ソフトウェア工学 118－7
（1998．3．10）

RDB処理の3階層コンポーネント構築の一方式

北畠重信*1、小泉寿男*1、白鳥則郎*2

*1 三菱電機(株)
*2 東北大学

概要

３階層アーキテクチャによるＲＤＢ処理ソフトウェアの構成単位をコンポーネントのセットとして生成することによる、ソフトウェア開発効率化を検討している。その支援の要素技術として、テーブル間の関連に着目した木構造ＥＲ図を活用する。本稿では、支援の概念を説明し、木構造ＥＲ図活用の原理とそれを適用した試作ツールの例を説明する。

A Method for Building Software on Reloational Database Processing as a collection of components in three tire architecture

Shigenobu Kitabatake*1, Hisao Koizumi*1, Norio Shiratori*2

*1 Mitsubishi Electric Coorporation
*2 Tohoku University

The authors are planning to assist the Software Construction on relational database processing by generating source codes for components in three-tier architecture. We use ERT diagram(Entity-Relationship Tree Diagram) to specify the parameters in generating components. We propose the use of ERT diagram as a helpful tool to extract the information from the E-R Diagram.

In this paper, we explain the outline of support、and explain the principle of applying ERT diagram, and show the use of it by using a tool prototype.

1．はじめに

ソフトウェアの生産性・信頼性向上の有力手段として、再利用技術が古くから期待を集めてきた。近年では、ブラックボックス再利用に基づくコンポーネントウェアや、継承によるクラス再利用を基本とするフレームワークが脚光をあびている。

容易に利用できることが期待される用途には、コンポーネントが適切であるが、ブラックボックス化できる範囲には限界がある。そこで、ＲＤＢ処理ソフトウェアの開発の支援において、コンポーネント化可能な範囲を、見かけ上広げる方式を検討した。これは、幾つかのシステム事例の調査の結果、大量データの蓄積・検索・更新の定型処理という共通性から、個々の基本処理単位の骨格構造には類似性が見られ、再利用効果があると考えたからである。ＤＢ処理パターンを部品化し、設計時カスタマイズ機能のレベルでＥＲ図（Entity-Relationship Diagram）情報をパラメータとして与えることにより、実行時に実際に動作するコンポーネントのセットを生成する。コンポーネント生成のパラメータを木目細かく効率的に与えるために、ＥＲ図の注目部分を抽出した木構造図（以下、木構造ＥＲ図と略称）を用いる。

この方式により以下の課題の解決をねらう。

（1）近年、情報基盤技術や開発基盤技術の変化のスピードが加速度的に増大している。これに同期をとるのを容易にする為に、以下の策を取る。

・任意の使いやすい開発環境と共存し相乗効果が得やすいように、標準インタフェースに準拠したコンポーネント・セットを生成する。

・軽量・コンパクトな支援単位の集まりを提供し、個々の改良を容易にするとともに、最重要部分からの段階的サポートでも大きな効果のある部品形式とする。

（2）ウイザード形式によるデータベース処理プログラム生成によく見られる以下の欠点を解消する。

・単一、あるいは２～３のテーブルの視野内での生成しかできない。

・ビジネス・ルール、業務ロジックの組み込みが困難である。

・モデルに基づいた構築の支援が行えない。

（3）RDB処理を中心としたシステムにおいて、データモデルの分析・設計・理解にＥＲ図は不可欠であるが、エンティティの数が増えると、図が見難く把握しずらくなる。設計・検討時のモデルの理解には、思考の順序にしたがって、注目するエンティティとその関連先にのみ焦点を当てることによって、縦横に視点を変えてモデルを見る手段が欲しい。

本報告では、支援イメージと、木構造ＥＲ図適用の原理を説明する。また、木構造ＥＲ図による生成パラメータ指定の一部機能を検証する試作ツールの例を示す。

以下、第２章では、支援イメージと木構造ＥＲ図利用の基本的な原理、第３章では、試作ツールでの適用事例を説明し、第４章に考察を述べる。

2．支援イメージと、木構造ＥＲ図適用の原理

2.1 支援イメージ

ＲＤＢのスキーマ定義には、ＥＲ図相当のものが必須であるし、それが無い場合でも、既存ＤＢからのリバースによるＥＲ図作成は一般に容易である。

最も単純な場合の支援としては、ＥＲ図からＤＢ処理基本パターンに基づいてＤＢ基本処理

プログラムを作成して、即実行可能なものとする。

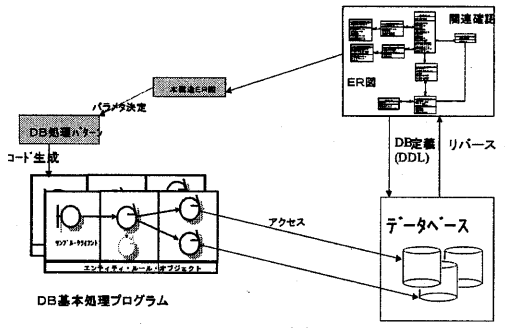

図2－1 単純な場合の支援

一般の支援の場合には、業務固有のビジネス・ルールの反映、人間の理解しやすさを考慮したユーザインタフェース改善、業務からの要

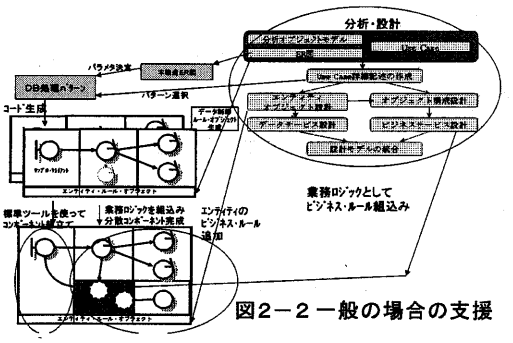

図2－2 一般の場合の支援

求やユーザの好み、操作性の観点から、複数の基本処理の組合わせによる処理単位組立て、などを行うことによってアプリケーションを完成させる。

2．2　木構造ＥＲ図適用の理由

木構造ＥＲ図は、指定したエンティティをルートとして、「１：ｎ」や「ｎ：１」の関連を辿って作ったエンティティの木構造である。エンティティは、ＲＤＢテーブルに直接対応するので、この木構造はＳＱＬに密接に対応が取れる。ＲＤＢ処理ソフトウェアの機能は、画面や帳票に含まれるビューを通して情報が伝達されることによって実現されるので、この木構造のノードをエンティティではなくビューの視点で見ると、一定の条件を保って、木構造を自由に

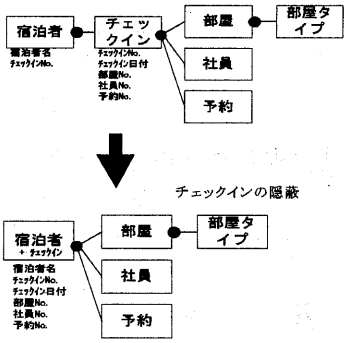

図2-3 木構造の圧縮（ノードの隠蔽）－n:1の1側をn側に併合

変更することができる。

コード生成のパラメータとして木構造を使う場合、この変形による最適化を行える場合がある。例えば、図2-3では、後にのべるように、検索パスを圧縮し、画面遷移を少なくする例である。

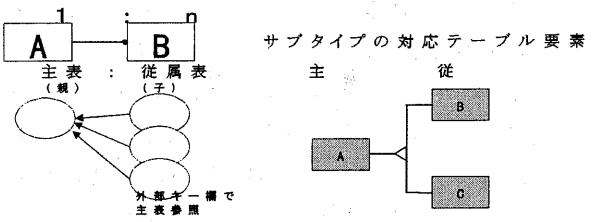

図２－４ 関連とサブタイプ　①n:1（従：主）木構造 ②1:n（主：従）木構造

また、サブタイプ化の関係は、１：ｎ関連と同様に扱うが、木構造上にサブタイプ関係であることを明示することにより、サブタイプ固有の処理パターンとビューの調整に木目細かに対応可能となる。

以上のような操作は、木構造上で行うのが、最もわかりやすく、かつ容易である。

２．３　隣接エンティティ間の関連

最も多く見られる基本的なデータベース処理のパターンでは、一般に、関連を順次辿って、隣接エンティティへと処理が波及して行く。各エンティティに対する処理は基本的に、CRUD（Create、Read、Update、Delete）なので、関連で結ばれた隣接する２つのエンティティ間での関連した処理の組み合わせは限られており、同様の関係が、繰り返し、再帰的に存在する。この視点で見た各木構造の特徴は以下の通りである。

（１）　ｎ：１木構造

ｎ：１木構造は、指定されたエンティティ・タイプをルートとして、すべてのｎ：１関連を辿って得られる木構造である。この木構造は、例えば、以下の２通りの解釈あるいは利用が可能である。

①ルートのエンティティ・タイプのビューを定義する。

ルートのエンティティ・タイプのインスタンスが一意に定まると、そのｎ：１木構造に含まれるすべてのエンティティ・タイプのインスタンスが（高々）一意に定まる。すなわち、ルート以外のインスタンスの属性は、ルートのインスタンスを詳細に説明するものと解釈できる。

これは、直接、ＳＱＬのビューに対応し、ビジュアル操作でのＳＱＬ文生成が可能である。

②可能なすべての検索パスを示す。

リーフ・ノードからルート・ノードに至る各経路は、インスタンス一覧からの選択により、順次、候補を絞って行くことが可能な、すべての検索パスを網羅している。

ただし、そのような処理プログラムを生成する場合には、すべてのノードに対して画面を生成すると冗長になるので、通常、途中のノードを隠蔽するのが適切である。

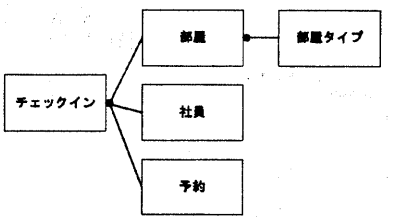

図２－５　ｎ：１木構造図

また、入力処理画面において、「支店番号」や「取引先番号」の入力をデータ一覧からの選択により入力するのは、①と②の両方の解釈を利用しているといえる。

（２）　１：ｎ木構造

１：ｎ木構造は、複数のデータ・ビューの可能性を示す。マスターディテイル関係などを包含する、より一般的な形式を表現することができる。

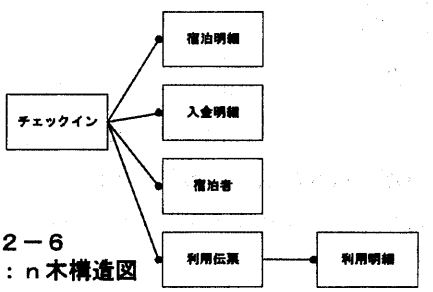

図２－６
１：ｎ木構造図

２．４　隣接しないエンティティ間の関

（１）同じ外部キーを持つ場合

これは、通常、１：ｎ木構造においてルートの直下（第１レベル）に来るノード同士の関係にあたる。例えば、図２－６において、第１レベルのエンティティは、すべて、チェックインの主キーを外部キーとして持つ。これは、特に、集計・サマリ情報のためのビュー生成に重要で

ある。

（2）一般の場合

任意の２エンティティを含むｎ：１木構造、１：ｎ木構造の抽出、論理を追ってアクセス・パスを辿るために任意のノードから任意のレベルずつ順次１：ｎ木、ｎ：１木を展開する機能等により、エンティティ間の関係の理解が非常に助長されるであろう。調査した実システム事例の中には、プログラム構造はシンプルで、データ抽出条件が非常に複雑なものも幾つかあった。このような場合は特に、アクセス・パスの論理をビジュアルに追えるような支援が有効となる。

３．木構造ＥＲ図適用事例

代表的な一部機能を実装して確認した。以下では、第２章での説明に対応する、試作ツールでのデータベース基本処理プログラム骨格の作成例を示す。

（1） ｎ：１木構造

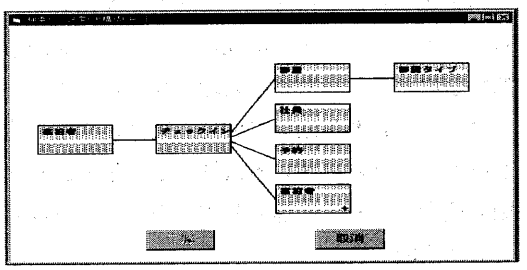

図 3-1 n:1木構造

上図は、「宿泊者」をルートとするｎ：１木構造の例である。これは「宿泊者」に至るすべてのパスを示しており、任意のリーフから順に候補を絞っていくことができる。

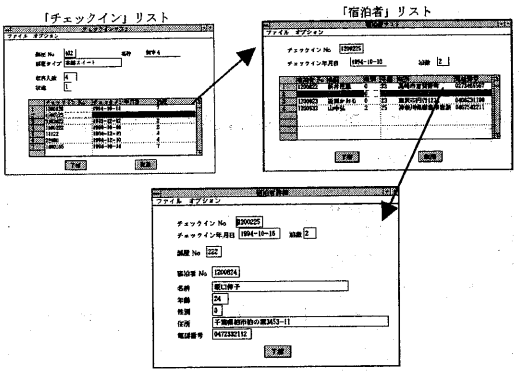

図 3-2 検索

すべてのノードに対して画面を生成すると冗長になるので、通常、途中のノードを隠蔽するのが適切である。

下図では、「チェックイン」が定まれば、それに対する「部屋」、「部屋タイプ」、「社員」、「予約」は、一意に定まり、その「チェックイン」の情報を人間に分かりやすく伝える為の補助情報となっている。

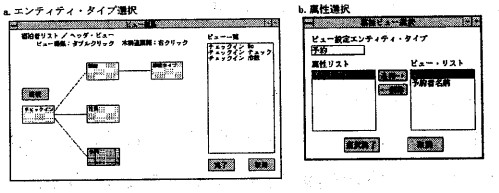

図 3-3 ビューの定義

次は、入力画面でデーター覧から補助情報を選択して入力する例である。

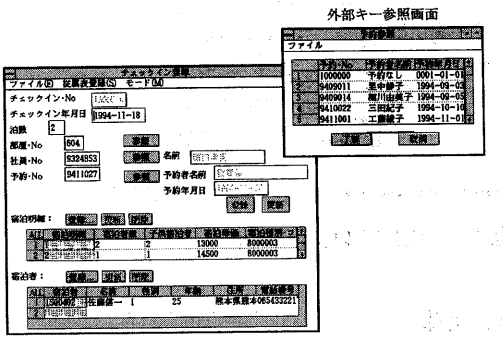

図 3-4 入力画面

（2）１：ｎ木構造

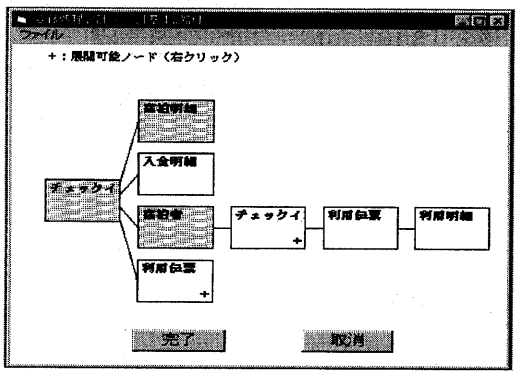

図 3-5 1:n木構造

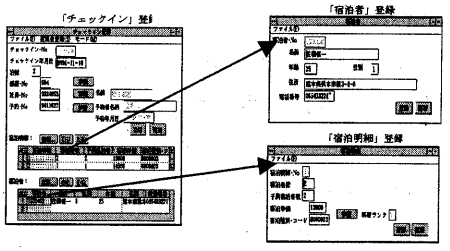

図 3-6 1:nによる入力画面

（3）生成プロトタイプ例

標準パターンのプロトタイプは、瞬時に生成可能であり、複数処理をメニューでまとめておくことができる。

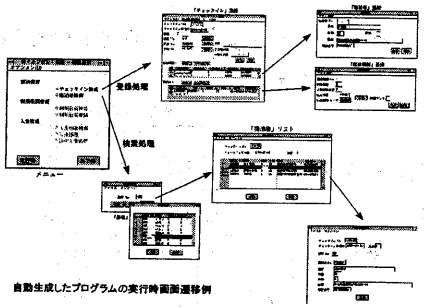

図 3-7 生成プログラム例

４．考察

試作ツールによる検証により、木構造ＥＲ図の理解しやすさ、操作容易性の確認、及び、登録の再帰的繰返し構造を一つのトランザクションとする場合の内部バッファ処理の要件等が確認できた。

プログラムの生成が、データベースの性質を直接用いた単純な原理に基づいているため、試作したツールでは、任意の指定でとにかく実行可能なコードを瞬時に生成し、修正・実行・確認のサイクルを短時間で行えた。また、どのパターンも、「①処理パターン選択」、「②対象テーブル選択（画面遷移決定）」、「③ビュー決定」、「④画面レイアウト決定」、「④コード生成」のステップで行えることが確認できた。

ただし、今回の試作では、パターン定義のスケルトンコードとコード生成制御ロジックの分離が十分ではなかったので、さらに整理・洗練化して方式の改善を図りたい。基本処理パターンを部品として容易に追加してゆけるようにする為にも、技術要素としてこれは必須である。

５．おわりに

本報告では、３階層アーキテクチャによるＲＤＢ処理ソフトウェア開発において、ＥＲ図情報から必要なＲＤＢ基本処理のコンポーネントのセットを自動生成して開発を効率化する方法を説明し、それに用いる図法として木構造ＥＲ図を紹介し、その原理と利用法を試作ツールとともに説明した。

これまで、個々の要素技術をそれぞれ検討／検証行ってきたが、今後、個々の検討の深化と支援の統合の検証を行う予定である。

参考文献

(1) OMG Document : CORBA Component Imperatives, ORBOS/97-05-25

(2) OMG Document : Common Business Objects, bom/97-11-11

(3) ERwin 手法ガイド, Logic Works, Inc. (1996)

(4) Gamma, E., : Design Patterns : Elements of Reusable Object-Oriented Software , Addison-Wesley, (1995) (本位田真一, 他監訳：オブジェクト指向における再利用のためのデザインパターン, SOFTBANK, 1995)

(5) Pree, W. : Design Patterns for Object-Oriented Software Development, Addison-Wesley, (1995) (佐藤啓太, 他訳, デザインパターンプログラミング, トッパン, 1996)

(6) Hay, D. : Data Model Patterns - Conventions of Thought, Dorset House Publishing , (1996)

Orfali、R., : Client/Ｓｅｒｖｅｒ Programming with Java and CORBA, Addison-Wesley, (1997) (並河英二, 他訳：Java & CORBA C/S プログラミング, 日経ＢＰ, 1997)

(6) 鈴木、萬木、原田、徳本、他 : ビジネス系システム開発におけるオブジェクト指向開発技法(1)～(4)、情報処理学会第 54 回全国大会