集合体》と呼び、これには、木集合体を構成するエンティティ間のリレーションシップをも含む。ここで定義した木集合体が、次に述べるデータベース間操作の対象である。

4. 複数データベース間操作

(1) 基本操作

今、あるモジュール群 M_1, M_2 と、これ等がアクセスする共通テーブル V をDB間複写する場合を考えて見る。複写対象はそれぞれの情報を表現する部分木から構成される木集合体である。これをソース木集合体とよぶ。 M_1, M_2 が呼び出す基本モジュール N やテーブル V の定義仕様 U は、ソースDBにもオブジェクトDBにも存在しているものとする。従って、 N や

Uは、複写する必要はないが、M1, M2, VとN, Uの間のリレーションシップは、複写しなくてはならないため、N, Uを表わす部分木から構成される環境情報をも操作時に指定するものとする。すなわち、DB間複写操作の基本指定は、

Copy (M1, M2, V)
To (M1', M2', V')
With (N, U).

である。本操作による動作例を図2に示す。

(2) パス式

付加情報を、きめ細かく指定するためにパス式を導入する。これは、図3-aに示すように、主として、ソース木集合体に含まれているエンティティを起点として環境木集合体のデータをも複写対象に組み入れる働きをする。例えば、パス式2は、EセットE2に属するエンティティを起点として、RセットR2のリレーションシップを辿った先のエンティティの任意のWセットによる子孫を示している。

これは、さらに、図3-bのように、呼出し関係は複写するが、呼出される関係の複写を抑止したりするためにも使用することができる。

(3) フィルタ

通常は、SDBとODBとのスキーマは同一であるが、これと《似た》スキーマに従うDBとの間の複写操作も可能にし、さらに、不要な構造をスキーマレベルで抑止するためにフィルタを使う。フィルタは、ソース木集合体とパス式によって指定されたSDB側の対象を入力とし、ODB側のスキーマに合った構造に変換し出力するものである。

フィルタ定義は、SDBのスキーマとODBのスキーマの間のマッピングである。ここで、「似た」とは、マッピングが、EセットはEセットへ、RセットはRセットへというように、単純な対応がつけられるという意味である。例えば、

- f1: Eセット「手続き」⇒「モジュール」
- f2: Eセット「関数」⇒「モジュール」
- f3: アトリビュート「変更日」⇒ε
- f4: アトリビュート「バージョン」⇒ε

とマッピングを定義すると、SDBで、「手続き」と「関数」というEセットの区別があるものを、「モジュール」と統一化したり、「変更日」や「モジュール」等の管理用のアトリビュートを抑止したりすることが可能となる。

5. おわりに

本稿で述べた複数仕様DB間操作は、複数DBを一括して扱うというよりは、むしろ、あるDBの操作中に、一時的に、他のDBからデータを流用/再利用するための高効率な方式を指向している。一般のDBとは異なり、仕様DBでは、完備な操作体系よりも、このような、ある程度制限されつつも、きめの細かい指定が可能な操作体系の方が有用であると考えられる。

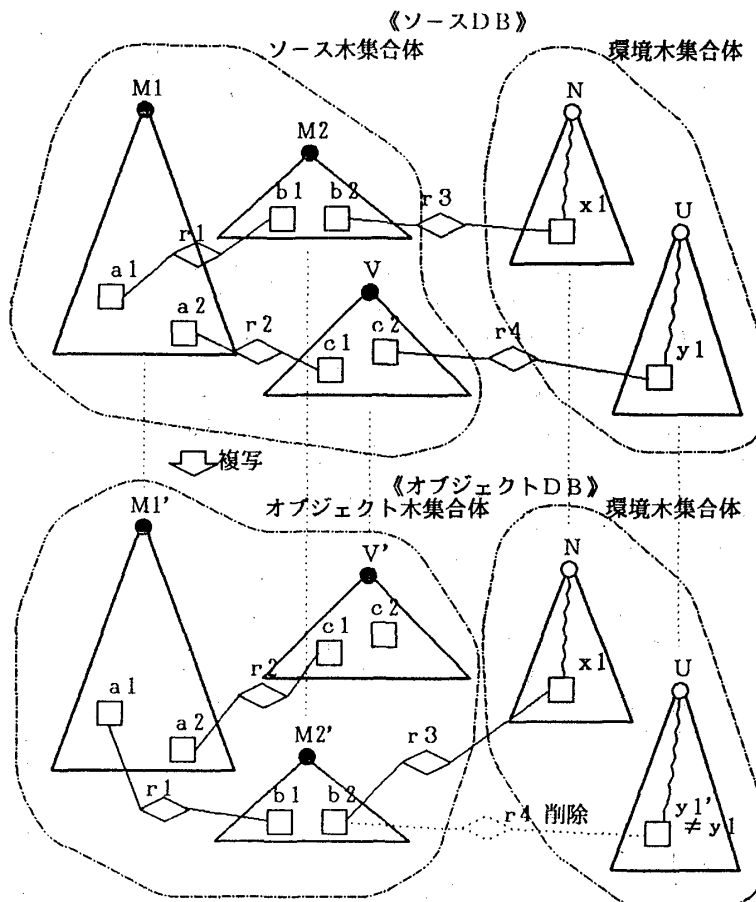


図2. 基本操作

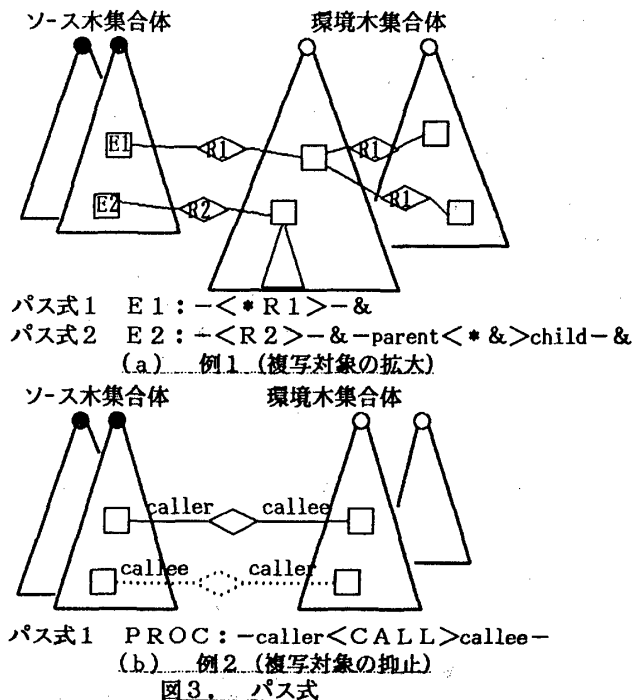


図3. パス式

<参考文献>

- [1] Chen, P.P., "The Entity-Relationship Model--Toward a Unified View of Data", ACM Trans. Database Systems, Vol.1, Mar. 1976
- [2] 大槻他, ソフトウェア仕様データベース-インターフェースの一考察, 第29回全国大会, 1984