データベース・システム 76－6
（1990．3．12）

ADABAS ENTIREを用いた 意味論データベースの構築とCASEのリポジトリ

末舛史郎　山谷茂

株式会社ソフトウェア・エージー

CASEを「工業製品の生産管理とAUTOMATION」に対応する技術と捉え、プログラムの生産管理システム構築のためのリポジトリはどうあるべきかについて述べる。

リポジトリとしては純粋なリレーショナル・データベースでは不適当であり、リレーション間にあるセマンテックスやリレーション内での非正規形が表現できなければならない。

これらを実現したデータベースとして、オブジェクト指向E-Rデータベースがある。ADABAS ENTIREはこのコンセプトに基づき開発され、システム開発の効率を向上させる。

ADABAS ENTIREはCASEのリポジトリとして用いられ、アッパーCASEツールのベースとして活用される。

The Implementation of the Semantic Database Using ADABAS ENTIRE, and CASE Repository

Shiro Suemasu, Shigeru Yamaya

Software AG of Far East, Inc.

2 - 5 Kanda Surugadai, Chiyoda-ku, Tokyo 101, Japan

We discuss what should be the repository for implementing the production control system of programs, considering CASE corresponding to a technology for the "production control system of industrial products and automation."

A pure Relational Database is not suitable for the repository; it should be able to represent semantics existing between relations and non-normal forms in relations. These can be realized by the Object-Oriented Entity-Relationship Database. ADABAS ENTIRE designed based on this concept will increase the efficiency in system development.

ADABAS ENTIRE is utilized as CASE repository. Further, it is used as a basis for the upper CASE tool.

1 CASEとデータベース

1.1 バックログの増大とCASEツール

CASEツールの発生は自然であり、必然的である。従来からシステム開発業務の自動化ツールや生産性を向上させるメソドロジが提案されてはいるが、多くの場合単発的であり、一貫性をもったものではなかった。この結果、ユーザにおけるアプリケーション開発要求の急激な増大につれて、バックログも増加の一途を辿っている。これらのバックログの解決策として第四世代言語による開発が話題になったのも記憶に新しい。

1.2 システムの生産管理

開発ツールを導入する前に、開発業務についてのメソドロジによる標準化、ルール化が必要である。さらに、誰が開発しても同じにできれば機械化、自動化が可能となる。

コンピュータを使って「ソフトウェア開発」業務を自動化すること、換言すれば、開発業務をどこまで製造業並みの生産管理システムに近づけることができるかが「ソフトウェア開発」業務の成熟度の指標ともいえる。

一般の製造業の生産管理システムは生産性の向上(低コスト)、品質の向上、生産計画に対するフィードバック、リソース管理、ロジステックスの管理、原価管理……などにおける評価などを目的としている。

生産管理のデータベースはこれらの目的を達するため各工程間の作業をスムーズに連動するように構築される。また、将来的に生産方式が変化した場合でも対応しやすい構造でデータベース化されている。

CASEシステムにおいても、CASEツールが必要とする情報を将来の変化に対応しやすい構造でディベロップメント・データベース(以下DDB)として貯えることが最も重要である。何故なら当面採用するCASEの手法が変化もしくは他の手法にとって代わる可能性がある。また、開発・保守に必要なドキュメントも時とともに変化することは充分に考えられるからである。このようなCASEツールの基盤となるDDBを維持管理するのがADABAS ENTIREである(図1)。

ADABAS ENTIREは当社が提供するリレーショナル型DBMS ADABASをベースにしたオブジェクト指向E-Rデータベース(Object-Oriented Entity Relationship DataBase)[1]である。

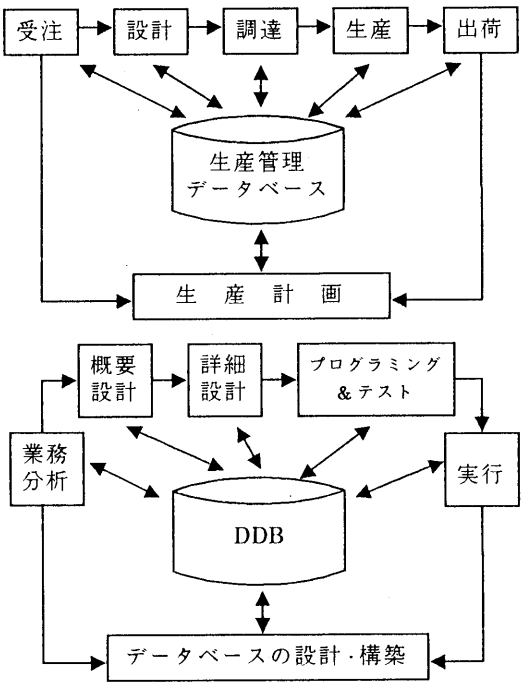
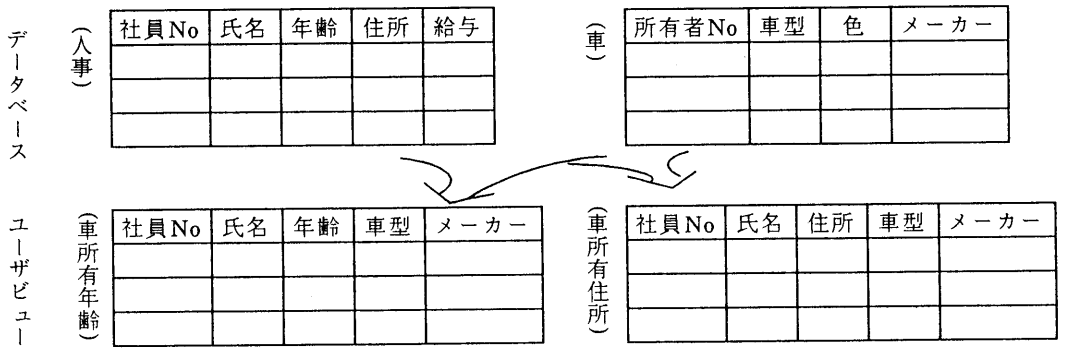


図1. 生産管理システムと
ディベロップメント・データベース

従来のリレーショナル・データベースは第三正規形に準拠したリレーションを平坦な2次元表に展開したもので、ユーザが複数のテーブルを使用するとすれば、ユーザビューとしてビュー定義にその利用条件を定義するか、アプリケーションで処理を記述する必要がある(図2)。

データの独立性を考えた場合、各テーブル間の関連はアプリケーションによって定義づけられるべきであるという考えもあるが、テーブル間に恒久的な関連がある場合は、データの属性として定義し、テーブル間での継承、参照整合性を可能とすることでむしろアプリケーションからの独立を図ることができる。

このような考え方から、テーブル間にある関連(セマンテックス)をそのままエンティティの関連(リレーションシップ)として表現できるようにしたデータベースがADABAS ENTIREである。




```
CREATE VIEW 車所有年齢
        AS SELECT 社員 NO、氏名、年齢、車型、メーカー
        FROM 人事、車
        WHERE 社員 NO = 所有者 NO
```

図 2. データベースとユーザビュー

2. データベースの方向

E-Rモデルとリレーショナル·データベースの関連からデータベースの方向について述べる。

2.1 E-Rモデル

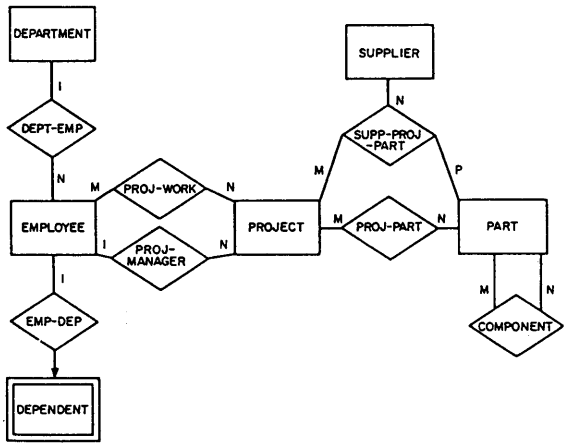
E-Rモデルが発表されたのが1976年である。その後E-Rモデルが指摘された欠点を克服しつつ、拡張E-Rモデルとして数多くのバリエーションを持ち今日に至っている。

リレーショナル·データベースに対応してデータモデルを表現する方法は各種あるが、その中でE-Rモデルが根強い人気を博しているのは、なによりも概念表現の容易性とエンティティがもっている柔軟性(既にエンティティ/オブジェクトという並列表現もある)によるものである。

デザインの工程が概念の整理とコンセンサス作りにあるとするならば、これらの特長をもつE-Rモデルでの作業は極めて自然である。

2.2 E-Rモデルとリレーショナル·データベース

E-Rモデルとリレーショナル·データベースの親和性は第三正規形を実体記述型と関連記述型とに分類しそれぞれをエンティティとリレーションシップに対応させる(図3)ことで、得ることができる。しかし、E-Rモデルでトップダウン·デザインした内容と正規化手法を使ってボトムアップ·デザインした内容は必ずしも一致するとは限らない。



実体記述型
社員 (EMPLOYEE)=社員 No+氏名+職能+…
プロジェクト (PROJECT)=
　プロジェクトコード+プロジェクト名
　+プロジェクト予算+…

関連記述型
プロジェクト工数 (PROJ-WORK)=
　社員 NO+プロジェクトコード+期間
　+工数+…

図 3. E-R 関連図[2)]

2.3 拡張E-Rモデル

E-Rモデルが発表されて以来、エンティティ間には汎化(Generalization)/特化(Specialization)や集約化(Aggregation)/分解(Decomposition)[9),13)]などの関係があることや、実務的には非正規化(Non-normalization)のリレーションとして入れ子のリレーション(Nested Relation)や導出データ(Derived Data)、履歴データ(Historical Data)などの扱いが必要とされることが提唱されてきた(図4)。これらの問題に対して、E-Rモデルは拡張E-Rモデルを作って表現[3)]することで対応を可能としてきた。

これらのうち非正規形の表現形式がデータベースの利用にとって非常に理解しやすく、かつ実世界でも非正規形が通常であるといわれてきた。[4)]

2.4 意味論データモデル

拡張E-Rモデルに必要とされた要件はリレーショナル・データベースに対しても同様に求められる。

リレーショナル・データベースは個々のリレーション間の独立性が高いにもかかわらず、リレーション中に相互のリレーションを関連づける外部キー(Foreign Key)を持っている。この外部キーは2つ以上のリレーションを用いて処理を行う上で必要不可欠なものであるが、参照整合性(Referencial Integrity)という課題を残した。

リレーショナル・データベースにとって、個々のリレーションの独立を実現させることと、リレーション間における結び付き(セマンテックス)あるいは整合性を保証するという二面性が要望されている。このために各種のセマンテック・データベースが発表されている(ユニシスのSIM, ヒューレット・パッカードのIRISなど)。

また、複雑な構造をもつデータベースを統合するために関数型データモデル[5)](FDM: Functional Data Modelなど)が研究されている。しかし普及という面からは —従来のリレーショナル・データベースを使用しているものにとっては— 指定の方法などに難しさを感じる。

実務的な視点や観点からすれば概念の整理の中心がエンティティであり、リレーションシップはエンティティ間の関連を、また属性はエンティティ/リレーションシップの性質、特性を示す。この程度の素朴さが望まれるからである。

2.5 オブジェクト指向エンティティ・リレーションシップ・データベース(以下OO/ERDB)

オブジェクト指向データベースにはまだはっきりと共通認識の得られていない面がある。[6)]しかし、ピータ・チェーンが提唱しているOO/ERDBにはこれらの議論を超えて実世界の認識モデルとしてのE-Rモデルの良さがある。ここに紹介するルールはピーター・チェンが1989年9月に述べたOO/ERDBに関するルールである。

1) すべての情報に関するルール
OO/ERDBの全情報は、エンティティ、リレーションシップ、属性/値(ルール/プロシジャを含む)、システムが制御する一意の識別子という形で表される。

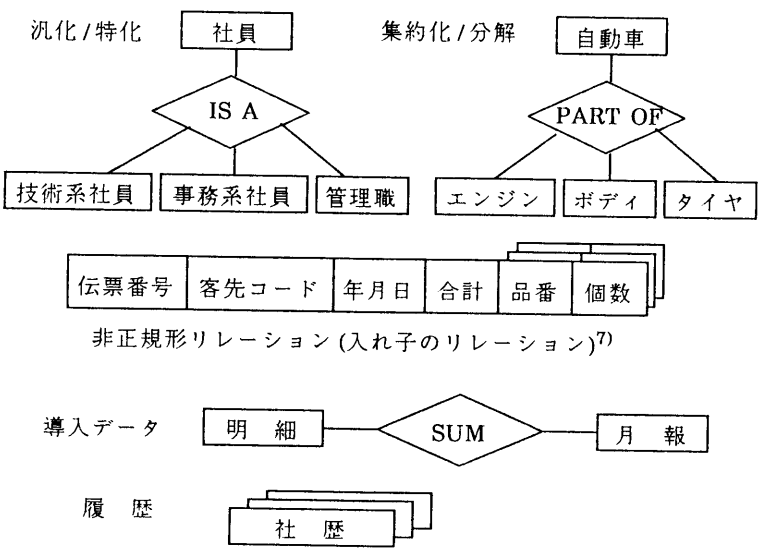


図4. 実務的に必要とされるリレーションの構造

2) システムが制御する
—意の識別子に関するルール
OO/ERDB内でのオブジェクトのオカレンスは、システム自体が制御する識別子で一意に識別されなければならない。

3) 明確なリレーションシップに関するルール
OO/ERDBでは、実世界のエンティティあるいはオブジェクト間に存在するすべてのリレーションシップは明確に表現、保存できるものでなければならない。明確に表現されたリレーションシップの存在は、あらゆるエンティティあるいはオブジェクト·タイプのあらゆるデータ値から独立していなければならない。

4) 汎化·特化、継承に関するルール
OO/ERDBでは、他のオブジェクトからオブジェクトの汎化そして特化が可能でなければならない。このような場合、レベルの低いオブジェクトはレベルの高いオブジェクトの特性を継承しなければならない。

5) 集約に関するルール
OO/ERDBでは他のオブジェクトとそれらの明確なリレーションシップからオブジェクトを合成できなければならない。

6) 自己定義されたデータベースに関するルール
OO/ERDBでは実際のデータだけでなく、実際のデータについてのデータ定義(エンティティ、リレーションシップおよび識別子定義)が不可欠である。

7) 包括的インテグリティに関するルール
OO/ERDBは、エンティティおよびリレーションシップのレベルでインテグリティを維持しなければならない。さらに、ユーザ定義されたデータ·タイプが高度なユーザ定義によるインテグリティを保証することを可能にしなければならない。

8) データ操作に関するルール
OO/ERDBからデータを処理するために、構成あるいは特化オブジェクトのあらゆる属性をベースとして制限を加えることなくSELECTIONとPROJECTIONが使用可能でなければならない。

9) 暗黙の内に得る
リレーションシップに関するルール
ストラクチャ内で展開されたリレーションシップを明確に指定する必要なく、関連のあるオブジェクト·ストラクチャから全オブジェクトの選択が可能でなければならない。

10) 再帰的検索に関するルール
データ定義で指定されている明確なリレーションシップに基づいて、再帰的検索およびシステムが制御する複雑な検索が可能でなければならない。

OO/ERDBは端的に表現すると、今までリレーショナル·データベースの弱点として指摘されてきた、リレーション間の複雑な構造や非第一正規形、再帰型を可能とするデータベースである。それは、実世界中にあるエンティティ/オブジェクトの構造とその関係をそのまま表現することを意図している。

3. ADABAS ENTIRE

3.1 ADABAS ENTIREの基本コンセプト

ADABAS ENITEは前述のOO/ERDBのコンセプトをベースとして開発されており、拡張されたE-Rモデルの他にリファレンシャル·インテグリティの保証やエンティティの再帰構造を可能にしている。ENTIREの名はENTIty RElationshipに由来している。以下にADABAS ENTIREが用いるいつかの用語について述べる(図5参照)。

1) エンティティ—エンティティ·カテゴリ
ADABAS ENTIREでは再帰型のエンティティの定義を許すので「エンティティ·タイプ」ではなく「エンティティ·カテゴリ」という用語を使用する。
エンティティはエンティティ·カテゴリのオカレンスである。例えば、鈴木一郎や佐藤次郎はエンティティであり、名目上エンティティ·カテゴリ社員のオカレンスになる。
それぞれのエンティティはエンティティの名前やエンティティがメンバーになっているエンティティ·カテゴリの名前によって識別される。エンティティは属性によって記述され、他のエンティティとはリレーションシップによって関連付けされる。

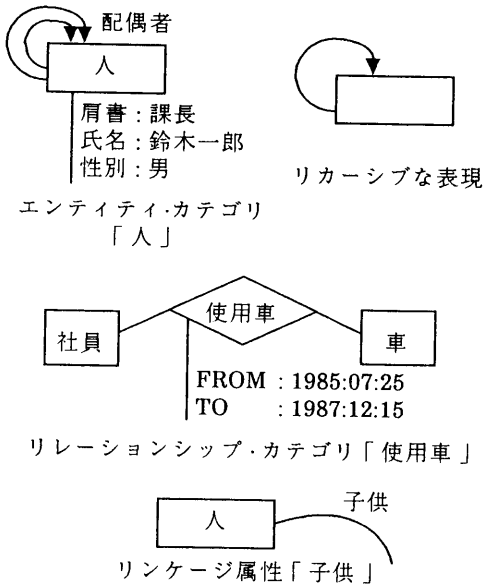


図5. ADABAS ENTIREのコンセプト

2) リレーションシップ— リレーションシップ・カテゴリ

リレーションシップはエンティティ間の関連を表し、属性を持つ。リレーションシップ・カテゴリはエンティティ・カテゴリによって特徴づけられる。例えば、使用車というリレーションシップ・カテゴリは社員というエンティティ・カテゴリのオカレンスと、車というエンティティ・カテゴリのオカレンスをリンクする。そしてFROMとTOの属性をもつ。

3) 属性(アトリビュート)

エンティティとリレーションシップは属性によって記述される。例えば、エンティティ・カテゴリ「人」のオカレンスは「肩書」「氏名」、「性別」、… などの属性をもつ。このような属性を記述型属性という。

また、属性をもたないリレーションシップの存在を許す。このようなリレーションシップに対して、リンケージ属性を設定する。このリンケージ属性を用いてエンティティの関連が設定できる。

一般的に属性は名前と値からなる。例えば、生年月日と対応する1950:05:20から生年月日という属性ができている。これに対し、リンケージ属性の値はそれが指し示すエンティティの名前がそれに当たる。

- 属性カテゴリ

  エンティティとリレーションシップは値をもつ属性と一緒になった時に始めて意味をもつ。ADABAS ENTIREでは、それぞれの属性は1つの属性のカテゴリに属する。各々の属性カテゴリ名(例えば 名前、年齢、登録コード、販売日)と情報のタイプ(例えば名前は32桁の英数字、年齢は3バイトの数字、登録コードは7桁の英数字、販売日は8バイトの数字フィールド)、そしてそれぞれ特別の妥当性ルールをもつ。

- 記述型属性

  名前と値のペアからなりエンティティやリレーションシップを記述している属性。

- グループ属性

  いくつかの属性を集めて一緒にし、1つのグループとしてグループ属性を定義することができる。これにより、属性間の階層構造が表現できる。また、ピリオディック(グループのくり返し構造)をもつ属性を可能としている。

- リンケージ属性

  リンケージ属性はエンティティへのポインタである。E-Rダイヤグラムでは二つのエンティティを結びつける曲線のアークで表現される。

3.2 ADABAS ENTIREの利用

以上述べたようにADABAS ENTIREはOO/ERDBのルールをカバーしていることがわかる。したがってOO/ERDBで表現できるような世界について、ADABAS ENTIREを用いてデータベースを構築することが可能である。

OO/ERDBの利用例としてCASEリポジトリと、E-Rモデルを用いてデザインするアッパーCASEツールへの応用がある。

4. ADABAS ENTIREの利用分野

4.1 CASEリポジトリ (Repository)

ソフトウェア・エージーではCASEリポジトリをデベロップメント・データベースと呼んでいる。

当社のプロダクトPREDICT CASEでは図6のような開発過程で作り出されるエンティティを想定し、これらのエンティティを図中に表現されているような関連で表現している。これらのものはISOTEC注)の手法を使用した場合の標準的なエンティティとリレーションシップを示している。この図でも明らかなように、開発中に生産される生産物をエンティティとして捉え、これらの生産物の関連をリレーションシップとして設定することができる。

ADABAS ENTIREはPREDICT CASEにおいて、E-Rモデルで表現される各エンティティとエンティティ間に設定されたリレーションシップを管理する。

4.2 PREDICT CASEにおける主要エンティティ/オブジェクト

PREDICT CASEのデータ構造はE-Rモデルがベースであり、ADABAS ENTIREが使われている。開発で対象とするエンティティ/オブジェクトの1例として、PREDICT CASEが設定するエンティティ/オブジェクトの一部と関連について述べる(図7参照)。

1) プロジェクト
アプリケーション・システムの最上位の機能で、サブ・プロジェクトに分割定義できる。

2) 機能(ファンクション)
最終的な目的を達成するための活動を意味する。特に次の2つのオブェクトを扱う。
- 業務内に格納されている情報(エンティティ・タイプ)
- 業務内でやりとりされる、あるいは業務とその業務外との間でやりとりされる情報(データ・フロー)

3) データ・フロー
データ・フローはデータ項目の集合であり、あるコミュニケーション・タイプの「コミュニケーション媒体」を介してアプリケーション・システムのある「機能(ファンクション)」により入力/出力される。

4) コミュニケーション媒体
送り元から受取側のコミュニケーション・パートナーにデータ・フローを送るための仲介役。例えばDPコミュニケーション・ネットワークや電話、用紙、会議などがある。

5) エンティティ・タイプ
同一特性を持つデータの集合体で、データ・エレメントによりその特性が表現される。

6) リレーションシップ
2つのエンティティ・タイプ間の結合(関係)を表す。

7) データ・エレメント
アプリケーション・システムが使用するデータの最小論理単位である。データ・エレメントは、エンティティ・タイプの特性、データ・フローのデータ・フィールド、あるいは特性とデータ・フィールドの両方として働く。

8) データ・エレメント・タイプ
1つあるいは複数のデータ・エレメントに割当てるもので、データ・エレメントの長さ、フォーマット、値の範囲を記述する。

9) 外部パートナー
アプリケーション・システムの外部の世界を表し、当アプリケーション・システムにデータを送ったり、逆に送られたりする。データ・フローの送り元、あるいは受取り先を示すもので、例えば以下がある。
- 人あるいは組織
- 当アプリケーションに属さないファンクション(機能)
- 当アプリケーションからサービスを受け取る、あるいはサービスを渡す他の部門

10) リソース
機能(ファンクション)を実行するために必要な資源。例えば、ホスト、PC、端末、プリンタなどがある。

4.3 主要エンティティ/オブジェクトの関連

1) 情報構造分析
機能構造に対応した形でエンティティ・タイプが洗い出される。次にエンティティ間にあるリレーションシップを設定する。エンティティがもつ属性をデータ・エレメントに、またデータ・エレメントのタイプとしてデータ・エレメントの特性を定義する。

注) 西独のコンサルテーション会社EDV Studio Ploenzkeのシステム・デザイン手法

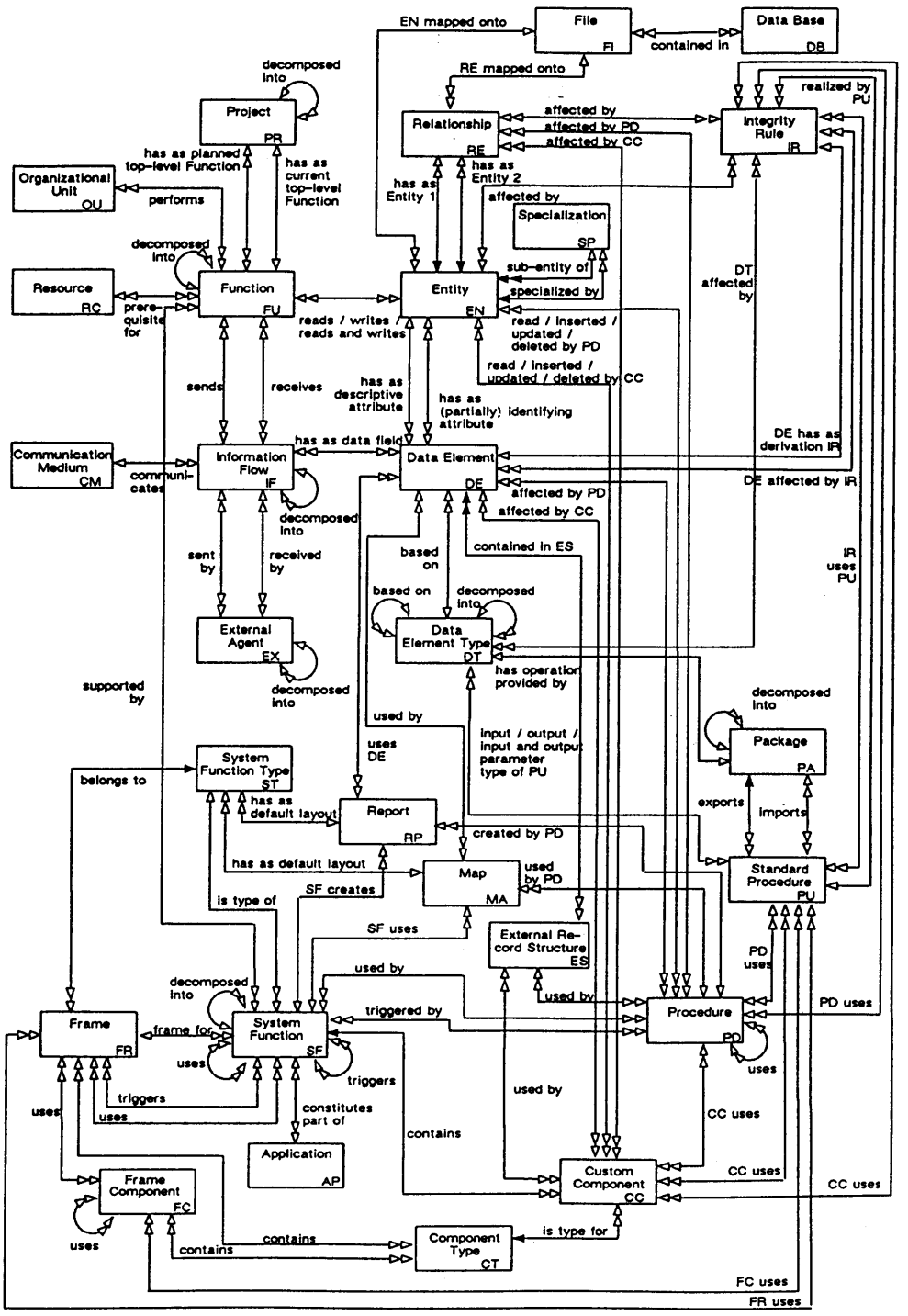


図6. PREDICT CASEのレポジトリ内の全エンティティ

2) 機能構造分析

対象となる機能に対してトップダウン的に機能を分析し、それぞれの機能で使用されるデータフローを定義していく。

情報構造と機能構造は同じ詳細レベルで並行して分析していく。

5. アッパー CASE への適用

トップダウン・デザインで作成する概念データモデルを E-R 図で表現するということがよくある。例えば、組織分析から各組織単位に機能を洗い出し、さらに各機能ごとにエンティティを洗い出すといった手順で設計がなされる。

この時、トップダウン的に把握されるエンティティに対する属性の構造やエンティティ間のリレーションシップの形が問題となる。ごく自然にエンティティをあるがままに分析すれば、すでに述べたように属性がグループで扱われたり、あるいは属性値が繰り返しの構造をとる(リレーションの入れ子の構造の)もの、関連

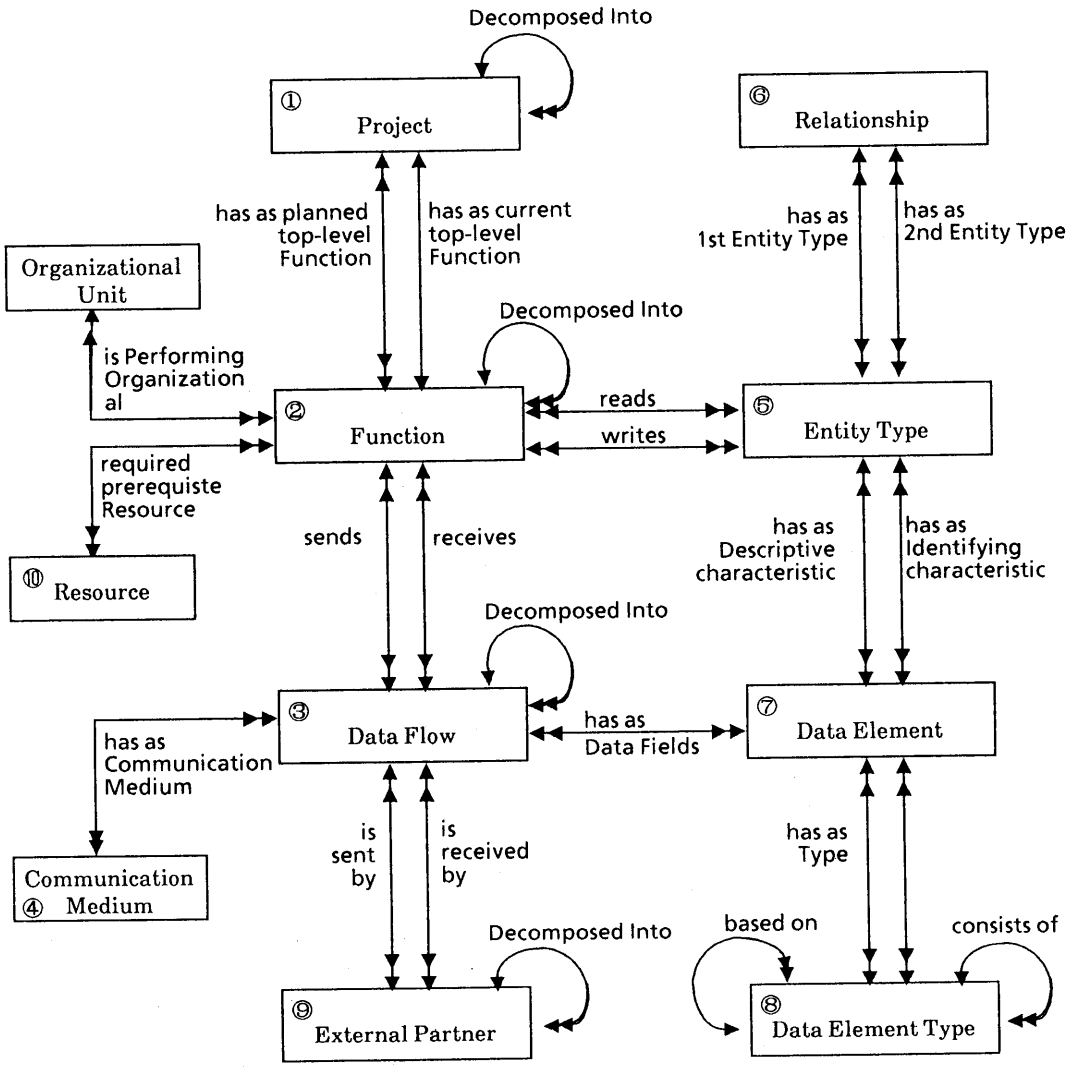


図7. PREDICT CASE レポジトリの主たるエンティティ

に属性をもたないものなどが現れる。

これらのエンティティの構造は平坦な2次元表のリレーショナル·データベースにとってそのまま実現することはできない。このため、リレーショナル·データベースを前提とした開発は上記のエンティティに対して、データの独立以外の目的で正規化を実施する必要が生じる(第一正規化、第四正規化)。また、エンティティ間の関連(セマンテックス)に対してもプログラムで工夫することが要求される。

このように、トップダウン·デザインの内容をそのままリレーショナル·データベースへ展開はできなかった。これに対してADABAS ENTIREはADABASというリレーショナル(型)データベースをベースとした拡張E-Rモデルをサポートしている。ADABASは従来から非正規形のリレーションの構造を可能にしている。例えば、グループ構造、グループでの繰り返しフィールド(ネステドリレーション)、複数の値をとるフィールドなどがある。したがって、拡張E-Rモデルを使った分析からそのままデータベースの設計フェーズまでADABAS ENTIRE(実際にはPREDICT CASE)を用いて設計することができる。

6. おわりに

今回はCASEとデータベースの関連からCASEを実現するためのデータモデルはどうあるべきかを中心に議論した。そのなかでオブジェクト指向E-RデータベースADABAS ENTIREを紹介した。ソフトウェア·エージーではリポジトリだけでなく、次のようなCASEツールを開発している。

1) PREDICT CASE
システム開発上流工程を受けもつ設計ツール。各種ドキュメントも出力される。

2) PREDICT CASE WORKSTATION
システムやデータベースの設計をワークステーション上でDFDなどの図形入力により行うことができる。

3) NATURAL CONSTRUCT
第四世代言語NATURALのユーザ·プログラム·ジェネレータ

これらツールの開発言語は第四世代言語NATURALを用いている。したがって、上記ツールはNATURALが稼働するすべての機種、OS、OLTP上で利用できる。開発規模が大きくなった場合、他の環境への移植が問題となるが、これらのツールは複雑な開発環境下においても充分に対応できる。これが我々のめざす第四世代テクノロジの方向である。

最後に本稿を書くにあたり資料の提供など協力を頂いた弊社吉舗紀子氏に感謝いたします。

参考文献

1) Parker Hodges : A Relational Successor?, *Datamation*, November 1, 1989
2) Peter, Pin-shan Chen : The Entity-Relationship Model ; toward a Unified View of Data, ACM *TODS*, Vol. 1, No. 1, March, 1976
3) 西山智、小花貞夫 : ERモデルに基づくデータベース設計支援システムの提案とそのERエディタの実装、データベース·システム 67-2, 1988
4) Kitagawa, H., Kunii, T. L. and Ishii, Y. : Design and Implementation of a Form Management System APAD Using ADABAS / INQ DBMS, *Proc. COMPSAC*, 1981
5) 清木康、加藤和彦 : 関数型計算モデルのデータベース処理への適用、情報処理、Vol. 29, No. 8
6) Catriel Beeri : Formal Models for Object Oriented Databases, *Deductive and Object-Oriented Databases*, December, 1989
7) 石井義興 : 三次元データベース管理システム、データベース·システム 69-8, 1989
8) 味村重臣、山田進、堀内一 : データベースシステムの設計と開発、1983, オーム社
9) 酒井博敬 : 情報資源管理の技法、1987, オーム社
10) ソフトウェア·エージー : ADABASデザイン·コース テキスト
11) Software AG : *ADABAS ENTIRE - Concepts and Facilities*
12) Software AG : PREDICT CASEマニュアル
13) Smith, J. M. and Smith, D. C. P. : Database Abstraction : Aggregation and Generalization, ACM *TODS*, Vol. 2, No. 2, 1977