マルチメディア通信と分散処理 86－1
グループウェア 26－1
（1998. 1. 29）

RDB処理の分散コンポーネント構築支援方式

北畠重信*1、萬木優子*1、原田道明*1、鈴木由美子*1、
小泉寿男*1、白鳥則郎*2

*1 三菱電機(株)
*2 東北大学

概要

RDB処理ソフトウェアの開発を対象とした、分散コンポーネントの実装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図の活用を提案する。本稿では、木構造ER図活用の原理を説明し、試作したツールの例に基づいて、分散コンポーネント構築支援への適用の可能性について報告する。

A Support Technique for Building Distributed Component Objects on Relational Database Processing

Shigenobu Kitabatake*1, Yuko Yurugi*1, Michiaki Harada*1, Yumiko Suzuki,*,
Hisao Koizumi*1, Norio Shiratori*2

*1 Mitsubishi Electric Coorporation
*2 Tohoku University

The authors propose the use of Entity-Relationship tree diagram as a helpful tool for building distributed component objects on relational database processing.

In this paper, we explain the principle of applying Entity-Relationship tree diagram, show the use of it by using a tool prototype, and we consider the advantage of applying it to developing distributed component objects.

1. はじめに

分散オブジェクト基盤の標準化は、基本部分がまずは一段落し、次のステップの一つとしてコンポーネントに必要な機能のサポートの動きが有る。

これによって、今後、コンポーネントをビジュアルに組み立ててアプリケーションに組み上げるビルダ技術は、発展・普及にさらに拍車がかかるものと思われる。そこで、次に必要なのは、コンポーネント開発者が、業務固有のロジックを組み込んだ分散コンポーネントを効率的に開発する技術である。共通的なビジネス・オブジェクトの標準化は進められている。一方、自前で開発・実装の必要なコンポーネントでは、その対象に最適な実装自由にその要件に合った最適な実装方式を選択できるようにして、開発効率を上げたい。

本報告では、RDB処理ソフトウェアの開発を対象とした、分散コンポーネントの実装支援の要素技術の一つとして、木構造ER図の活用を提案する。 木構造ER図とは、RDBスキーマ・レベルのER図(Entity-Relationship Diagram)において、注目するエンティティ(RDBテーブルに相当)から「1:n」あるいは「n:1」の関連を辿ることによってエンティティの木構造を表現したものである。SQLのビューの定義や、関連するテーブルへのアクセス・パスの確認や選択が容易に行える特徴がある。大量データの蓄積・検索・加工の処理や画面、帳票は、RDB テーブルとSQL処理に直感的に対応しており、分散オブジェクト環境においても、SQLを有効活用した内部実装方式をとりたい場合がある。一方、それ以外の方法も、種々、実現可能であり、別の方法をとりたい場合もある。そこで、オープンな開発環境で利用可能なコンポーネントの設計時機能として、そういった方式の実装支援機能を組み込んでしまい、最も効果的な支援を状況に応じて選択可能にしたい。木構造ER図は、原理が明確で単純化されているので、この目的にかなった要素技術の一つと考える。

以下、第2章では、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、第4章では、分散コンポーネント構築支援への適用について考察する。

2. 木構造ER図適用の原理

本稿では、木構造ER図に基づく説明のみを行うが、実際の分散コンポーネント生成では、分析・設計で抽出したビジネス・ルールが組み込まれる。

以下の説明は、データモデルに基づくリレーショナル・データベースに直接アクセスを行うシステムを対象に考える。あるトランザクションで扱われるエンティティは、互いに関連を辿って到達できるアクセス・パスを持ち、アクセス・パスの向きを定めるのは、関連のカーディナリティである。そこで、注目するエンティティをルートとして「1:n」や「n:1」の関連を辿ってエンティティの木構造を作ると、エンティティ間の関連の理解が非常に容易になる。

2. 1 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3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。

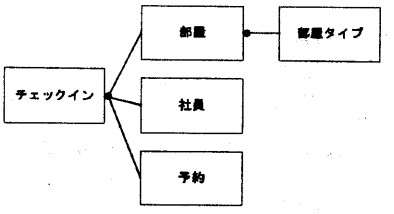

図2－1　n：1木構造図

また、入力処理画面において、「支店番号」や「取引先番号」の入力をデーター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。

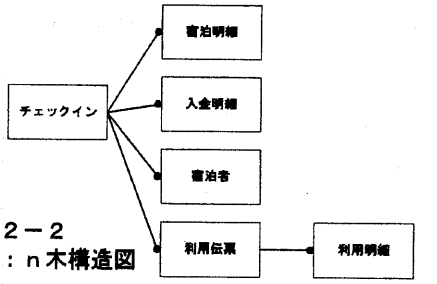

図2－2
1：n木構造図

2.2　隣接しないエンティティ間の関係(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2－2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。

2.3　隣接しないエンティティ間の関連(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

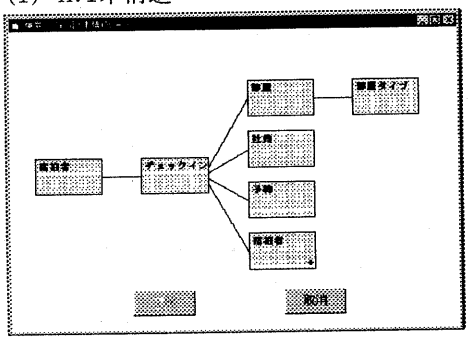

図 3-1 n:1木構造

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。

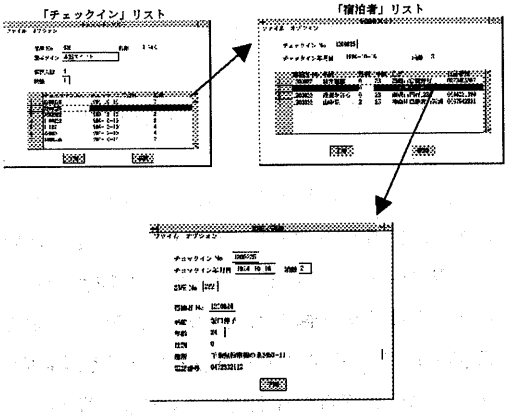

図 3-2 検索

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。

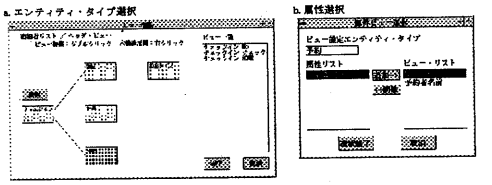

図 3-3 ビューの定義

次は、入力画面でデータ一覧から補助情報を選択して入力する例である。

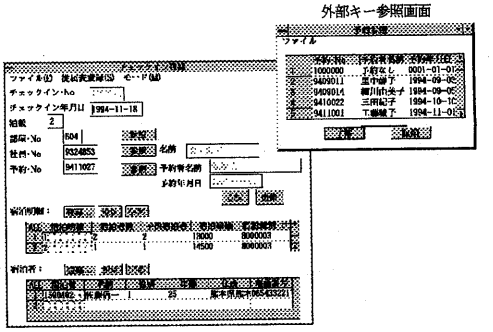

図 3-4 入力画面

(2) 1:n木構造

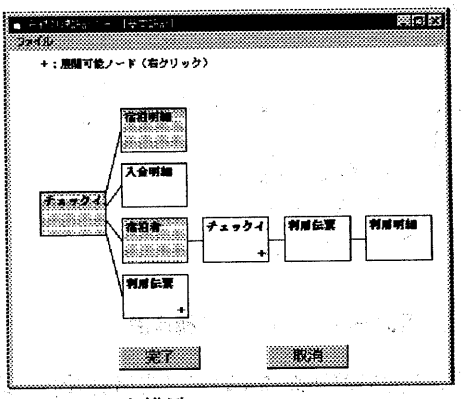

図 3-5 1:n木構造

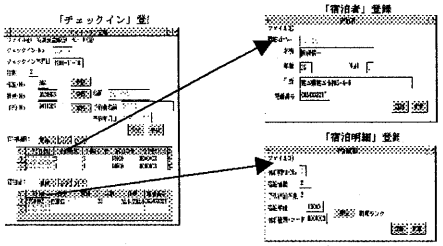

図 3-6 1:nによる入力画面

(3) SQL文の生成

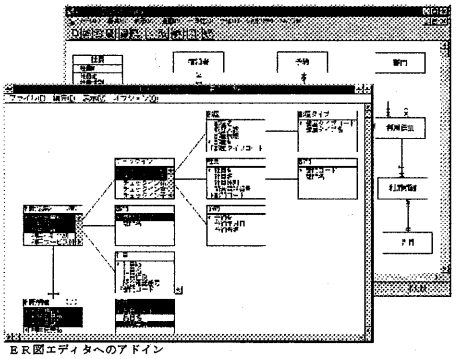

図 3-7 ビューの選択

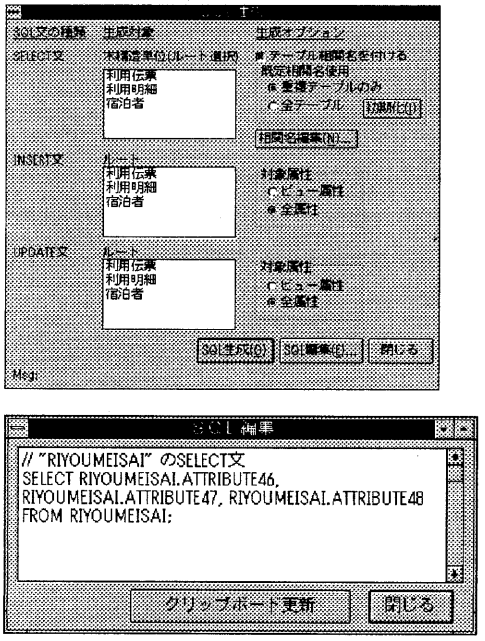

図 3-8 SQL 文の生成・編集

(4) 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。

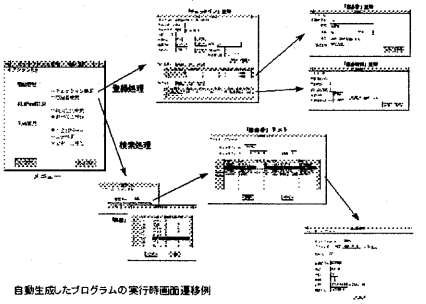

図 3-9 生成プログラム例

4. 考察

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、任意の指定で、とにかく実行可能なプログラムが瞬時に生成でき、修正・実行・確認のサイクルを短時間で繰り返せる。

サブタイプもフラットなテーブルに還元して、同様に取り扱うことになるが、現実世界では、各サブタイプを画面の可変部として、サブタイプごとの画面切替えといったようなことがよく行われるので、サブタイプを意識したパターンは考慮すべきだろう。ただし、分散コンポーネントの実装の支援のみを目的とすることにより、画面遷移の詳細等、プレゼンテーション層に関することは、コンポーネント利用者にまかせることができるので、ユーザ・インタフェースは、テスト・ドライバとしての簡単な確認用のものさえあればよく、考慮すべき事項が大幅に削減される。

実システム事例の調査では、データ構造の複雑さを、非常に複雑なデータ抽出条件としてSQLの検索式で吸収している例も幾つかあり、ER図を木構造を使って色々な視点からながめ、実装設

計時の思考を助けるのも重要な機能であろう。

フラットなエンティティ間の2項関連だけの利用に単純化することによって、支援がかなり容易化されている。これによって、オープンな開発支援ツールで利用可能な標準仕様のコンポーネントの設計時機能としての組み込みも比較的容易になると思われる。

こうすることによって、最も効果の高いパターンの支援だけでも、有効な効果が得られることを期待している。

4. おわりに

本報告では、試作したツールの例に基づいて、木構造ER図の活用方法と、分散コンポーネント構築支援への適用の可能性について述べた。
今後、Java-CORBA による分散コンポーネント実装にターゲットを絞り、パターンの洗練(デザインパターン、フレームワーク、基本処理と業務固有処理の分離方式)を進め、ツール支援方式を具体化していく計画である。

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) 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)

(4) 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