ERwinインサイダー™

& BPwin インサイダー™

ERwin & BPwin

ユーザーへの

チップス、裏ワザおよび一般記事

世界各地の ERwin & BPwin ユーザーから

編集 Ben Ettlinger

著者

Ben Ettlinger　Data Administrator
New York Power Authority, White Plains, NY
President
New York Enterprise Modeling User Group

Gary Gramm　Consultant, DataSmart Business Solutions Inc.
Vancouver, BC Canada

Karen Lopez　ListMistress, ERwin Users Discussion Group
InfoAdvisors, Inc.
www.infoadvisors.com
Toronto, Ontario Canada

Lucie S. Johnson　Consulting System Engineer
Bank of America, San Francisco

Doug Stone　Information Modeler
NUSCO (NU Service Co.) Berlin, CT.

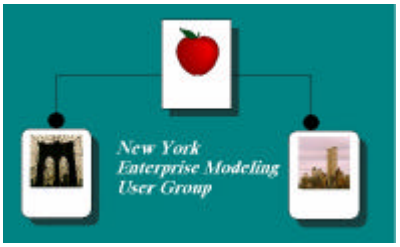
翻訳

Japan Enterprise Modeling User Group（**日本** ERwin **ユーザー会**）

監修

松本 聰

（註）このドキュメントの中に記載された内容に関しては、一部正式にサポートされてない部分が含まれています。したがって、日揮情報ソフトウェア㈱にサポートを依頼されても、サポートはできません。読者の皆様の責任範囲で実行していただくようお願いいたします。



c/o Ben Ettlinger
10 Overlook Terrace
New York, NY 10033-2268

Satoshi Matsumoto
2-11-24 Tsukiji, Chouou-ku
Tokyo, Tokyo 104-0045
Japan

はじめに

ここ数年で,ERwin は世界のモデリングツールのリーダーとしての地位を確立しました。その使いやすさと機能は、データベース管理者、データ管理者、データモデラーに対して、リレーショナルデータベースのすべてのプラットフォームと定義についての構築と保守に有効な基盤をもたらしました。

すべてのツールは、標準的に規定された使用方法と方法論を定めています。ある意味では、ソフトウェアのライセンスは、自動車を持つことと違っていません。

自動車メーカーはドライバー・マニュアルを提供します。しかし、同じ自動車の他の所有者とかメカニック、自動車のクラブ・マガジンおよびネットワーキングなどからあなたが学習する内容は、実際のきれいで、すてきな効率的な使い方およびある意味でのトリック（裏技）ではないでしょうか。この出版の目的はまさにここにあります。 データモデラーや管理者がより良い仕事をするための手助けやフランクな意見、アイデアの交換の場として有効なものにしたいと考えています。

わたしたちは、コンピュータ・アソシエイツより、この出版に参加することを依頼されました。しかし、この出版は、私たちがエンタープライズ・モデリング・ユーザ・グループを運営するのと同じ方法で行います。これらのグループは、CA から組織的な支援を受けますが、独立したエンティティとして存在しています。

IDEF1X 専門用語でいえば、ユーザー・グループおよびこの出版は、CA と「非依存の関係」を持っているといっていいでしょう。私たちは自分の識別性、独立性を持っています。

ただし、この出版に当たって、コンピュータ・アソシエイツのステーシー・ジェンセンが私たちに与えてくれた熱心な支援とエネルギーについては､よく認識してほしいと思っています。

また、私がこの記事を書き、調査をすることおよびニューヨーク・エンタープライズ・モデリング・ユーザー・グループの活動に熱烈な支援を与えてくれた、NYPA の CIO ラッセル・クラウスおよびアプリケーション・ディレクターのピーター・ポギーにもお礼の言葉を述べます。

ベン・エトリンガー
Data Administrator
New York Power Authority

President, NYEMUG
New York Enterprise Modeling User Group

ERwin キーボード・ショートカット

Ben Ettlinger
Data Administrator, New York Power Authority
New York Enterprise Modeling User Group

Key Stroke	Function
Cntl A	すべて選択
Cntl B (論理モデル)	独立属性ブラウザ
Cntl B (物理モデル)	独立カラムブラウザ
Cntl C	Window へ
Cntl S	セーブ
Cntl T	Erwin ツールボックス
Cntl X	Window の削除
Cntl V	ペースト
Cntl +	ズーム・イン
Cntl -	ズーム･アウト
Cntl *	拡大しない
Cntl ↑	論理モデルへ
Cntl ↓	物理モデルへ
Cntl →	右へシフト
Cntl ←	左へシフト
Cntl insert	Window のコピー
Cntl shift→	右へシフト
Cntl shift ←	左へシフト
Cntl shift ↑	ダイアグラム上へ移動
Cntl shift ↓	ダイアグラム下へ移動
Cntl alt ↑	ダイアグラム上へ移動
Cntl alt ↓	ダイアグラム下へ移動
Cntl alt →	右へシフト
Cntl alt ←	左へシフト
Cntl home	ダイアグラムのトップへ
Cntl end	ダイアグラムのボトムへ
Cntl shift F6	拡大しない
Cntl shift F4	ダイアグラム window のクローズ
Cntl shift enter	ハイライトオブジェクトのエディタのオープン
Cntl alt enter	ハイライトオブジェクトのエディタのオープン
Cntl enter	ハイライトオブジェクトのエディタのオープン
Cntl shift delete	ハイライトオブジェクトの削除（ウォーニング）
Cntl alt delete	ハイライトオブジェクトのエディタのオープン
Cntl delete	ハイライトオブジェクトのエディタのオープン
Windows icon	スタート・メニュー
F1	オンライン・ヘルプ

Model Mart キーボード・ショートカット

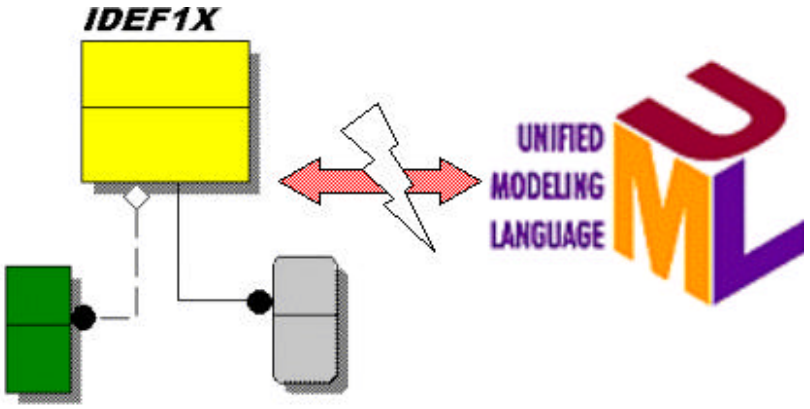
Ben Ettlinger
Data Administrator, New York Power Authority
New York Enterprise Modeling User Group

Key Stroke	Function
Cntl shift B	レポート・ブラウザ
Cntl shift F	モデルマート・ダイアグラムを window としてセーブ
Cntl shift L	モデルマート・ログ
Cntl shift Q	Window へクエリ送る
Cntl shift V	容量計算エディタ
Cntl shift Z	モデルマート・タイミング
Cntl shift➔	右へシフト
Cntl shift ←	左へシフト
Cntl shift ↑	ダイアグラム上へ移動
Cntl shift ↓	ダイアグラム下へ移動

これらの機能に関しては、技術サポートは行っておりません。コンピュータ・アソシエイツおよび日揮情報ソフトウェアでは一切のお問い合わせに対して、対応いたしませんので、ご了解ください。

IDEF1X VS UML

比較分析



By: Ben Ettlinger
Data Administrator
Information Technology Division
New York Power Authority

訳：松本 聰
JNT システム㈱
IRM グループ シニア・コンサルタント

IDEF1X VS UML – 比較分析

1．なぜ比較するか

なぜ、この分野で誰がこのようなことを行いたいと思うのでしょうか。なぜ、UML、オブジェクト指向でのモデルリング言語としてのハイブリッド言語と、IDEF1X 最もポピュラーなリレーショナル・データ・ベースのモデリング言語を比較するのでしょうか。

リレーショナル、そしてオブジェクト、異なる 2 つのパラダイム？異なる 2 つのアーキテクチャー？異なる 2 つの記法？…リンゴとオレンジ…どっちが正しいか。ほんとに？

確かに、オブジェクトおよびモデルリング言語は、リレーショナル・パラダイムの単純性を超えて概念をカプセル化することができるという特徴を持っている。

オブジェクト・データ・ベースは、リレーショナル・データ・ベース環境では想像できなかった、あるいは想像したかった（けどできなかった）ビジネスルールおよび機能を示すことが可能である。

もし、オブジェクトがより大きな視覚的な詳細で「システムの静的構造および動的な振る舞い」[1] を捕らえる能力を提示するのであれば、なぜ、わざわざ「古い」技術をふり返って見る必要があるのでしょうか。答えは単純、単純性。

オブジェクト・パラダイムは複雑である。継承、カプセル化、ポリモフィズム、ステレオタイプ、リアリゼーションなどは理解が非常に難しい概念である。モデル化することが困難であり、人に説明しようとしても本当に難しいものである。

考慮すべき別の重要な要素はなんでしょうか。実は、ほとんどのビジネスアプリケーションはリレーショナル・データベースを使用して、十分に構築することが可能であるということである

最近の CA-World で、このトピックをカバーするプレゼンテーションのあと、大手の通信販売会社のデータ管理者に真面目に質問された。なぜ、皆さんはオブジェクト・データ・ベースを構築したいのでしょうか。

彼女の会社は、顧客が計測数値を入力することによって、衣服が合っているかどうかが分かる、最新の大きなウェブ・ページを備えた e-コマースを含むシステムを着実に実行してきた。使用しているリレーショナル・データベースは素晴らしくそれを支援している。

2 つのパラダイムの比較についての私のオリジナルの関心はオブジェクトの複雑性によって作られた。私は、私が参照することができ、比較することができ、分析することができる、それから理解することができる参照ポイントを必要とした。この記事は、UML をハンドリングするために私が使用したプロセスを表わす。

---

[1] The Unified Modeling Language Reference Manual p.3 James Rumbaugh et al 1999 Addison Wesley

私は、それが私の単なるもがきであるとは思っていない。私が出席したすべてのユーザ・グループのミーティングあるいはカンファレンスで、いつも 2 つの質問をする。
1 番目。「あなたは、UML を聞いたことがありますか。」ほとんどすべての手が上がる。
2 番目。「それが何か知っていますか。あるいは、それを理解していますか。」
ほとんど手は上がらない!
しかし、比較は実際には参照ポイントを越えるものである。オブジェクトからリレーショナルの世界へデータをマッピングする、実際の必要性があった。
Component Strategies 誌の 1999 年 4 月号の記事で David Linthicum が、純粋オブジェクト・データベース側から、非常に詳しく議論している。私たちは、最近のハイブリッドまたはユニバーサル・データベースの成長を見ている。これらは、リレーショナルおよびオブジェクト･データベースを横断的に繋ぐ知的なミドルウェアとしてのミックスされたパラダイムとツールを持っている。
Linthicum によれば将来の方向性として、リレーショナルとオブジェクトの横断的な必要性はさらに増加するといわれる。「OO データ・ベースの需要が上昇し、リレーショナル・データベースの人気が残り、開発者はオブジェクト・リレーショナルのトランスレーション・レイヤによって応急処置を求める。」Linthicum の結論は、このユニバーサル・データベースはオブジェクトとリレーショナル統合であるという。

2．なぜ、ERwin とParadigm Plus か

リレーショナル・パラダイムおよびオブジェクトパラダイムのこの分析で、私たちはリレーショナルおよびオブジェクトのモデリングツールを使用する。
リレーショナル・サイドでは、コンピューター・アソシエーツのデータモデリング・ツール ERwin を使用して IDEF1X モデリング言語をベースとしてスクリーン・ショットで表示する。
オブジェクト・サイドでは、Paradigm Plus を使用する。このツールは CA が提供する UML（統一モデリング言語）のモデリング・ツールである。
これらのツールを使用する理由は単純である。
ERwin は最も広く用いられている IDEF1X のモデルリングするツールである。このツールは、私が最も快適に使っているものである。
また、PLATINUM（今の CA）から Paradigm Plus を入手した。Rational Rose、もっともポピュラーな UML ツールは使わなかった。

3．ER ダイアグラム対クラス・ダイアグラム

IDEF1X はそのパレットとして、1 つのダイアグラム（エンティティ・リレーションシップダイアグラム）を持っている。

UML は、アプリケーション開発に当たってのより大きな範囲（ハイ・レベルのビジネス工程分析からアプリケーションの配備まで）を含むより強健な 8 つのダイアグラムを含むパレットを持っている。

図 1 は、UML がどのようにシステムを構築するかを示したものです。上流の USE CASE ダイアグラムからパッケージ･ダイアグラムに示されたハードウェア・コンポーネントおよびソフトウェア･コンポーネントの関連までを含み、かつそれぞれに構築され配備されたビジネス・アプリケーションを開発する。クラス・ダイアグラムは UML の中心および基本である。

クラス・ダイアグラムは、そのリレーショナル相当としてエンティティ・リレーションシップダイアグラムがもたらすものと同じように、ビジネス・ルールと要求を記述するやり方としてデータベースのオブジェクトをマッピングする。

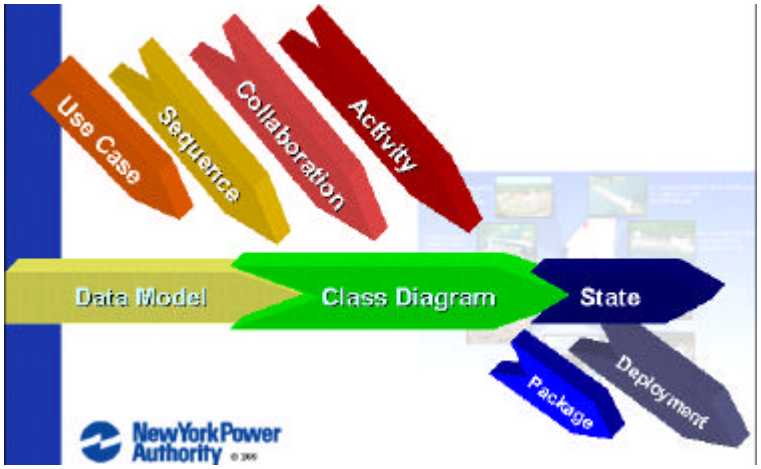


図 1

この議論は、2 つの方法論の類似、2 つのダイアグラムの構築およびビジネス・ルールの表現の比較に集中される。つまり、エンティティ・リレーションシップダイアグラム（データ・モデル）およびクラスダイアグラム。

リレーショナル・パラダイムとオブジェクトパラダイム間のトランジショナル･レイヤ、ここがポイントである。

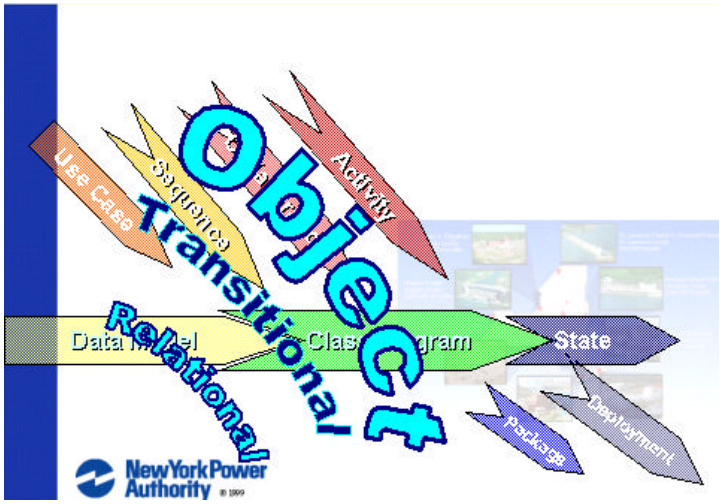


図 2

4.エンティティ対クラス

表面上、エンティティとクラスは類似している。それぞれは、それぞれのパラダイムの基礎である。双方はモデルの中で最低のレベルの要素、つまり属性のコンテナーである。実際、エンティティは単純なタイプのクラスと呼ぶことが可能である。

クラスは「類似した構造、振る舞いおよびリレーションシップとの 1 セットのオブジェクトのためのディスクリプタ」[2]といえる。

クラスは、すべての属性とオペレーションが付けられた構造を持っている。オブジェクトモデリングは、データモデラーの見方に、データを備えたプロセスのモデルを融合することである。ここに大きな大きな違いがある。

オペレーションまたはメソッドは振る舞いに影響するクラスが任意のオブジェクトに要請することが可能なサービスである。[3]クラス内にカプセルに入れられた、手続き的なコード・セットである。それらは、クラスの振る舞いに影響する、属性上でオペレーションを実行する。つまり、エンティティはオペレーションのないクラスと呼ぶことができる。

一見したところ、クラスの中へのオペレーションの包含は、リレーショナル・エンティティとの大きなそして劇的な違いのように見えるかもしれない。しかしながら、エンティティはそれ自体は実

[2] p. 42 The Unified Modeling Language Reference Guide by Rumb augh et. al. 1999 Addison Wesley
[3] p. 49 The Unified Modeling Language User Guide by Grady Booch et. al 1999 Addison Wesely

際にはカプセル化はできないが、リレーショナルの世界ではストアド・プロシジャー、これはオペレーションに非常に似た機能を持っている。この部分は、この記事の後のほうで詳しく述べる。

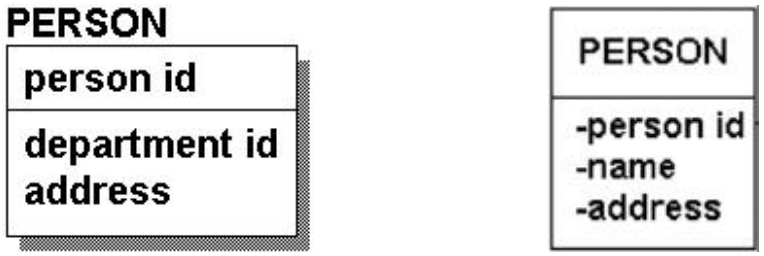
クラス中の振る舞いのオペレーションの存在は、エンティティが直面しない問題とすることができるかもしれない。オペレーションが他のクラス中のオペレーションによってさらに要求されたある属性を要求し、同じ構造の中での属性およびオペレーションのコンビネーションを作ることが可能である。

したがって、1つ以上のクラス中の属性を複写する傾向がある。これは冗長性を作る。冗長性は、つまり非正規化はビジネス・ルールを見えにくくしてしまう。

この真の理由によって、リレーショナル・モデルの単純性と一緒だといえる。

多くの人々は、オブジェクト・リレーショナルや純粋オブジェクト・モデルを作る前にリレーショナル・モデルを作成することが本質的であると考えている。

グラフィカルに見れば、IDEF1Xのエンティティ／テーブルおよびUMLクラスはほとんど同一である。



IDEF1X エンティティ/テーブル Table　　UML Class

図3

5. リレーションシップ対アソシエーション

IDEF1Xでは、リレーションシップは、黒丸（肉団子）を持った、実線あるいは破線によって表わされる。実線のラインは依存関係を表わし、破線は非依存関係を表わす。違いは、リレーションシップのキーがどこに置かれるかである。

送られたキー（すなわち外部キー）が子のキーの一部になる場合、関係は依存している。複合キーの一部になる。これは、親から子へ移行するキーが子の識別（キー）の一部になることを意味する。

移行されたキーあるいは外部キーが子エンティティ非キー属性になる場合、リレーションシップは非依存である。

依存、非依存リレーションシップに付いては、この後ダイアモンド記号とNULLとタイトルをつけられたセクションでさらに議論する。キー・マイグレーションは、オブジェクト・パラダイムがクラスを識別する自然主キーを使用しないという、UMLには存在しない概念である。

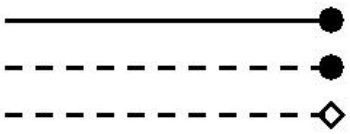
UML では、単独でオブジェクト id（OID）と呼ばれる、代理キーを使う。したがって、UML では、クラスとクラスのリレーションシップまたはアソシエーション（UML ではこう呼ぶ）に実線のライン持っている。
註：他のタイプのリレーションシップが UML の中にはある。少しは、このドキュメントの後半のセクションで説明する。他のものに関しては議論の外とする。

依存および非依存のリレーションシップのオプションを供給ができることは、UML およびオブジェクト・パラダイムに対して IDEF1X およびリレーショナル・パラダイムが持つ長所である。自然キーあるいは代理キーの使用について、リレーショナル・データベース・デザイナーはその選択権を持っている。

自然なキーは、他のエンティティとの関係を直接に作成する代替を示すことによって、重要な貢献を提供する。またそれは、プログラマにとってより容易な階層的ロード・マップでもある。

自然キーが下へ移行されるとき、SQL クエリが作成される場合、階層的にはいくつかの JOIN が要求される。自然なキーは、さらに複製情報を含んでいる列の数を減少させることを支援する。

依存リレーションシップは非 IDEF1X 世界では移動可能リレーションシップとも呼ばれる。



IDEF1X

UML アソシエーション

依存および非依存リレーションシップ

図 4

6. キー対 OID

オブジェクト・パラダイムでの OID の使用は、代理キーを制限し、IDEF1X と UML の間の重大な差をさらに大きくする。これらは、後のセクションで議論する。
自然キーと代理キーの議論は、技術的な出版およびウェブ・フォーラム中で沢山の注意が与えられた。OID のオブジェクト・パラダイムでの使用は、データ冗長性に関する重大な意味合いを持つことができるということであり、リレーショナル・パラダイムとの重要な違いを強調する。
他方で、OID は、正規化のより大きなレベルを達成し、かつ OLTP からデータ・ウェアハウスまでのより容易な変化に対してより容易なパスを提供する。さらなる議論はこの研究の範囲外である。

下図で気がつくように、OID はオブジェクト・クラスにおいては目に見えません。属性、person_id、committee_id、「ユーザ」がクラスの特別のインスタンスを識別するように、それらはこれらのクラスに加えられる。これらはユニークにクラスのインスタンスを識別するためにデータ・ベースで内部的に使用されるキーではない。

これは、C. J. Date が「もっともらしい」[4]と呼ぶ、状況を作ります。これは起きることができるから、別個の 2 つのオブジェクトはすべてのユーザにおいて同一かもしれない。つまり、互いの複写ということである。 また、OID によって識別できるか。 ユーザはどのようにして外部的にそのような 2 つのオブジェクトを識別することができるか。[5]

7. カーディナリティ対マルチプリシティ

カーディナリティとマルチプリシティはリレーションシップの終わりに現われるインスタンスの数を意味するために使用される用語である。カーディナリティはリレーショナルの世界の中で使用される用語であり、マルチプリシティはオブジェクト世界の中で使用される用語である。

IDEF1X のテキストの著者であるトマス・ブルースは、カーディナリティを「リレーションシップの終わりに参加するか参加するに違いないエンティティインスタンスの数のステートメント」[6]として定義している。

一方、Rumbaugh によれば、カーディナリティは誤称である。カーディナリティはサイズを意味する。しかし、マルチプリシティは 1 セットの数を意味する。Rumbaugh は次のように述べる。「カーディナリティという用語はいわゆるマルチプリシティを意味するために多くの著者によって誤用されている。しかし、カーディナリティは数学的定義としては、数として表されます。ただし、それは一連の数ではない。」[7][8]

リレーショナルの著者はカーディナリティを使用し、オブジェクトの著者はマルチプリシティ[9]を使用する。 遷移のレイヤのためには、リレーショナル、オブジェクト、リレーショナル･オブジェクトの開発者は、これらが同じものであることを認識しなくてはならない。

---

[4] p. 638 An Introduction to Database Systems (6th Edition) by C.J. Date 1995 Addison Wesely

[5] ibid

[6] p. 528 Designing Quality Databases with IDEF1X Information Models by Thomas Bruce 1992 Dorset House

[7] p. 183 The Unified Modeling Language Reference Guide by Rumbaugh et. al. 1999 Addison Wesley

[8] UML 関連の本の中で 2 つの用語の議論の包括的なレビューは、Scott Ambler による「Software Development Magazine」June 1999. (www.sdmagazine.com) を参照。

[9] いくつかの例を見ると面白い。「Oracle Press' Oracle 8 Design Using UML Object Modeling」Dr. Paul Dorsey、Joseph R. Hudicka, 1999 McGraw-Hill によるオブジェクト・リレーショナルの本によれば、カーディナリティという用語が排他的に使用されている。マルチプリシティはこの本では現れない。de Champeaux などによる「In Object Oriented System Development」1993 Addison Wesley 同様にカーディナリティが使用されている。私は次のように考えている。「オブジェクト・リレーショナルでの用語は自由に両方の用語を交換しながら、取捨選択的になる。」

明確に言えば、IDEF1X のリレーションシップを参照する場合には、カーディナリティを UML のリレーションシップ（アソシエーション）を使用する場合にはマルチプリシティを使用する。

7.1 IDEF1X のカーディナリティ

リレーショナル/IDEF1X の中でカーディナリティは単純である。カーディナリティは、リレーションシップが常に 1 つから「何か」へのリレーションシップである際に表現される。1:N または 1:1~N など。

図 5 は、IDEF1X がどのようにグラフィカルにカーディナリティを表わすかを示す。

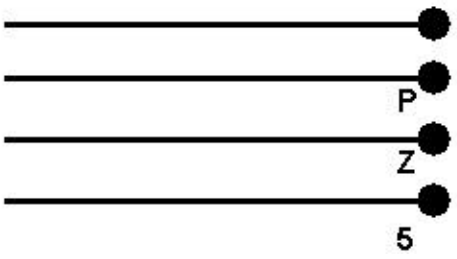


図 5

リレーションの横に、1 のカーディナリティを示す記法はない。リレーションシップ・ラインの反対側に、カーディナリティ「多（N）」を示す黒丸。数字あるいは文字は、特定の値「N」の数を特定する。

文字「P」は 1~N を示す。したがって、2 番目のリレーションシップは 1 対 1 もしくはそれ以上のリレーションシップを示す。文字「Z」はゼロまたはそれ以上を示す。したがって、図 5 の 3 番目のリレーションシップは 1:0 or N を示す。番号は固定の値を示す。図 4 の 4 番目は、1:5 のリレーションシップを示す。

私は、IDEF1X の数値でなされるアルファベットの表現が、若干の不便であることを見つけた。また、私はそれらがごちゃ混ぜになったり何を意味するかを忘れ易いことも見つけた。カーディナリティの値が頻繁に使用されない場合が特にそれに当たる。次のセクションの中で示されるように、UML の記法はこの点に関してもっとクリアであり、数値を表わすためにアルファベットに依存する必要がない。

これらの表現は、UML との比較のセクションでより詳しく議論される。

7.2 UML のマルチプリシティ

マルチプリシティは非常に正確な、従って非常に複雑な表現が可能である。

データ・ベース・デザイナーは、インスタンスのセット数、インスタンスの範囲およびインターバルおよび数値が単に増加する場合など、インスタンスの特定の数値を持っている。

図 5 の中で示されるように、クラスのインスタンスの固定のセットは、そのセットの中でのインスタンスの数を表わす 1 つの値によって示されます。
図 6 の最後の例で示すように、範囲は、2 つの数の省略形、始めの数と終わりの数で示される。
UML のマルチプリシティは、さらに間隔および数の範囲のコンビネーションを示すことができる。

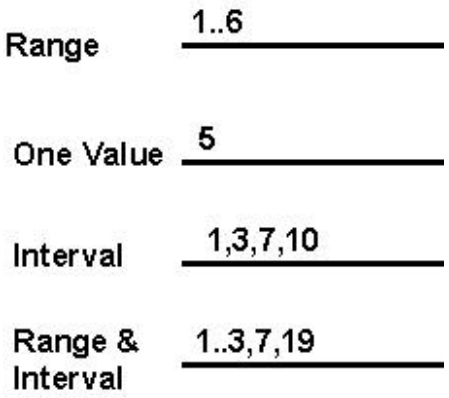


図 6.

図 7 に示されるように、UML の長所はアソシエーションの両側に付けられたマルチプリシティを表わす能力である。これは、複雑な関係（Complex Relationships）とタイトルをつけられたセクションで議論される非常に複雑なビジネス・ルールの表現を表わしている。

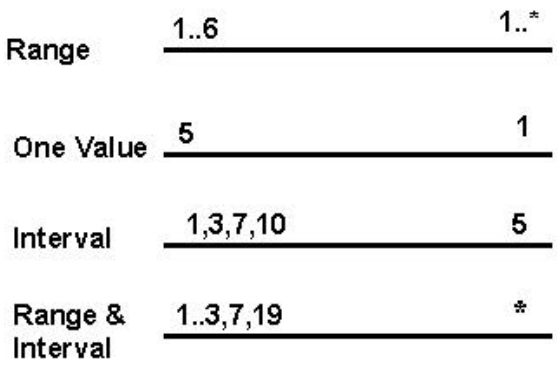


図 7

アソシエーションの両側でのマルチプリシティの値は、始めは、混同することが多い。私は、両刃の剣の上の上に立つ有用な方法を見つけた、それは以下のとおりである。
アソシエーションの横のマルチプリシティの値を見る場合、表現について考える。
例として図 8 を見ると;

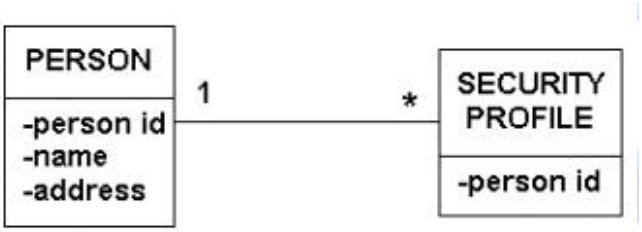


図 8

PERSON の 1 つのオカレンスに対して、SECURITY PROFILE は多数のオカレンスを持つことができる。（*は、多のマルチプリシティを示す。)

IDEF1X での範囲の考え方

連邦情報処理規格 184（Federal Information Processing Standards）（FIPS 184）は IDEF1X ルールを政府の規格として定めている。FIPS 184 の表現によれば IDEF1X ダイアグラムの範囲は言語として有効である。図 9 は FIPS 184 セクション 3.5.2 の一部であり、範囲が IDEF1X ダイアグラムがの中でどのようにしてグラフィカルに示されることになっているか示している。

ある理由のために、この範囲の表現は、トマス・ブルースの本の中では IDEF1X の表現として取り上げられなかった。また、ERwin、IDEF1X をモデル化するツールのベストセラーの中でも Logic Works[10]によって含まれなかった。

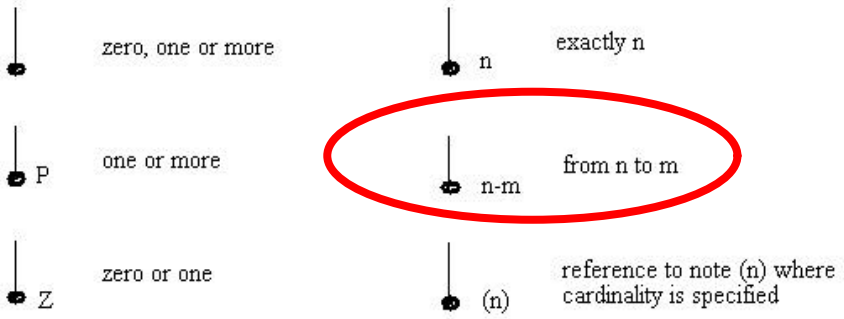


図 9

[10] Logic Works は 1998 年 Platinum Technology に買収され、さらに 1999 年の初めに Computer Associates に買収された。

8．リレーションシップ対アソシエーション

ブルースによれば、IDEF1X のリレーションシップは「親のエンティティの個々の主キー属性が子エンティティの外部キー属性になる 2 つのエンティティの関係」[11]である。

UML 参照マニュアル（UML Reference Manual）によれば、UML のアソシエーションは「インスタンスの接続を含んでいる 2 つ以上のクラシファイヤーの意味的な関係」[12]であると述べている。移行の重要な問題を除いて、リレーションシップおよびアソシエーションは本質的に同じものである。それらはエンティティ/クラスを接続し、カーディナリティ/マルチプリシティの規則を持っている。

8.1 1:1 対 N:M

IDEF1X から UML まで横断的に見た場合、単純な実線のラインのカーディナリティ/マルチプリシティに関する少しの混乱がありる。FIPS 184 では、1 対 1 を実線のラインとしてのリレーションシップ（依存）で表現し、カーディナリティの表現はしない。[13]（図 10 を参照）他方、UML のアソシエーションは多対多の表現をこの実線で表現する。N:M のアソシエーションを参照する実線のラインは UML シリーズで定義されているように正確な表現である。しかしながら、私たちは、すべてのオブジェクト・ツールがその表現に合意しているとは限らないことを知っている。Paradigm Plus は IDEF1X の表現に似ていながら、1 対 1 のオブジェクト関係を意味するために簡素なアソシエーションを持っている。

移行する場合は次ぎのことに注意する。そしてそれを保証するパラダイム間での、簡素な実線のラインのカーディナリティ/マルチプリシティは混乱しない。

図 10

[11] P. 536 Designing Quality Databases with IDEF1X Information Models by T. Bruce 1992, Dorset House

[12] p. 152 The Unified Modeling Language Reference Manual by J. Rumbaugh, 1999 Addison Wesley

[13] 1:1 リレーションシップは、実線で表現されるが、ERwin 3.5 でもサポートされていない。しかし、FIPS 184 で定義されているに有効な IDEF1X の表現である。

8.2 1:1 リレーションシップ

図 11 は、1 対 1 の両方のパラダイムのリレーションシップ関係を表わしたものである。パネルの上部の部分は ERwin の拡張を備えた IDEF1X 形式それを示す。ボックスはエンティティを示します。また、キー・インディケーターは、属性がキーであることを示す。それに相当する UML の 1 対 1 では図の下のほうに示されている。図 11 は UML ガイド[14]に定義されている UML 規則にしたがっている。

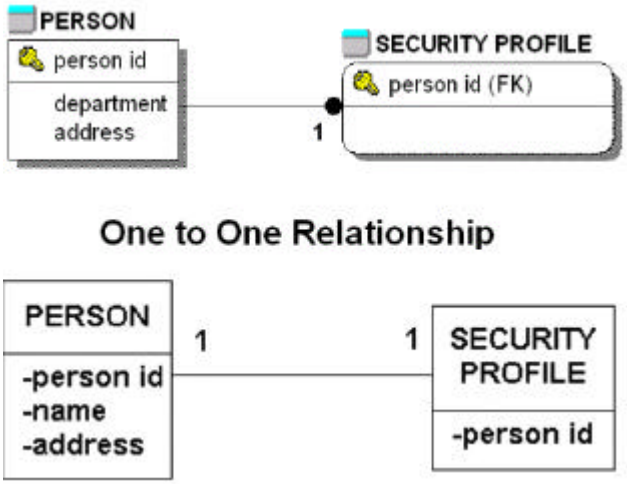


図 11

8.3 単純なリレーションシップ

図 12 から図 15 は、IDEF1X 対 UML の中で複雑でないカーディナリティ/マルチプリシティの相対的な表現を示す。

註：UML が 0、1 またはそれ以上の表現をしていることに注目。これらは、省略、もしくはアスタリスク付き、またはアスタリスクだけで表現されています。（図 16 をその後参照）

[14] P. 347 The Unified Modeling Language Reference Guide by Rumbaugh et. al. 1999 Addison Wesley

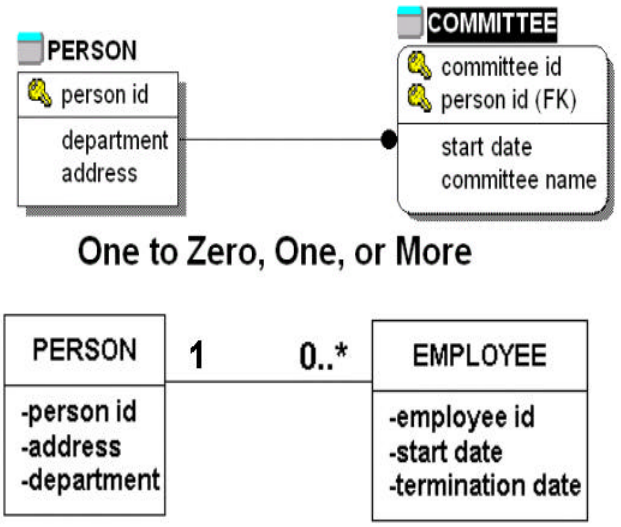


図 12

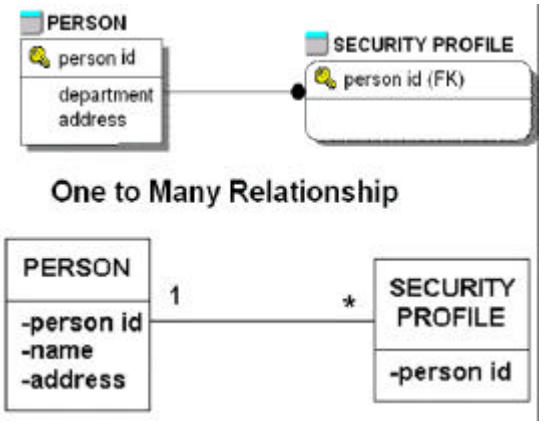


図 13

図 14 **および図** 15 **は、**UML **のより強健なマルチプリシティの記法を読むことが容易であることを示す。**

図 14 **は、**0 or 1 **対** 0 or 1 **の比較を含んでいる。** UML **がリレーションシップのの両側で範囲を明確に示していることがわかる。必要なものはリレーションシップの両側の“**0..1**”である。**

しかしながら、IDEF1X では同じものを表わすのに 3 つの要素を要求する。ダイヤモンド記号は、関係の左側が値を持つことができないこと（NULL）を示す。また、このリレーションシップが非依存であることを示す。

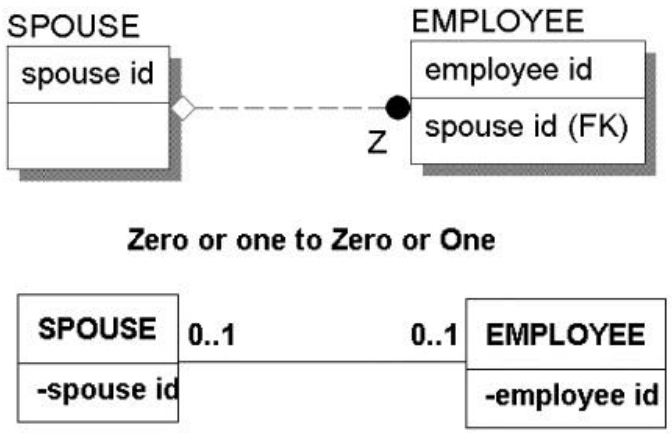


図 14

図 15 は、0 or 1 : 0 or 1 or more を示す。図 16 は、0, 1 or more を表わす 2 つの有効な記法を示す。

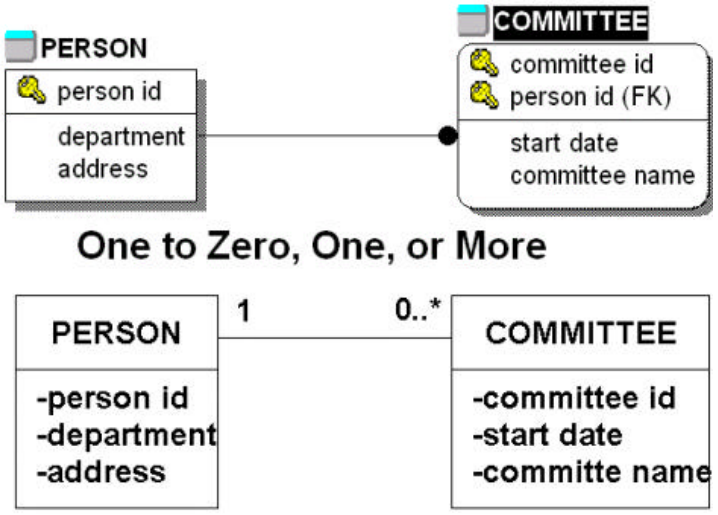


図 15

0..*　＝　*

図 16

図 17 は、UML のアソシエーションの両端でのマルチプリシティの使用が IDEF1X（特にカーディナリティを表わすためにアルファベットが使われている場合）に比べてより、もっと表現力があることを示している。下に示したのは 1 : 1 or more である。

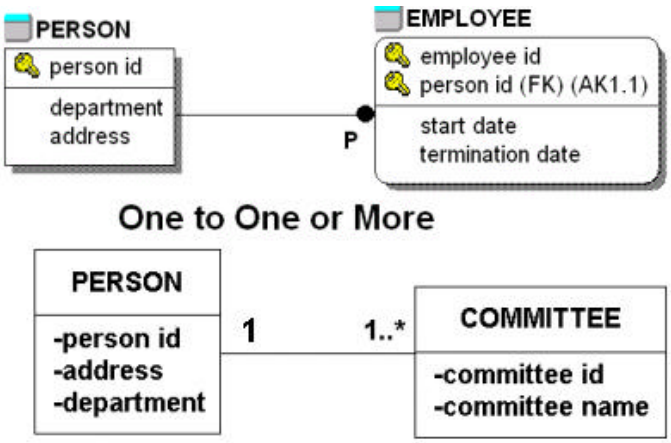


図 17

8.4 簡潔なリレーションシップ

両端のマルチプリシティの値は、IDEF1X の中で表現することができないいより簡潔な関係を示す能力を提供する。図 18 は、このカテゴリーに分類される 2 つの関係を示す。1 or N : 1 or N および N : 1or N の関係。

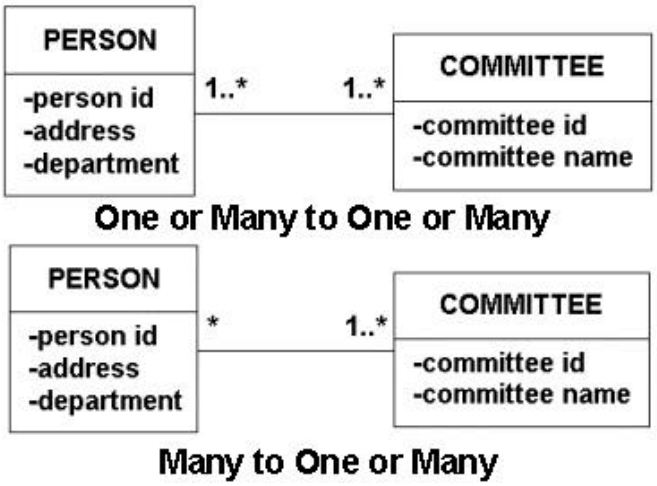


図 18

ここに示されたアソシエーションの左側の明示的なマルチプリシティは、カーディナリティを表現するより総括的な記法の IDEF1X の中で 0 or 1 : N として示すことが可能である。

8.5 N:M リレーションシップ

第 1 のエンティティの各インスタンスが第 2 のエンティティの多くのインスタンスと関係がある場合、N:M の関係は 2 つのエンティティの関係でありその逆も正しいといえる。IDEF1X 記法では、N:M のリレーションシップは、黒丸のドットを備えた実線のラインとして親子の両者につけられる。N:M の関係は 0, 1 or more : 0, 1 or more として説明できる。

図 19 は、グラフィックな表現として、0, 1 or more : 0, 1 or more を IDEF1X と UML で記述したものである。

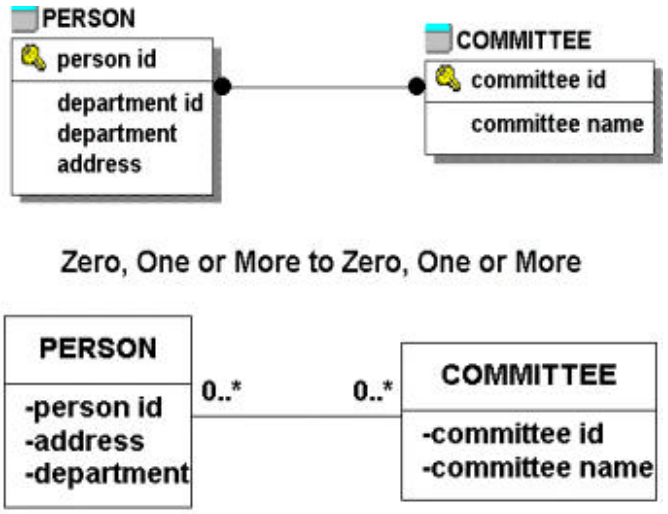


図 19

リレーショナルデータベースは基本的構造としてフラットテーブルである。したがって、リレーションシップは単にバイナリな関係として物理的に説明できる。（2 つのオカレンスの関係）。

N:M の関係は多次元である。したがって、リレーショナル・データベースにそれを構築するためには、それを別のテーブルとして解決しなければならない。

(リレーションシップはそれぞれのリレーションシップの横の多数のオカレンスの間に存在する。)

このテーブルはアソシエーション・テーブルとして知られている。N:M の関係は IDEF1X 論理モデル中では可能である。しかし、論理的な構造としてのみ存在することが可能である。物理データ・ベースにそれをインプリメントするためには、N:M の関係を解決しなければならない。従って、物理モデルの中で N:M の関係は、モデルリングツールでは許されていない。

一方 UML は、物理モデルの中で N:M の関係を許す。オブジェクトリレーショナル・データベースのモデルを作成するために UML を使用する場合、これは問題である。

この状況で、モデラーは、アソシエーション・クラスの作成により N:M の問題を解決することができる。

図 20 は IDEF1X の N:M の解決策であるアソシエーション・エンティティと UML のアソシエーション・クラスの解決策を比較している。（関連付けエンティティとも呼ぶ）

アソシエーション・クラスは有効な UML 記法であるが、いくつかの UML モデルリング・ツールはツール・ボックスの中にアソシエーション・クラスを備えていない。

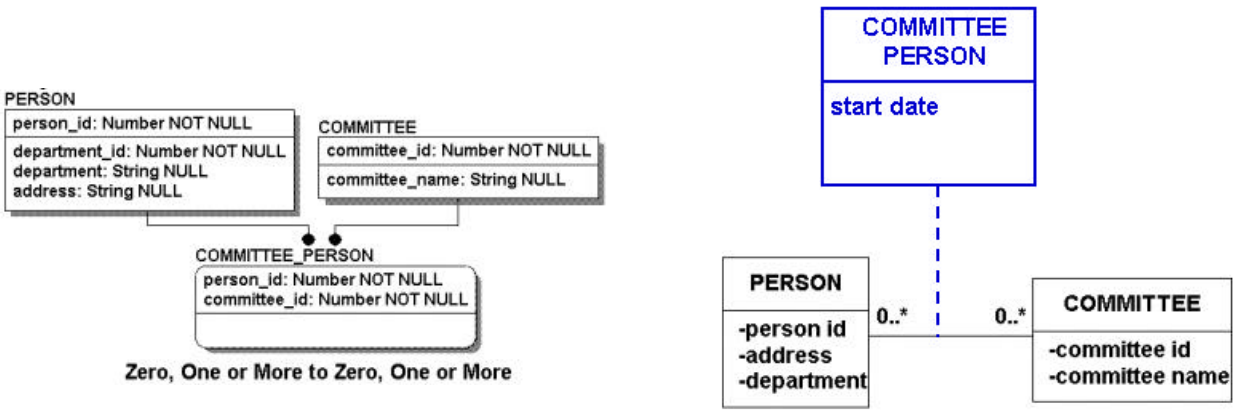


図 20

8.6 複雑なリレーションシップ

数的に複雑な関係がある場合、UML のマルチプリシティの記法のは威力を発揮する。PERSON（人）クラスと COMMITTEE（委員会）クラスの関係を見てみると、この場合次ぎのルールが適用される。

人は任意の委員会のメンバーでなくてもよいが、彼がメンバーである場合には、2 つ以上のメンバーでありえない。
各委員会は少なくとも 3 人のメンバーを持っていなければならないが、20 人以上は持つことができない。
図 21 は、UML で容易にこれらのルールを表わすことが可能であるか示す。

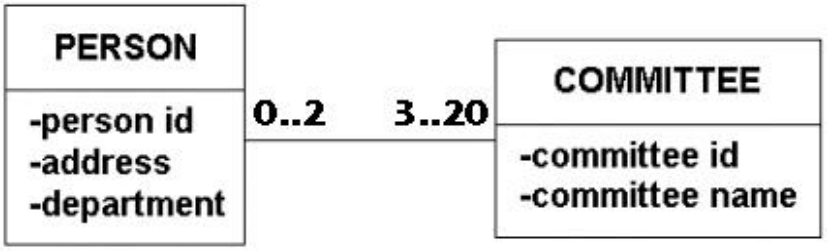


図 21

初期のセクションで引用されたメモを適用すると、“ 0..2 ”は次ぎのことを表わしている。

「PERSON は、0から 2 つの COMMITTEE まで要求する。」“ 0..2 ”の 0 は PERSONが任意の COMMITTEE にある必要がないことを示し、2は、彼が 2 以上の COMMITTEE にいることはできないことを示す。

“ 3..20 ”は COMMITTEE が少なくとも 3 人の PERSON を持っているに違いないが、20人以上を持つことはできないという規則を関連づけている。

IDEF1X 記法は、グラフィカルにこれらの規則を表わす能力はない。ただし、リレーショナル・データベースの中でこれらの規則を実行することができないことを意味していない。

ストアド・プロシジャーはこれらの制約をデータ・ベースによってコントロールされることを保証するために作成することができる。CA-ERwin のような IDEF1X のモデルリングツールでは、テーブルに属するストアド・プロシジャーをコード化することができる。

ダイアグラムのテーブル上でクリックすると、ストアド・プロシジャーはウィンドー・メニューのポップ･アップから見ることが可能である。

しかしながら、それらは、ダイアグラムそれ自身の方法論の記法では表現できない。エンティティの近くに置かれたテキストで規則を述べることはできる

8.7 ダイアモンドと NULL

主キーは NULL 値を含むことができない。したがって、親が子の主キーを継承する場合、IDEF1X のリレーションシップでは、継承されたキーは NULL になりえない。このリレーションシップは義務的な依存関係となる。（図 22 を参照）

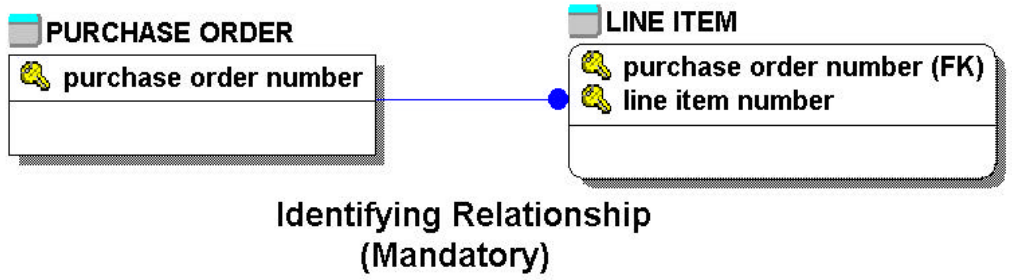


図 22

リレーションシップの中で、親のキーが子の非キーエリアに存在する場合（非依存リレーションシップ）ビジネス･ルールにもよるが、外部キーには NULL 値が許されるというオプションがある。継承された外部キーが NULL でありうる非依存リレーションシップは、オプショナル非依存リレーションシップとして知られている。

IDEF1X は，ダイアモンド記号を使うことによって，非依存関係の外部キーが NULL 値を含むことができるかどうかをグラフィカルに示すことができる。

図 23 は、必須の非依存リレーションシップ。
図 24 は、ダイヤモンド記号を持ったオプショナルな非依存リレーションシップを示す。

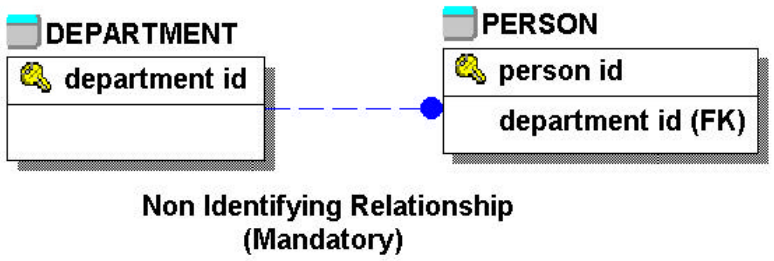


図 23

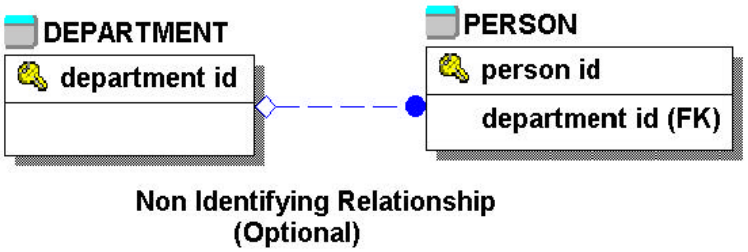


図 24

以前に引用されたように、依存、非依存の概念は OID の使用のために UML には存在しない。UML 関係はすべてオプショナルで、また非依存である。
UML は、次のセクションの中で説明されるような NULL の記法を提供する。

8.8 データ型

IDEF1X の物理的なダイアグラムと同様に UML は、属性データ・タイプのディスプレイを備えている。図 25 は、IDEF1X のデータ型を示す。また、図 26 は、UML のデータ型を示す。

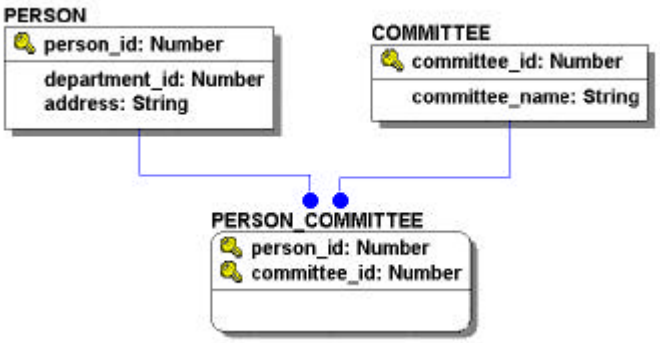


図 25

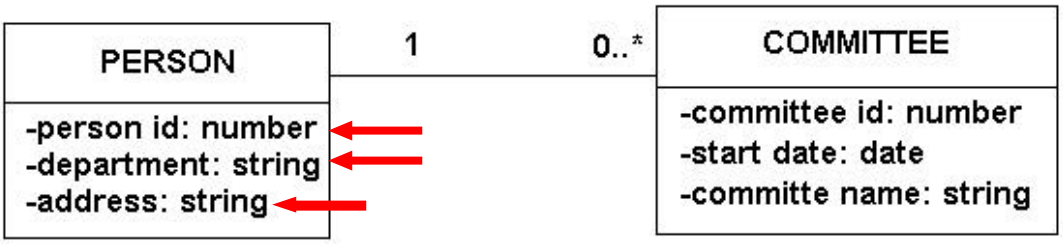


図 26

UML および IDEF1X はダイアグラムに初期値のディスプレイを含んでいる。図 27 は、初期値が IDEF1X にどう表示されるか示。また、図 28 は UML の中で同じことを示す。
(したがって、属性の初期値が NULL の場合、示されるような形で UML にもそれを表示することができる。)

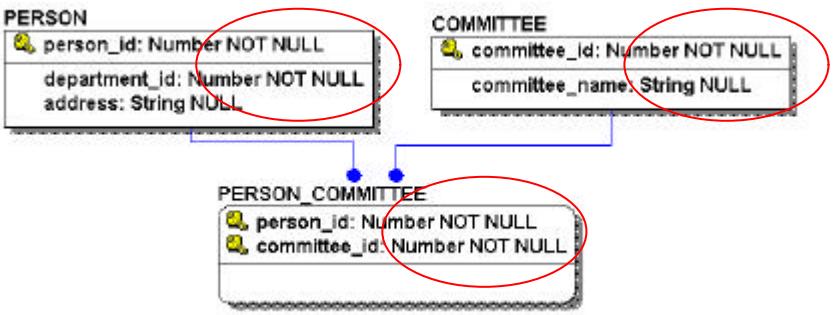


図 27

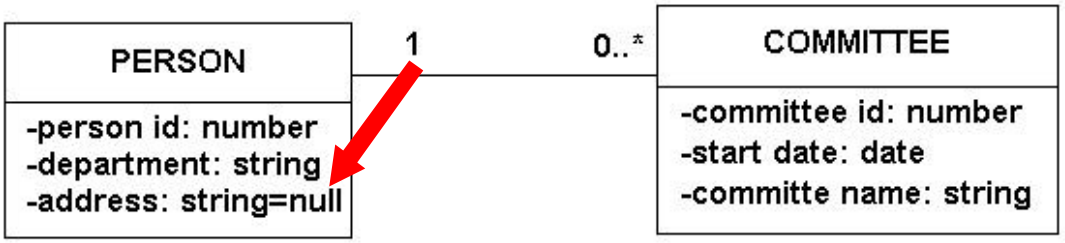


図 28

IDEF1X は若干の冗長性を持っている。オプショナルな非依存関係にある移行されたキー、ダイヤモンド記号および NULL リテラルの両方によって NULL のオプションを示す。(図 29 参照)。それらは両方とも同じものを示す。

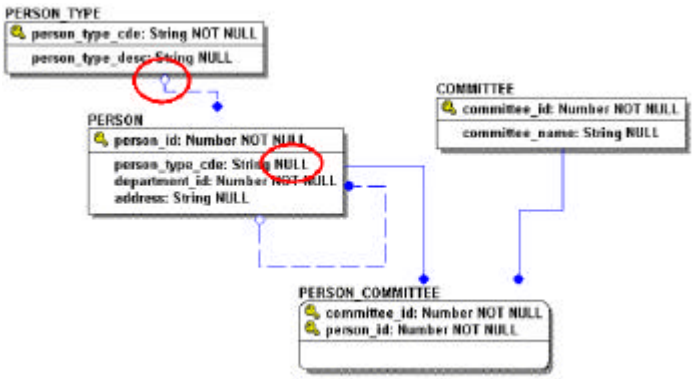


図 29

8.9 N 項リレーションシップ

複数のテーブルにリレーションシップもしくはアソシエーションがあるとき、それを N 項リレーションシップという。

IDEF1X の N 項リレーションシップでは、リレーションシップが、関連付け・テーブルを通して必ず存在するので、上で示すように、リレーションシップは 2 項であり、2 つのエンティティを接続する。リレーションシップは、リレーションシップのすべての交差点の役割をする、関連付け・テーブルとして IDEF1X の中で示される。（図 30 を参照）

他のテーブルの各々は、関連付けテーブルにそのキーを移行する。これは、学生と教授およびコース・テーブルの間で 2 項の「ネットワーク」を作成する。

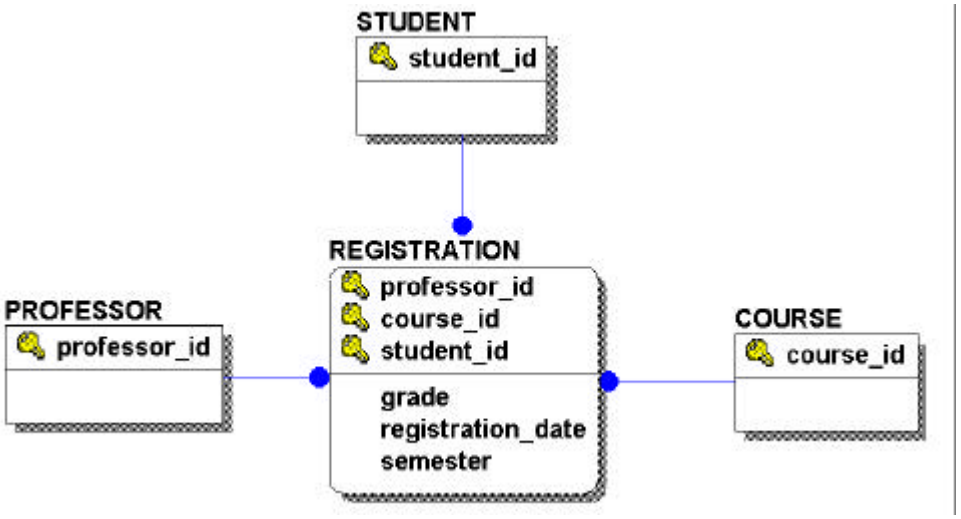


図 30

UML では、N 項アソシエーションは、N 項アソシエーションとして示される。ダイアモンドと各クラスとのこれは大きなダイヤモンドを持ったアソシエーションである。（IDEF1X のダイアモンドは NULL を意味する。）しかし、UML ではアソシエーション・クラスを参照している。
図 31 は、N 項のアソシエーションが UML でどのように表示されるかを示す。

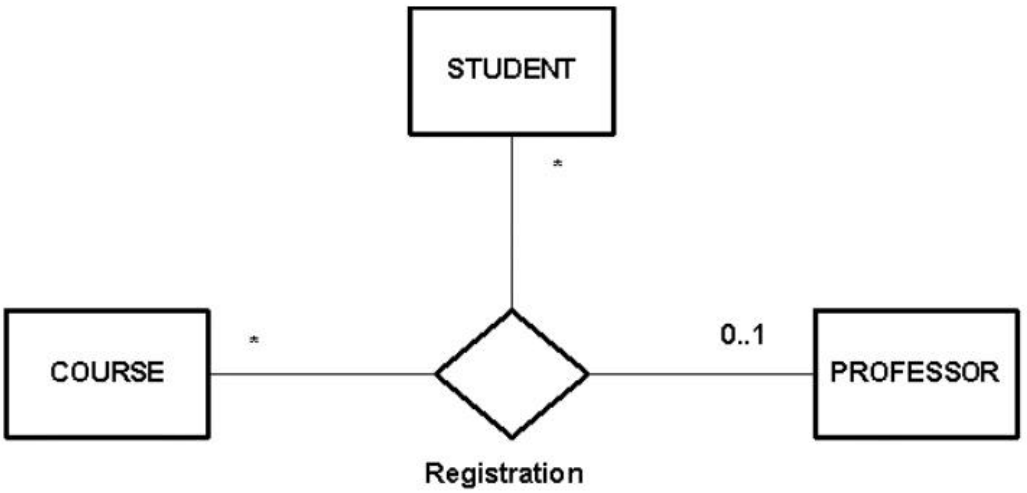


図 31

アソシエーション・クラスのシンボルは点線でN項アソシエーションのダイアモンドに付けることが可能である。（図32参照）このアソシエーション・クラスはULMではオプショナルである。一方，IDEF1Xでは，関連付けリレーションシップでは，物理モデルである。

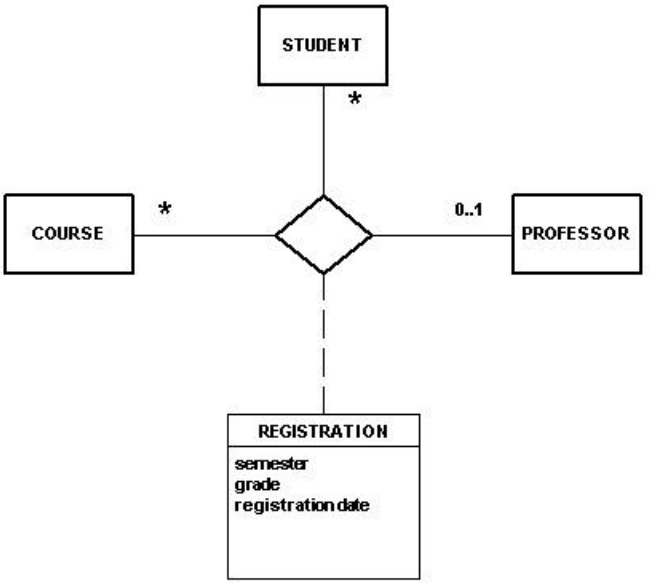


図32

8.10 動詞句

動詞句は、リレーションシップによって定義されたルールについて記述することを支援する。ERwin/IDEF1Xでは，一方または両方からのリレーションシップ記述を提供する。（図33を参照）上の動詞句は親エンティティから子エンティティへのリレーションシップを記述する。下の動詞句は，子から親へのリレーションシップを記述する。[15]

図33

[15] ERwinの初期では，親から子への動詞句しかなかった。ユーザーの要求によって，ERwinでは子から親への動詞句が追加された。

動詞句は UML でも同様の役割を果たす。それは，ERwin/IDEF1X と同じく、アソシエーションの両方向ための動詞句を含める機会を提示する。動詞句に加えて、UML は、物理データ・ベースの制約になるアソシエーションに名前をアサインする能力を提供する。（図 34 参照）

IDEF1X では、フォワード・エンジニアリングのモデルを物理データ・ベースへ移行する場合、親子供関係に割り当てられた動詞句が自動的に関係制約名になる。さらに、UML ではフィルド・アローは方向を示すために使用することが可能である。そしてクラスをオーダーする。これは「ネームド・アロー」と呼ばれ、どの方向を読むべきであるか示しすアソシエーション名である。このフィルド・アローは、純粋に記法上のデバイスとして使用される。

図 34

9．属性

9.1 第 1 正規形対属性のマルチプリシティ

C. J. Date の第 1 正規形の定義「テーブル内のすべての列とカラムでは，1 つの値しかとれない。いくつかの値の集合ではありえない。さらに付け加えれば、リレーションは繰り返すグループ A を含んでいない。この条件を正規化された、あるいは第 1 正規形と呼ばれる」[16].

この規則はリレーショナル理論、したがって IDEF1X の要点の 1 つである。属性のマルチプリシティは IDEF1X 辞書中の受理可能な概念ではない。しかし、オブジェクト・パラダイムの中では! UML は、マルチプリシティが属性のために示されることを可能にする。

図 35 の中で示されるように、アソシエーションに当てはまるマルチプリシティ記法は属性にすべてを適用することが可能です。クラスに含まれている属性の多数のオカレンスありうることを示す属性に隣接している数、一連の数および多くの記法を置くことが可能である。

属性のマルチプリシティおよびアソシエーションのマルチプリシティの間に重要な違いがある。マルチプリシティない場合、どちらかのアソシエーションを示し、マルチプリシティは、多数のデフォルトを取る。属性にマルチプリシティが示されない場合、属性のマルチプリシティは 1 つのデフォルトしか取れない。

[16] P. 93 An Introduction to Database Systems 6th Edition by C.J. Date 1995 Addison Wesley

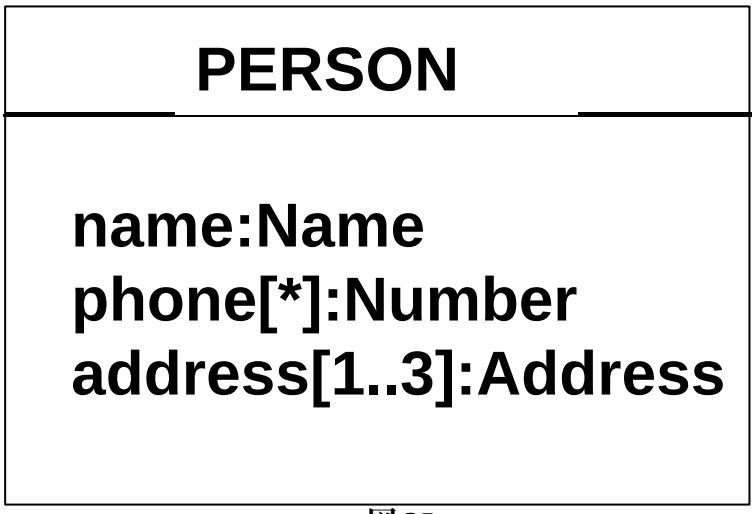


図 35

10. 制約

10.1 偶然性の制約

偶然性の制約に関する、IDEF1X と UML の間の類似点および違いを示す、最良の方法は、1セットのビジネス規則を示し、それらが両方の言語でどうモデルされるだろうかということを示すことである。

例において、PERSON とそれらが貢献することができる COMMITTEE の関係をモデル化する。次のビジネス規則が適用される。

- 各 PERSON（人）は 0 または 1 の COMMITTE（委員会）のメンバーである。
- 各 COMMITTEE（委員会）は少なくとも 1 人の PERSON（人）をメンバーとして持たなければならない。
- 各 COMMITTEE（委員会）は 1 人の PERSON（人）を議長として持たなければならない。
- 各 PERSON（人）は 0 または 1 の COMMITTEE（委員会）の委員長である可能性がある。
- 議長は COMMITTEE（委員会）のメンバーでなくてはならない。

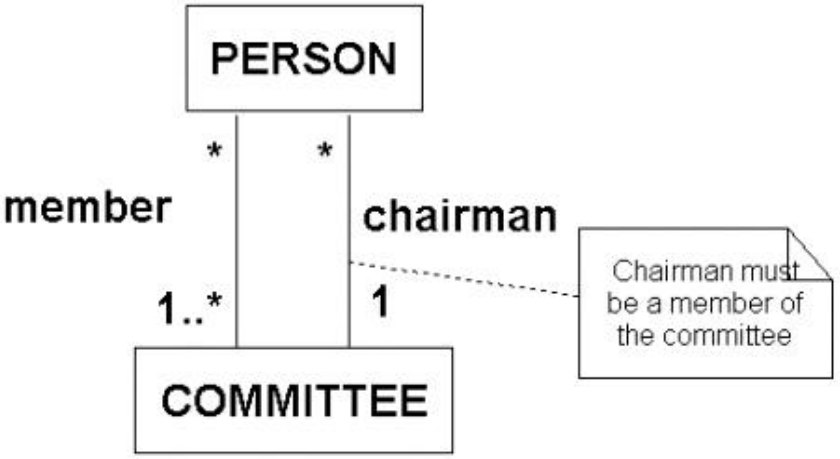


図 36[17]

MEMBER アソシエーションの終わりにある COMMITTEE の “1..*” のマルチプリシティは、1から無限のメンバーが委員会にいることを示す。MEMBER アソシエーションの終わりにある PERSON の ”*” は誰か 1 人の PERSON が無限の COMMITTEE のメンバーであることを示している。

CHAIRMAN アソシエーションは、いかなる PERSON も COMMITTEE（PERSON のアソシエーションの終わりの * を見よ）の任意の数の CHAIRMAN でありうることを示す。また、各 COMMITTEE は CHAIRMAN を持っているに違いないし、単に一人の CHAIRMAN を持つこと可能であることを示している。(CHAIRMAN アソシエーションのについている COMMITTEE の”**1**” に示される。

CHAIRMAN が COMMITTEE のメンバーであるに違いないという規則を示す UML において利用可能な記法はない。そこでここでは、UML のノートを CHAIRMAN アソシエーションに付けた。

UML の中のノートは IDEF1X ダイアグラムのテキストを付けるのと同じ機能を持っている。「その内容がモデルの代替の意味をもたないことを意味しながら、意味的なインパクトを持っていません。」 [18]

技術が進歩するとともに、いくつかのモデルリングツールは、URL または他のものを埋め込む能力を提示し、それらをもっとドラマティックにする。

図 37 は、同じ偶然性の制約が IDEF1X の中でどうモデル化されるかを示す。

[17] The inspiration for this example was from p237 of the Unified Modeling Language Reference Guide by J. Rumbaugh 1999 Addison Wesely.

[18] P. 78 The Unified Modeling Language User Guide by G. Booch 1999 Addison Wesely

これらの 2 つのダイアグラムは、IDEF1X と UML の最も著しい差のうちの 1 つを強調している。制約はすべて両方の図形に表示される。IDEF1X の単純化されたフォーマットは制約を含んでいるす。UML はより多くの強健な記法である。制約をすべてより表情豊かに表示する。しかし、読むことが少し困難である。平易に言えば、IDEF1X 対 UML は単純対複雑、抑制対表現ということになる。

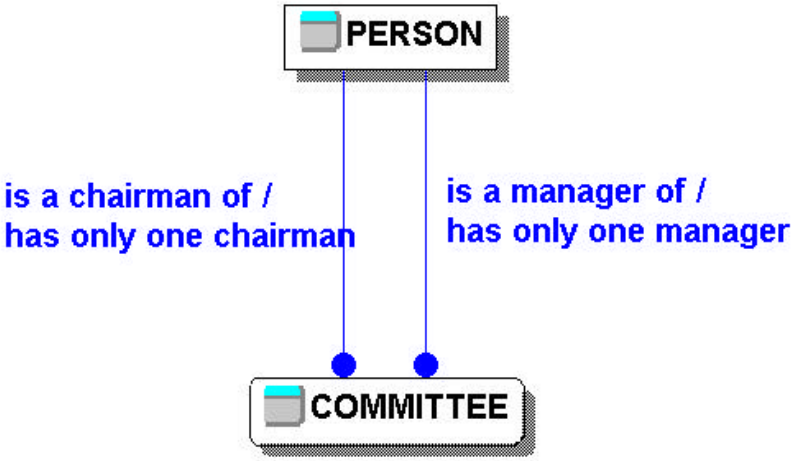


図 37

10.2 OR 制約

Dorsey と Hudicka[19]は、私が IDEF1X で飛ばしたすてきな記法を示す。OR 制約シンボル。

シンボルは、それらのすべてに共通のクラスに関係している 2 つ以上のクラスを横切って置かれたダッシュラインです。

共通のクラスの 1つのインスタンスについて、私はダッシュラインによってストラップされたクラスのどちらかの 1つのインスタンスを単に持つことが可能であることを示す。下の例において、アカウント・クラスのその 1つの実例はメーター・クラスあるいは非メータークラスに関係していることが可能である。しかも、両方でない。この構造に関する問題は、Grady Booch によるこの構造参照を私が見つけなかったということである。

[19] P. 290 Oracle 8 Design Using UML Object Modeling by P. Dorsey & J. Hudicka 1999 Oracle Press

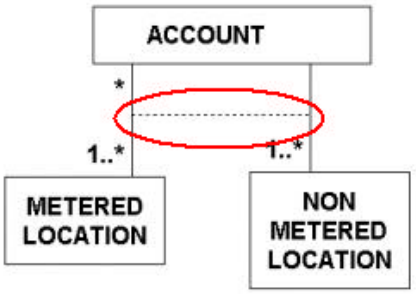


Figure 37A

10.3 カテゴリー対サブタイプ

IDEF1X のカテゴリーは本質的に UML の中のサブタイプと同じ概念である。これらの用語はエンティティまたはクラスの分類または種類を記述するために使用される意味論である。

航空機というエンティティは、ヘリコプター、単発航空機、ジェット機などのカテゴリーを持つことができる。形と呼ばれるクラスは、長方形、円、多角形と命名されるサブタイプを持つことが可能である。

カテゴリーを表わすために作成された構造は汎化階層と呼ばれる。UML および IDEF1X の両方で、関係のある親が汎化と呼ばれる。

汎階層中の子、サブタイプは、親の特性をすべて継承する。IDEF1X/リレーショナルダイアグラムでは、カテゴリーが、親の属性をすべて継承する。UML ダイアグラムでは、サブクラスが、親クラスからの属性およびメソッドをすべて継承する。概念と同じように類似し、したがって、構造類似性がある。

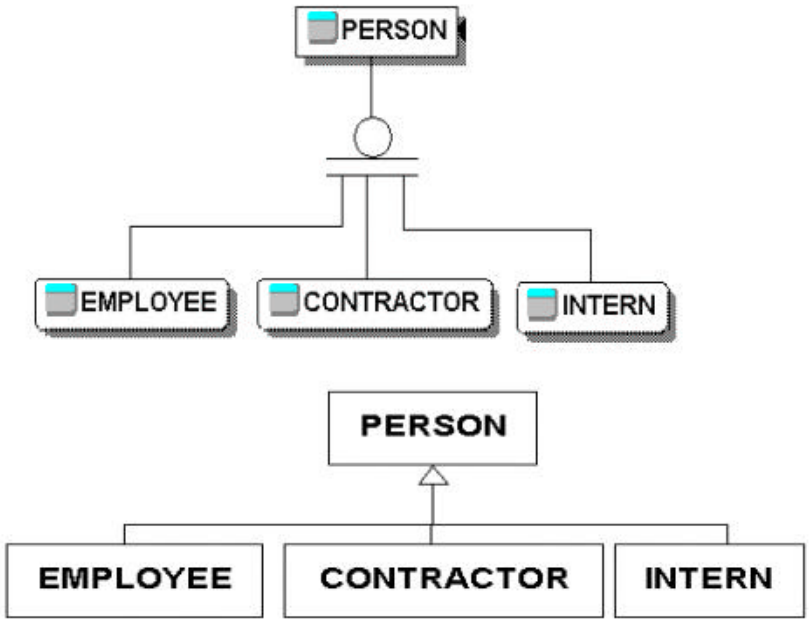


図 38

この例において、両方の構造は、組織、従業員、契約者あるいはインターンのための 3 つのタイプの人を持つことが可能であり、3 つのカテゴリーが相互に排他的なことを示している。これらの構造に追加の飾りがある。カテゴリー識別子(IDEF1X)および識別子(UML)がそれである。この構築によって、カテゴリー間の区別を決定する汎化の属性を表示する。この点では、IDEF1X は UML より少し表現する。正規化されたデータ・モデルでは、識別子属性はコード・テーブルのキーである。IDEF1X は、キーがダイアグラムの不可欠な部品なのでこれを表示する能力を持っている。UML は何回も述べたが、キーに関するオブジェクト・パラダイムの性質のために、識別子属性を備えない。

図 39 は、カテゴリーを備えた PERSON 汎化を示す。属性、PERSON TYPE コード・テーブル、誰のキー、person type code は外部キーとして person table に移行され、カテゴリー識別子のための識別子属性の役割を果たす。

(下に示す、そしてアローに隣接している) 図 41 は、UML 記法の識別子を示す。

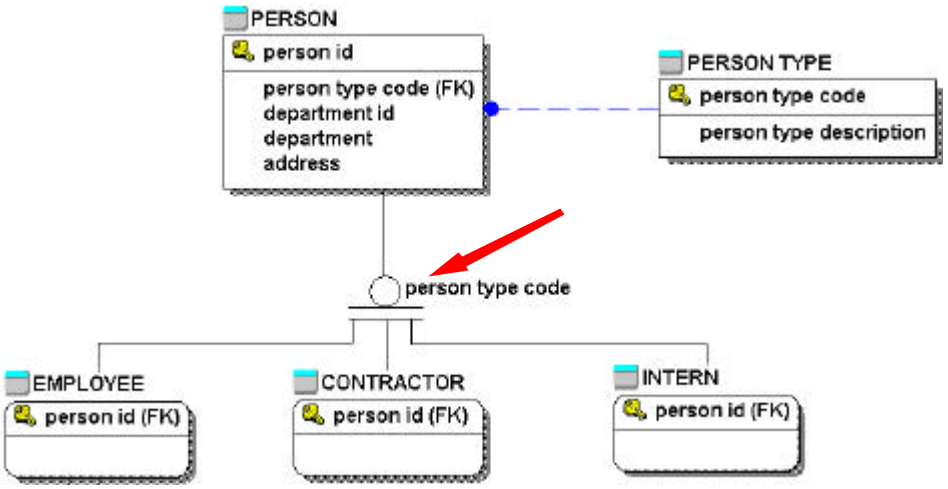


図 39

図 38 と図 39 は、「OR」のカテゴリーを表わす汎化構造である。これは、UML の例および IDEF1X の例の両方において、カテゴリーのインスタンスのためにカテゴリーの 1 つだけが存在することができることを意味する。言いかえれば、人が従業員、契約者あるいはインターンでありうるということであるというビジネスルールを表示している。

図 41 は、IDEF1X および UML の両方の「AND」カテゴリー汎化構造を示す。汎化の 1 つのインスタンス、カテゴリーのコンビネーションがありえることを示している。この例は、ACCOUNT（支払）が CHECK（小切手）、SAVE（貯蓄）、LOAN（ローン）であるというビジネスルールを示す。ここに、再び、IDEF1X と UML は同様にこれらの規則を表示する。次のことを注目する、「OR」カテゴリー構造は汎化にまとめられる、しかし「AND」カテゴリー構造は別々に付けられる。

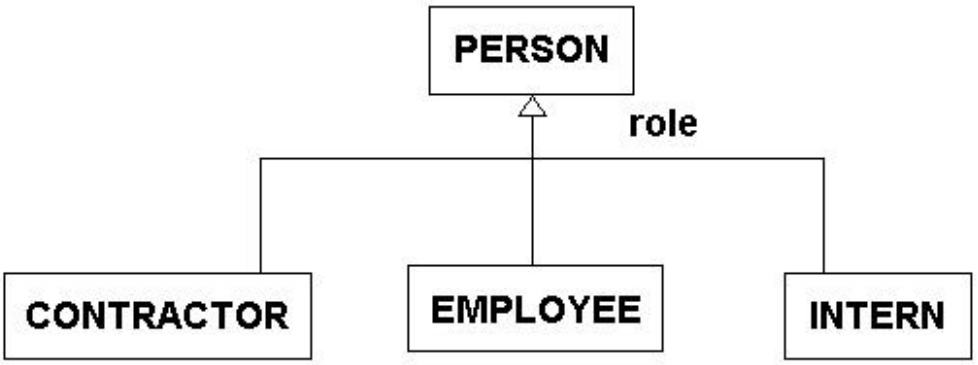


図 40

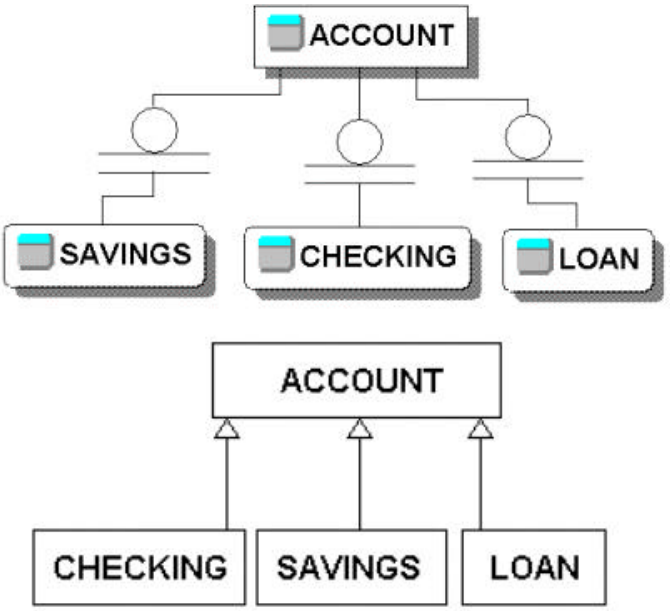


図 41

10.3 継承

継承はオブジェクト世界だけにに属すると皆さんが思っている概念である。継承は、「多くの特定の要素が構造および振る舞い、あるいは多くの一般的な要素を組込むメカニズム」といえる。[20] [21]。

しかし、継承は、リレーショナル世界の中で実際に存在する。そのカテゴリーに当てはまる汎化のすべての属性についての概念は、継承である。

しかしながら、UML では、汎化に表示された継承は拡張した次元を持っている。図 38、39、40 および 41 はすべて、単一の継承の構造を示す。これらのモデル構造では、子供がすべて、一つの親からの特性および振る舞いを継承する。UML は、マルチの継承を表示する記法を提供する。これは、子供が 1 つ以上の親からの特性および振る舞いを継承することを意味する。マルチの継承のための UML 記法と例は図 41 である。

しかし、IDEF1X はカテゴリー階層中のマルチの継承を考慮に入れない。概念的には、3 番めのエンティティとの関係の識別によって接続している 2 つのエンティティを使用しながら示すことが可能である。(関連付けテーブル)。結果は同じである。実際に関連付けテーブル識別する 2 つのテーブルの特性を継承する。図 27 を参照されたい。

[20] p. 462 The Unified Modeling Language User Guide by G. Booch et al 1999 Addison Wesely

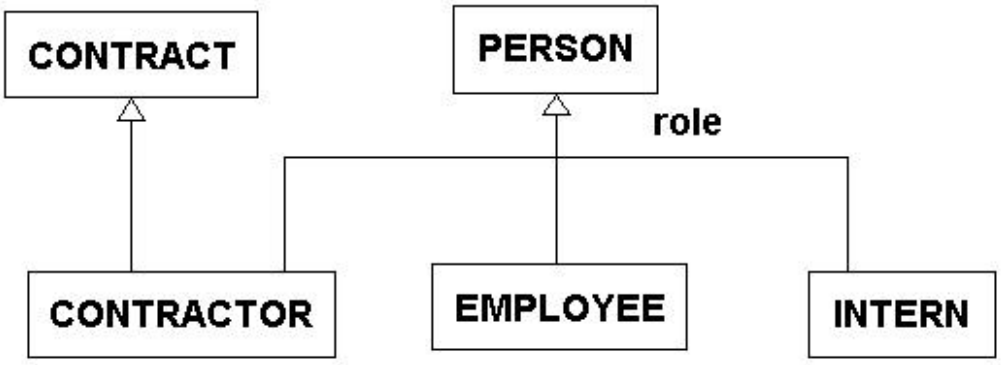


図 42

10.4 マルチおよび複合汎化

マルチおよび複合の汎化構造は「AND」および「OR」のカテゴリー構造のコンビネーションを使用しながらがら IDEF1X の中で表わすことが可能である。類似した方法で、マルチおよび複合の汎化は UML の中で表わすことが可能である。図 43 は、IDEF1X の中のマルチの汎化の例を示す。また、図 44 は、UML の中の多数のマルチ汎化を備えたクラスを示す。

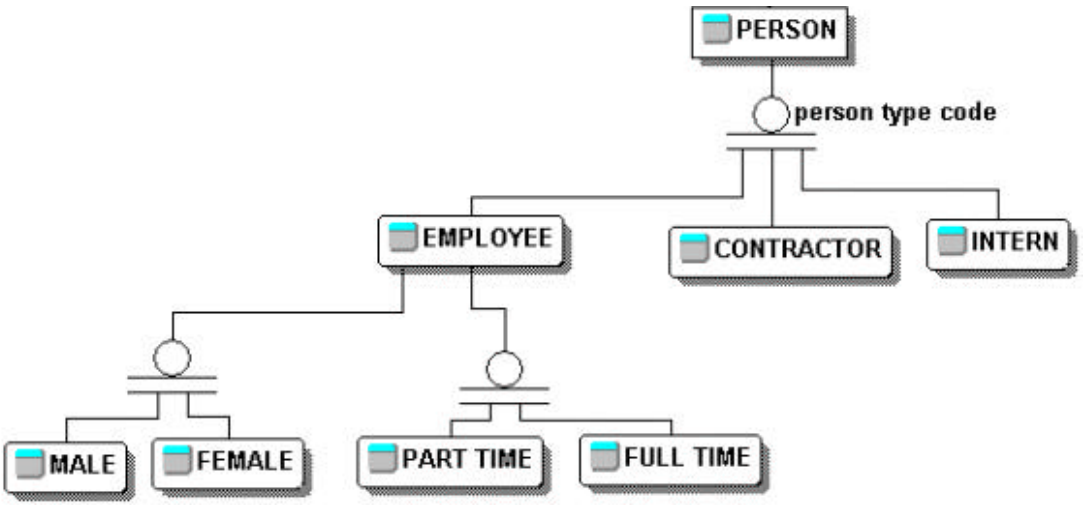


図 43

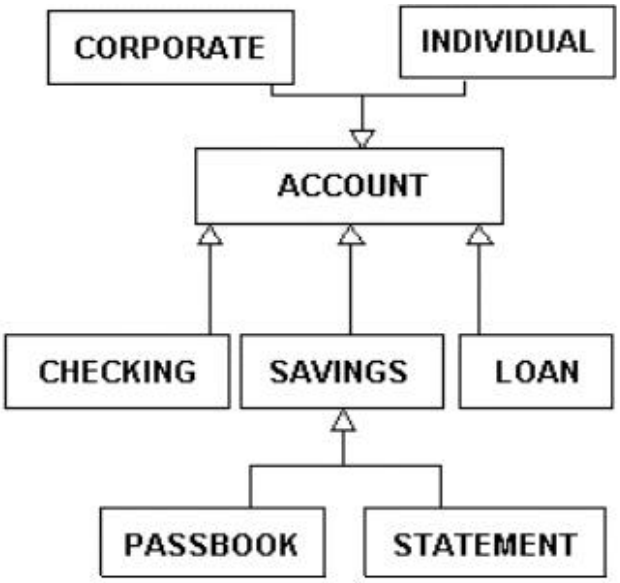


図 44

UML および IDEF1X の両方は、カテゴリー構造中のエンティティ/クラスとカテゴリー構造に外部エンティティ/クラスのリレーションシップ/アソシエーションを許す。これらのリレーションシップ/アソシエーションは「外部エクスターナル」のエンティティ/クラスと両方のスーパータイプあるいはサブタイプの間にありえる。
図 39 は、IDEF1X の例を示す。また、図 42 は、UML の例を示す。

UML はさらに高い複雑な汎化を取ることが可能である。UML は、汎化が適用されるカテゴリー（サブタイプ）間のビジネス規則を許可する。図 45 は、普通預金口座カテゴリーとアカウント・クラスの貸付勘定カテゴリーのリレーションシップを備えた図 45 の拡大状態を示す。[22].
リレーションシップは次ぎのルールを定義する。すべてのローンについて、ローンには付随的な少なくとも 1 つ以上の普通預金口座がある。これらのアソシエーションは、再帰型リレーションシップと同等のものである。

[22] The are refer to as “specialization object classes by Dorsey & Hudicka p. 350

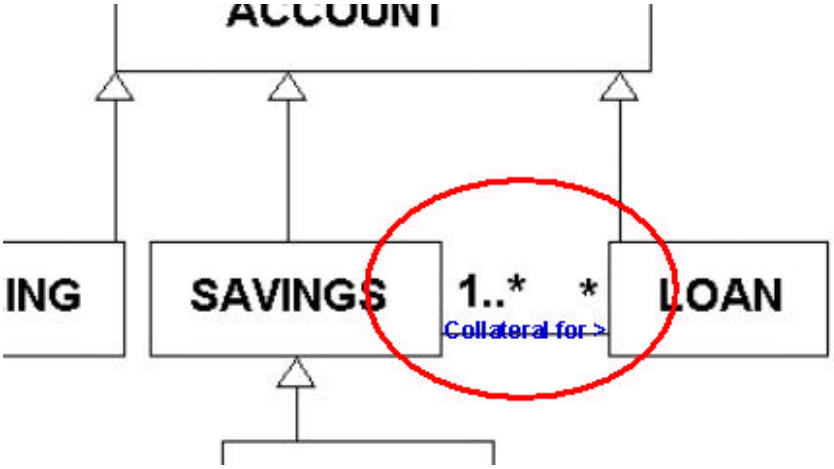
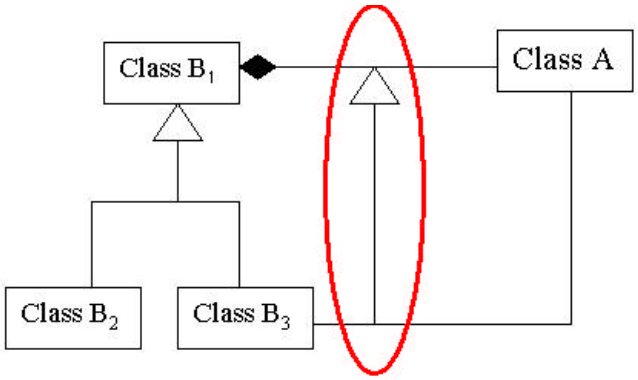


図 45

これらの複合リレーションシップは、「AND」「OR」両方のカテゴリー構造内に存在することが可能である。

これが複雑だと思うと、より悪くなりえる。ちょうど UML が汎化内のアソシエーションに備えるように、アソシエーション間の汎化の生成に備えます。Rumbaugh はこれをコンポジション・アソシエーションの汎化と呼ぶ。 汎化の詳細に関するさらなる議論については、「UML Reference Manual」を参照。[23].



[23] p 232 The Unified Modeling Language Reference Manual by J. Rumbaugh 1999 Addison Wesely

10.5 アグリゲーション（Aggregation）

アグリゲーションは「全体と構成する一部分の全体の一部分の関係を指定するアソシエーションの形式」である。[24]

より簡潔な定義をすると、「もし、クラスA のオブジェクトがクラスB のオブジェクトの集合であるなら、クラス A はクラスB のアグリゲーションであるといえる。クラスB のオブジェクトがクラスA のオブジェクトにアタッチする必要はない。マスターは詳細から作られる。しかし、詳細はマスターのコンテキスト外である。」[25]

Rumbaugh は、評価できる次の例を示している。いくつかのもの PC の上のパスはセグメントのコンビネーションである。それらのセグメントの各々は現実に、それ自身、存在することが可能である。パスはしたがって集合であると考えられる。

図 46 の中で示される例において、私たちは、チームがプレーヤーから構成されることを知る。チームにかかわらずプレーヤーについて話すことは適切である。アグリゲーションのシンボルは赤い円で強調されている。

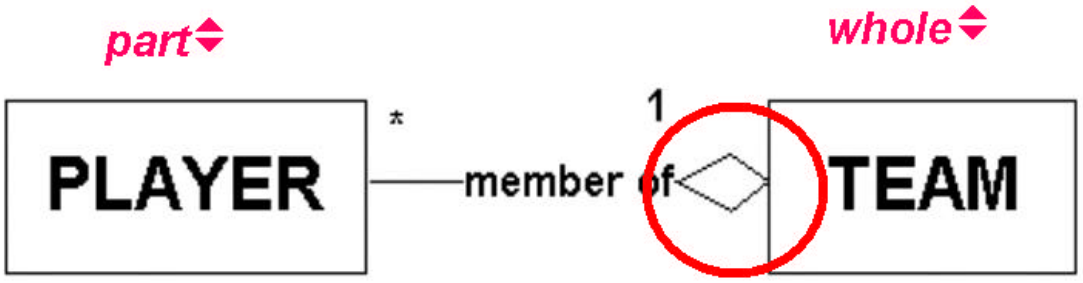


図 46

アグリゲーションの概念は、単純性を保つことによって、記号表現としてリレーショナル・モデルで開発されなかったリレーションシップの形式のうちの 1つである。アグリゲーションは、IDEF1X の中では、1 から多の関係として表現できる

UML では、アグリゲーションを表わすために、アグリゲーション・アソシエーションの横に黒のダイヤモンド記号を（図 46 参照）を使用する。

Booch は、アグリゲーションが意味論の中で練習であると述べている。「単純なアグリゲーションは完全に概念であり、部分から全体を識別する以外のものではない。単純なアグリゲーションは、

[24] P. 458 The Unified Modeling Language User Guide by G. Booch 1999 Addison Wesley

[25] p 233 Oracle 8 Design Using UML Object Modeling by Dorsey & Hudicka, 1999 Oracle Press

全体とその部分のアソシエーションを横断してのナビゲーションの意味を変更しない。同様に、全体およびその部分のライフサイクルをリンクしない。」

アグリゲーションを表わす IDEF1X シンボルはないが、概念的には参照整合ルールの形としてリレーショナル・モデラーによく知られている。

リレーションシップの親または子に削除を波及させないルールを置いたときがアグリゲーションである。ERwin の物理モデルダイアグラム(IDEF1X あるいは IE)では、スイッチが入れられた参照整合を表示するオプションをダイアグラムで表現することが可能である。ルールはは、2 つのルール要素およびコロンで各々の第 1 の文字を使用しながら表わされる。

D:R = Delete Restrict
U:R = Update Restrict
D:C = Delete Cascade

例として、図 49 を参照されたい。削除の波及（CASCADE）は表示されてない。

図 47

10.6 コンポジション（Composition）

コンポジションはアソシエーションのタイトな形式である。Dorsey および Hudicka は、「詳細オブジェクトはマスターが所有している。」としている。コンポジションでは、「マスターのコンテクストからの詳細について議論することは意味がない。マスターが削除されれば、詳細はすべて論理上削除されなければならない。」[26]

Rumbaugh は、コンポジションについてより形式的な説明をしている。「全体による部分のストリング所有権および一致するライフサイクルからのアグリゲーション・アソシエーションの形式である。部分は単に 1 つのコンポジットに属するかもしれない。固定でないマルチプリシティを備えた部分はコンポジット自身の後で作成されるかもしれない。しかし以前に作成されたそれらは、そ

[26] ibid 236

れらと伴に消滅する（すなわち、それらはライフサイクルを共有する）。さらにそのような部分はコンポジットの消滅の前に明示的に削除することが可能である。」[27]

発注はコンポジションの最も明白な例である。発注は、明細行に先だって作ることができ、一方明細行は発注なしに作ることはできない。明細行は発注のライフサイクル中で挿入、削除することができる。しかし、発注が除去される場合、明細行はそれによって除去されるに違いない。コンポジションのための UML 記法は黒のダイヤモンドである。図 48 は、UML で発注およびその明細行の表現を示す。

アグリゲーションのように、アグリゲーションを表わすための IDEF1X シンボルはないが、概念的には参照整合の形としてリレーショナルモデラーによく知られている。 コンポジションは、DELETE CASCADE あるいは UPDATE CASCADE 規則は親子関係の中で適用される。Codd によれば、「これらの規則はともに必要な存在依存性を捕らえて反映する。」ということである。[28]

Codd は、多対 1 のリレーションシップの中でこれらを「弱いエンティティ」と呼ぶ。ERwin 物理モデルダイアグラム（IDEF1X あるいは IE）では、これは、スイッチが入れられた参照整合規則を表示するオプションとしてダイアグラムで表現することが可能である。CASCADE DELETE のための記法（D:C）。(図 49 を参照)。コンポジションの記法は赤い円で強調されている。

図 48

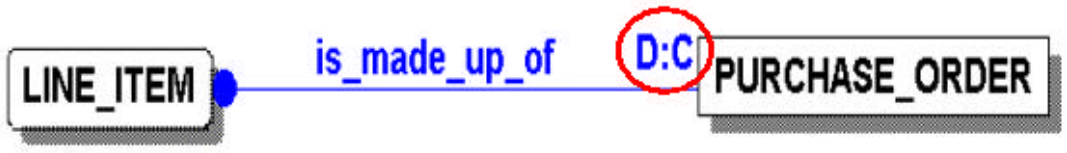


図 49

[27] - p. 226 The Unified Modeling Language Reference Manual – Rumbaugh, 1999 Addison Wesley

[28] p. 360 An Introduction to Database Systems 6th Edition by C.J. Codd 1995 Addison Wesley

オブジェクト・リレーショナルダイアグラムでコンポジションは、入れ子のテーブルあるいは「varrays」で表わすことができる。（それら両方はこの議論の範囲外である。）

10.7 依存性

依存性は、この議論にとって重要な意義を持つ別のタイプの UML のリレーションシップ関係である。（他の 2 つ、汎化およびアソシエーションは既に議論された。）

「依存性は、1 つのものの明細の変化がそれを使用する別のものが達成するかもしれないと述べるリレーションシップの使用法である。しかし必ずしも逆は言えない。あるいは、1 つの要素(サプライヤー)への変更が他方の要素(クライアント)によって必要とされる情報に影響するか供給するかもしれない 2 つの要素の関係」[29]

依存性のための記法は少し混乱させられる。それは依存クラスまたはインターフェース[30]の終了を指すアロー備えた点線である。変更が「FROM」から発することで、アローはクラスの方向を示す。図 50 は、依存性の例を示す。モディフィケーションはコンポーネントに作用することが可能であるオペレーションのコレクションである。

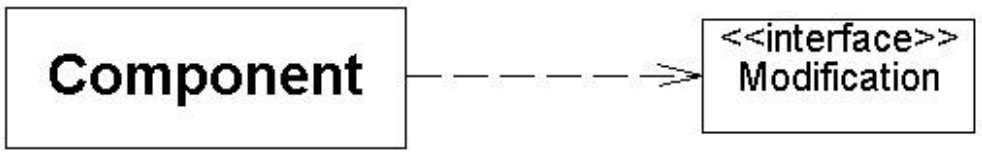


図 50

明白に、オペレーションは IDEF1X ダイアグラムではドキュメント化されていない。ここで、IDEF1X にはない概念で構築する。オペレーションの集合は、リレーショナル・テーブルに関連したトリガおよびストアド・プロシジャーに多少類似している。[31]

CA-ERwin のようなモデルリング・ツールはトリガとストアド・プロシジャーの作成およびドキュメント化するために機能性を提供している。

[29] p. 250 The Unified Modeling Langugage Reference Manual by J. Rumbaugh 1999 Addison Wesely

[30] An interface is a UML construct which represents a collections of operations which can be performed by a class. In

[31] This can be compared to a *package* in Oracle which is a group of related PL/SQL procedures and functions

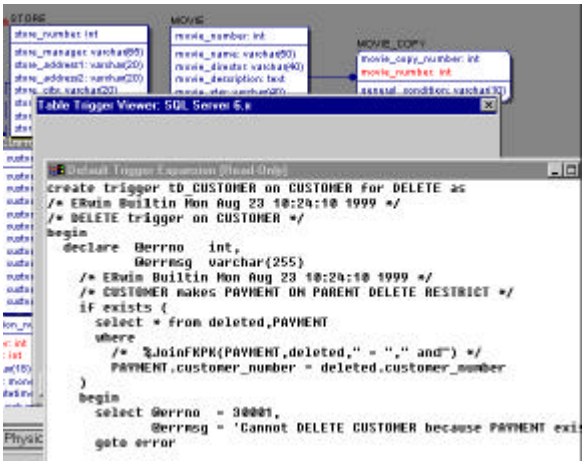


図 51

これは依存性への非常に浅い比較であり、参照のラフなフレームとして単に意味される。UML **では、依存性が** 1 **つのものが別のものを使用するか、分類された** 11 **の機能で意味論的に定義することが可能であるかを示すために使用される。一層の議論はこの記事の範囲外である。**

11. 再帰型リレーションシップ

リレーションシップがオブジェクトもしくはエンティティのそれ自体に存在する場合、再帰的なリレーションシップは周期的な構造であるといえる。このリレーションシップ・タイプは有用であり強力である。親がその子供でありうる場合、リレーショナル・モデリングで頻繁に使用される。

しかしながら、この記事の中で頻繁に引用され、オブジェクトをモデル化するツールとパラダイムの中で全く見落とされた、決定的な UML **テキストの中でほとんど無視される。実際、再帰という用語は、オブジェクト・パラダイムで使用される用語であるようには思えない。再帰的なアソシエーションは、**2**つのインターフェース・スペシファイヤーの任意のアソシエーションと単に見なされる。**[32]

(インターフェース・スペシファイヤーとは、「アソシエーション・クラスにアソシエーションの意図を満たすのに必要な振る舞いの明細である。要求された振る舞いを指定するインターフェース、クラスあるいは他のクラシファイヤーへの参照から成る。」リレーショナル化の中で言えば、単に動詞句である。)

[32] p 146 The Unified Modeling Language User Guide by G. Booch 1999 Addison Wesley

図 52 および 53 は、組織の管理階層を描写する、再帰的な構造を示す。この構造は従業員をモデル化している。従業員の他の適切な属性と共に、テーブル中の各従業員のオカレンスは、マネージャーに関するデータを含んでいるのと同じテーブル中のオカレンスを指している。実際、再帰的な 1 つのリレーションシップで、企業の階層全体を示すことができる。図 52 は、IDEF1X の再帰的なリレーションシップを示す。図 53 は、UML の再帰的なアソシエーションを示す。

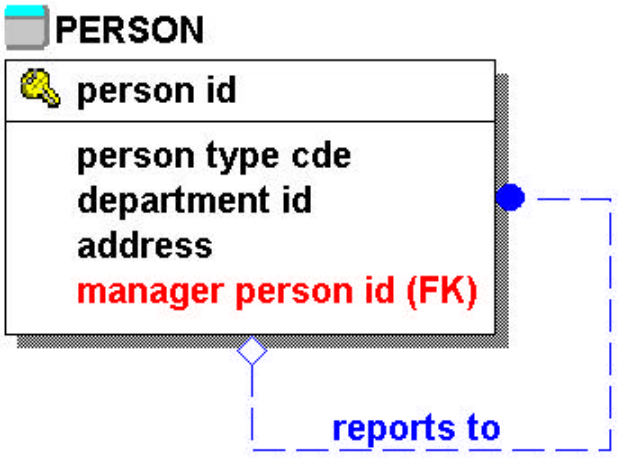


図 52

Dorsey と Hudicka のドキュメント 9 シチュエーションは、再帰的なリレーションシップ/アソシエーションが適切にビジネス・ルールをモデル化することを示した。それぞれのシチュエーションは、IDEF1X および UML 両方の中にほとんど等しく示すものでありえる。[33] それは両方のパラダイムの中において強力なモデルリングツールである。

[33] p260 Oracle 8 Design Using UML Object Modeling by P. Dorsey & J.R. Hudicka 1999 Oracle Press Dorsey & Hudicka a ddress UML in relation to Oracle's data modeling methodology. However, the concepts can be easily understood in terms of IDEF1X.

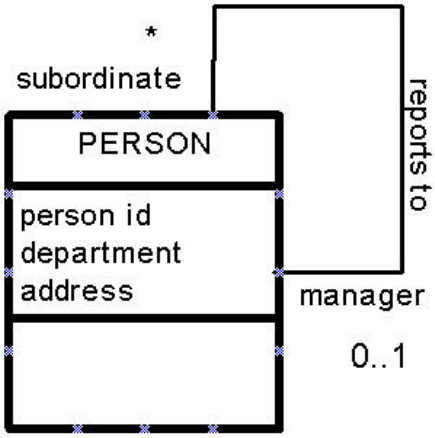


図 53

12. メソッド

リレーショナル/IDEF1X パラダイムとオブジェクト/UML パラダイムの大きな違いの 1 つ、あるいは最も劇的な違いは、モデルの中へのメソッドもしくはオペレーションの導入である。

ER モデリングのうちの 1 つの観点から、私たちが繰り返し自問自答していたのは、「データはプロセスではなく、プロセスはデータではない。」「プロセスによりモデルに影響を及ぼさない」ということであった。

私がアプリケーション・プログラマとモデルをレビューし、正規化された ER ダイアグラムを見る場合、プロセスを考慮しないことを心しなければならなかったことが、何回もあった。

メソッドが加わることにより、埋め込まれたテーブルと配列によって、より容易な程度までこれは変わりました。

メソッドは RDBMS パラダイムに存在せず、従って、IDEF1X ダイアグラムには表現方法がない。

前に述べたように、トリガとストアド・プロシジャーで、これらは比較可能である。図 54は、オブジェクト・クラスがデータに作用するデータ属性およびプロセスの両方をどのように含むかを示したものである。

13. IDEF1X 対IDEF4X 対DMT2

この議論の後で、あなた方は疑問に思うかもしれない;

IDEF1X と UML の間に非常に多くの類似があるのに、わざわざ新しい言語を学習するか。なぜ、オブジェクトを組込むことが可能な IDEF1X の拡張の作成をしないのか?

IDEF1X を拡張し、かつオブジェクト概念を含んだ試みがかつてあった。オブジェクト・バージョンの IDEF1X は「Designing Quality Databases with IDEF1X Information Models」[34]の中で DMT2 と呼ばれ、そのためのセクションが設けられた。

その本の以外に、私は何も聞いていないし、何も見ていない。私は最近ポピュラーなデータモデリングおよび ERwin リスト・サーバーにそれに関してこの質問をのメッセージをなげかけた。しかし、何百もの加入者のうちの誰一人からも応えを受け取らなかった。

私は、UML の出現が何かを窒息させ、DMT2 にスポットライトを当てるのを阻止しようとしているではないかと疑問に思っている。

Knowledge Based Systems, Inc.（http://www.idef.com）は、「次世代 IDEF メソッド」と呼ばれるものを開発している。それらは IDEF3 プロセス・フロー（IDEF3 Process Flow）およびオ Object State Description Capture Method、IDEF4（オブジェクト指向の設計方法）および IDEF5（Ontology Description Capture Method）を含んでいる。

彼らは、これらの方法論に伴う自動化ツールをに開発した。しかし、それは彼らが単に最小限の興味で作ったようにも見える。

[34] p. 276 Designing Quality Databases with IDEF1X Information Models by T. Bruce 1992 Dorset House

「130 文字以上のマクロについて」

Gary Gramm

カラムの名称を巧みに扱うためにドメインエディタにてマクロを使用したとき（すなわち、基準に則ってデータのネーミングを行うことであるが)、文字数はすぐに長くなってしまう。要するに、カラム名称は作成したマクロのコードなのである。

本来の ERwin では、これは問題とはならない。ModelMart に同じモデルを移行した際に、このマクロの長さが 130 文字を超える場合、仮想的には無限長を設定できる ERwin と違って、ModelMart ではカラム名称は 130 文字までしかサポートしていないため、130 文字で切り縮めてしまい、マクロは無効になってしまう。
少なくとも２つの方法でこれを回避できる。

1. 外部テキストファイルにマクロを記述し、ドメインエディタでのカラム名称のスペースにそれを参照させることである。（しかし、これはあまりお勧めしない。あまりに遅く、移動させられると問題になりやすい。）
2. アタッチされていないトリガーのテンプレートにそのマクロを記述し、ドメインエディタでのトリガーテンプレート名称のスペースにそれを参照させることである。

これが私の経験上、お勧めする方法です。

しかし、DB2 のようなトリガーをサポートしていない DBMS にプログラムがあった場合、ERwin はサーバーメニューのトリガーの要素をグレーアウトしてしまうので、うまい方法を見つけなければならない。

これらのケースでは、対象 DBMS を SQL Server のようなトリガーをサポートする DBMS に一旦変更し、トリガーコードを作成し、それから、元の DBMS に設定を戻す。もっとも変更前と後でいくつかのデータがコンバートされてしまう問題が起こるかもしれないが。

私は多くの場合、データ型変換のダイアログボックスには「いいえ」と答え、ドメインデータ型の変換もチェックしない。

一旦、元の状態に戻ると、再びトリガーのコードを見ることはできなくなるが、消えてなくなるわけではなく、存在している。

このケースは、実際 ERwin のバグと考えていたことが結局一つの特徴となることがわかる。

この TIPS の最後に、あなたが行ったことをドキュメントとして残すことを忘れずに!!

どうやって名前がジェネレートされているかを今から 3 ヵ月後に誰かから質問されたとき、マクロとトリガーまでは思い出せるかもしれないが、そのトリガーが不可視の状態か

らの戻し方は忘れてしまう可能性が大きい。

訳：㈱ウィッテイシステム　大竹

Mass Import of Definitions

ListMistress, ERwin Users Discussion Group
InfoAdvisors, Inc.
www.infoadvisors.com
Toronto, Ontario Canada

（訳者註）エンティティや属性の定義はデータモデルを理解する上で、ネーミングと共に非常に重要である。しかし、一旦 ERD を作成した後、または ERD とは別タイミングであるいは別な設計者によって行われることがある。ここでは、別々に定義された属性定義情報などを ERwin 上のモデルとして統合するやり方について紹介している。

最初に背景を少し述べる。エンティティやアトリビュートの定義情報を持っているデータモデルはそう多くない。担当者のマシンに ERwin がインストールされ、トレーニングが実施され、そしてモデルが数日の間にできてくる間に、モデルとは別に定義をおこない最後にそれらをマージしなければならなくなる。

-[ERwin での操作] エンティティの名前、定義、テーブル名、属性名、カラム名、属性定義のレポートを生成する。
＜重要＞
エンティティに所有された属性だけであれば、各々の属性は１回だけしか現れない。（もちろん、マニュアルで重複した属性を定義していなければですが）また、全てのオブジェクトは”論理のみ”をマークしておかないようにしなければならない。

-[ERwin での操作] この ERwin で生成したレポートを Excel スプレッドシートにエクスポートする。DDE の処理を軽減したいなら、ASCII ファイルインタフェースでエクスポートする。

- このスプレッドシートを定義のための作業用として担当者に渡す。何があってもオブジェクト名を変えないように指導することが必要である。オブジェクト名に引用符や特殊文字、アポストロフィーなどを使用しないように指導する。これはとっても難しいことであるが、これを守らないとインポートできない。
- （訳者注：日本語環境でもネーミングルールが制定され、用語が定義されて入れるのでほとんど問題にならない）

- 定義内容をレビューし、あらゆる問題について合意を得る。引用符や特殊文字、アポストロフィーを見つけたら変更または削除する。

- オリジナルモデルでデータベーススキーマを生成し、保存する。この時、Create Table,Column オプションのみを指定する。

- 最終のスプレッドシートファイルに、リバースエンジニアリングのための SQL 文を追加しする。

- テーブル名とカラム名に対して、連結機能を使用して SQL を生成する。

例えば、excel では次のようになる（UDB の場合）
CONCATENATE ("COMMENT ON COLUMN " ,B1,"."C1," IS ",D1, "';")

ここに、B1 はテーブル名（excel）列、C1 はカラム名（excel)列、そして D1 はカラムの定義内容（excel)列。この式をスプレッドシートの新しい列にコピーする。そして、この列を新しいスプレッドシートに文字列としてコピーする。

- 新しいスプレッドシートを ASCII フォーマットにエクスポートする。

- ERwin モデルから SQL スクリプトを生成した後で、定義文の SQL 文を CUT&PASTE で付加する。

例えば、ERwin に添付されている Movies.ER1 ファイルのの CUSTOMER テーブルは次のようになる。

```
------
CREATE TABLE CUSTOMER (
customer_number INTEGER NOT NULL,
customer_first_nam CHAR(15) NOT NULL,
customer_last_name CHAR(15) NOT NULL,
customer_address_1 VARCHAR(180) NOT NULL,
customer_address_2 VARCHAR(180),
customer_city CHAR(18),
```

```
customer_state CHAR(2),
customer_zip CHAR(10),
customer_phone INTEGER,
customer_credit_ca INTEGER,
customer_credit_ca TIMESTAMP,
customer_status_co CHAR(1)
);


COMMENT ON COLUMN CUSTOMER.customer_number IS 'An identifier for a customer
assigned at the first time a person rents a video.';

CREATE TABLE EMPLOYEE (
employee_number INTEGER NOT NULL,
store_number INTEGER NOT NULL,
employee_first_nam INTEGER NOT NULL,
employee_last_name CHAR(15) NOT NULL,
employee_address_1 VARCHAR(20),
employee_address_2 CHAR(20),
employee_city VARCHAR(20),
employee_state CHAR(2) NOT NULL,
employee_zip INTEGER,
employee_phone INTEGER,
employee_ssn INTEGER,
hire_date TIMESTAMP,
salary NUMERIC NOT NULL,
supervisor INTEGER NOT NULL
);

...
------
```

- **変更したスクリプトファイルを保存する。**

[ERwin での操作] Main　subject エリアにコピー元のモデルを開き、物理モデルモードにする。そして、「完全比較」（もし、ひとつひとつ定義をオリジナルとの相違を確認したいのなら）あるいは、もし自動的に全ての定義情報を変更したのなら「データベースの更新」を使用する。（訳者注：カラムディタ上の「属性定義を更新する」がチェックオンになっていることを確認する。）

- 特殊文字を使ったり、カラム名を変えたりすることがなければ、全てが正常終了し、定義情報が更新された新たな ER1 ファイルが得られる。

もう少し自動化したいと思ったんですが、高々１００定義しかないんです。何回もやるんだったら、もう少し時間をかけて取り組んだんですが。

追加コメント：

- インポートするためのデータベース機能を使用しているので、”論理のみ”とマークしてないオブジェクトが対象となる。

- データモデルのオブジェクトが新たに作られた時点で定義を行うことが最も望ましい。後になってから、バッチで定義を登録しようとすると嫌がられる。

- 良い定義は、ほかの人がモデルを理解するのにとても重要である。良い名前をつけただけでは、それが意味どおりに正しく理解されたと思ってはならない。

- オブジェクトとして使用される用語の定義を定義文中に記述することが無いように留意しなければならない。モデルにされたコンセプトを記述すべきである。
  例えば、「顧客」という用語の意味がディクショナリィに定義されていれば、エンティティ定義するときに厳密に定義する必要はない。

- 学校で習ったグレード２定義規約を厳密に適用する必要は無い。
  「発注日」の定義について以下の事例を見てみよう。
「実際にまたは概念的に組織に対して製品が要求された、時間概念のひとつ」
次のように書いたらもっとわかりやすい。
「自社で製品の注文を受け付けた日」

どちらも定義中に用語の定義の一部が含まれているが、２番目の定義は属性を定義したときにモデラーの言わんとすることがより鮮明になる。

訳：㈱CAC 真野 正

Moving Attributes to Different Entities on Reverse Engineering
リバースエンジニアリングにおいて属性をエンティティ間で移動する

Terry Fitzpatrick, P.E.
Architect and Instructor
ALM Advanced Technology Group
COMPUTER ASSOCIATES INTERNATIONAL

これは私が調べた Web <http://www.infoadvisors.com> には記述されていない Trick（裏技）である。この Trick はキーボードマクロと共に実行する。ERwin レポートブラウザを使ってデータモデルを修正する多くの人にとって大変便利なものである。

UDP（ユーザー定義プロパティー）はリバースエンジニアリングする際にソーステーブルやソースカラムをトラッキングするのに便利である。

メタデータを UDP として格納することは、データソースの追跡なしに属性を別エンティティーに移動することや命名を変更することが可能になる。

しかしながら、データモデルに数千のカラムがあれば大変なタイピングを強いられる。

キーボードマクロプログラム（Perfect Keyboad、Macro Express）は一連のタイピングやマウス操作を記録し、さらにホットキーとの組み合わせによって繰り返し同じ操作が可能となる。

リバースエンジニアリングが終了したら、属性を移動する前に、テーブル名・カラム名・ソーステーブル UDP・ソースカラム UDP のレポートを作成する。

ひとつのマクロを使ってテーブル名をそのソーステーブル UDP へペーストすることをマクロに記録します。; さらにカラム名をそのソースカラム UDP にペーストする。

その操作をホットキーに割り当てる。

例）<alt>キー＋<u>キー

ここで、次の行をクリックし、<alt>キー＋<u>キーを押下します。するとただちにコピー＆ペーストされることがわかる。

ERwin レポートブラウザは、通常、名前が重複しないように結果を表示する。

これは、テーブル名が ただ一つであるように表示することを意味する。

ここでのこの働きが Trik になる。

すべての行の内、値を移行したいカラムに対して、その 1 文字目にエクスプラネーション記号（！）を挿入することによって、このレポート結果のカラムヘッダをリネームする。

それらの結果をレポートビューとしてセーブする。

このレポートビューを実行するときに、これらがそれぞれのカラム行の入力となります。この機能はマクロが可能にする。

一度しかこのような編集をしない場合には操作を簡単にします。

それは、一行あたり、一回のマウスクリックと一回のキー押下に、TONS のタイピングを減らすことである。

ディクショナリマネージャやモデルマートに対して SQL-INSERT を実行することは、極めて危険で、かつメタデータの多大な知識を必要とする。

この方法に比較して、この Trik は、私が見つけた、最も簡単な UDP 値の大量の入力方法である。

注１）ERwin レポートブラウザとは
ERwin ダイアグラムと ModelMart の情報を参照し、レポートを作成するためのレポートモジュール。（ERwin リファレンスガイドより。また、ERwin ワークグループ・モデリングガイド第 6 章 ModelMart の参照とレポーティングを参照）

訳：㈱イートン 松岡 修司

Using Subject Areas for Logical & Physical Displays

Karen Lopez
List Mistress, ERwin Web Board (www.infoadvisors.com)
InfoAdvisors, Toronto Canada

あらゆるサブジェクトエリアについて、標準的な二つの表示（論理表示と物理表示）のためのストアドディスプレイを作成すれば、オブジェクトの配置や印刷に関する時間を節約できる。

論理表示と物理表示をストアドディスプレイで切り離すことによって､論理表示と物理表示を切り換えても,オブジェクトの再配置に何時間も費やす必要はなくなる。

また, WebPublisher(*1)を使用するとき，適切な表示を容易に選択できるようになる。

更に，印刷時にDisplayマクロ(%Display%)を指定すれば、それぞれのダイアグラムにストアドディスプレイの名前を印刷できる。

印刷のページ設定でヘッダーの設定値を、

%File%--%Display%/%SubjectArea%

とすれば、印刷時のヘッダー部は

DataModelFileName--Logical/Vendor Maintenance

となり、ダイアグラムについて、知りたいことが印刷できる。設定により、フッター部に、ページ番号とタイムスタンプも印刷できる。

*1.Platinum WebPublisher（WebPublisherは国内では出荷されておりません）

訳：㈱インテック 桶谷

属性値をビジネス名からカラム名へどのように省略するか

How to Abbreviate Attribute Business Names to Column Names In ERwin Models

Lucie S. Johnson
Bank of America, San Francisco
Bay Area Enterprise Modeling User Group

序文

このドキュメントは、ERwin/ERX を使って、論理モデルと物理モデルを構築するデータ管理者を対象にしている。

ERwin/ERX のパラメータを設定するクイックリファレンスとして使用することで、論理的なビジネスドメイン名から短い物理ドメイン名に自動的に変換できるようにすることを目的にしている。

概要

ERwin/ERX の%Lookup マクロは、属性の名前を自動的に論理ドメインのビジネス名から短い物理名に省略することができる。パラメターを正しく定義してしまえば、あとは単に論理ウィンドウから物理ウィンドウに切り替えるだけで、自動的にすべての論理モデルの対象となるカラム名が省略された物理名になる。

ビジネス名を省略する時に、この%lookup マクロは、ユーザーが作成するカンマ区切りのテキストファイルを使用する。ただし、カラム名を RDBMS の標準の長さに調整することは行わない。（例えば、DB2 の 18 文字など。）変換の結果を RDBMS の標準の長さにあわせるという作業は、アナリストの責任で行うこと。

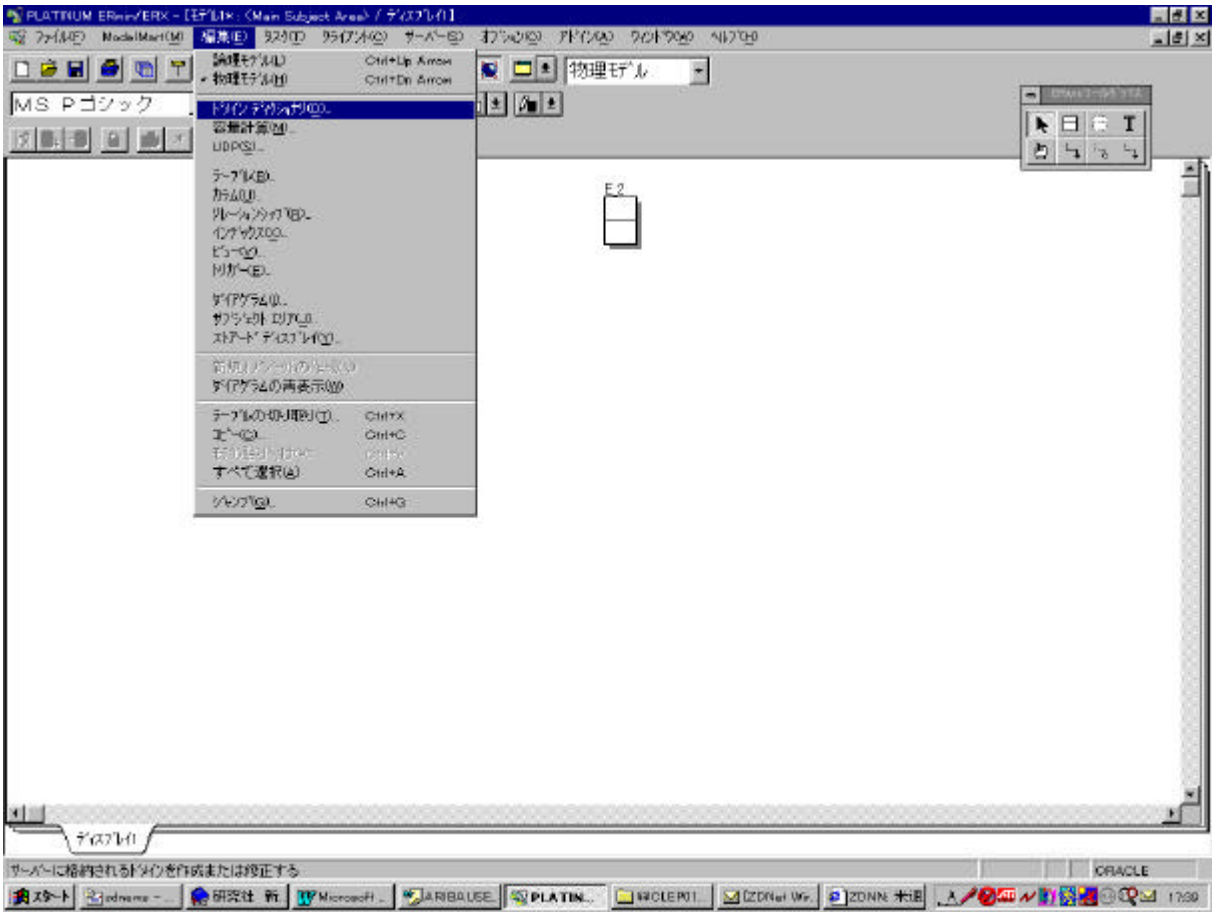
ビジネス名を変換するには、以下の２つの手順を行う必要がある。

1. ビジネス名とその略称を降順にソートしたカンマ区切りのテキストファイルを作成する。
   フォーマットの例
   Relational Database Management System,RDBMS
   name,nm
   business,bus
   Etc.
2. ERwin/ERX のドメインディクショナリエディタの物理モデルモードにおいて、全般タブの゛ｶﾗﾑに継承される名前”のフィールドに、%Lookup マクロとビジネス名と物理名を定義したカンマ区切りのテキストファイルを指定する。

新しいモデルにいちいち再入力しないようにするには、必要なマクロをすべて含んだ拡張子が.ERT の ERwin テンプレートを新規モデルのスタートアップテンプレートとして使うことができる。

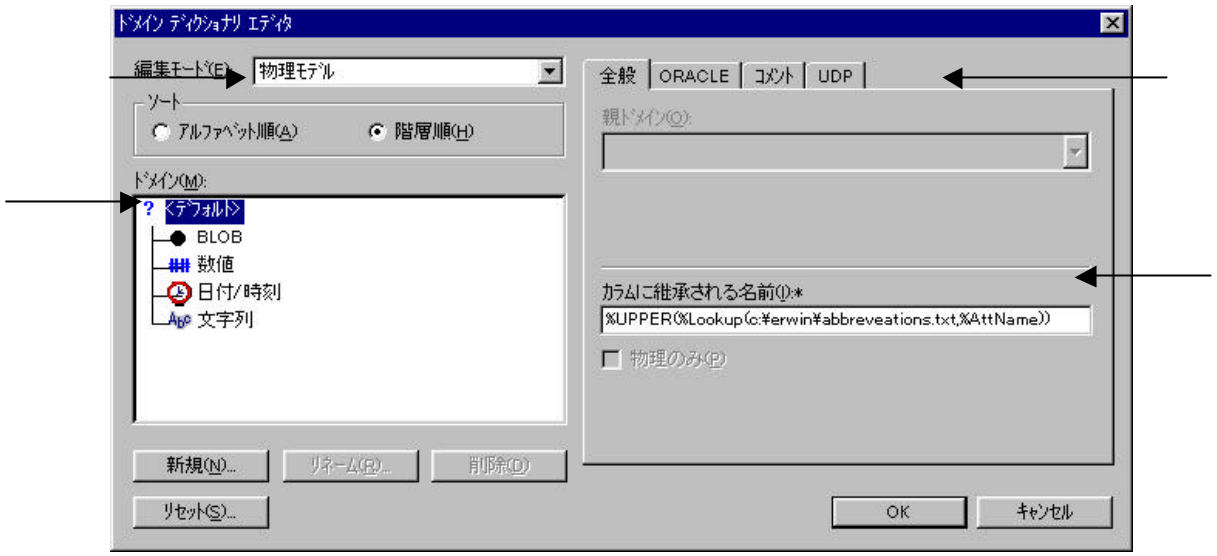
以降のページに、ERwin モデルにマクロをセットアップする方法が順に説明してある。

%Lookup マクロ設定



1) 新規または既存の ERwin ファイルを開く。

2) ドメインディクショナリエディタを開く。（上図参照）



1) **編集モードが物理モデルであることを確かめる。** (ダイアログ左上)

2) **全般タブを選択する。** (ダイアログ右上)

3) **選択されている、ドメインが？<ﾃﾞﾌｫﾙﾄ> であることを確認する。**

4) **次のマクロをｶﾗﾑに継承される名前に入力する**
   %UPPER(%Lookup(*filename*,%AttName))
   (*filename* **は省略された物理名が定義された、カンマ区切りのファイル**)
   例： %UPPER(%Lookup(c:¥Erwin¥abbreviations.txt,%AttName))

5) **OK ボタンを押す。**

この設定で、論理モデルから、物理モデルに切り替えると、すべてのカラム名がファイルに定義された省略された物理名に変換される。%UPPER **マクロは、小文字から大文字に変換するマクロである。また、スペースはアンダーバーに置き換えらえている。**

考慮すべき点

1. カンマ区切りのテキストファイルの順序:
カンマ区切りのテキストファイルの内容は、降順にソートされていなければならない。Lookup **マクロは、キーワードが一致すると直ちに省略名を使うので、昇順にソートされていると、下の昇順の例では、**workstation **は、**wrk **に変換されてしまう。それは、**work **が** workstation **より先に一致してしまうからである。**
. 同様に、複合語や頭文字はファイルの先頭に置かなければならない。

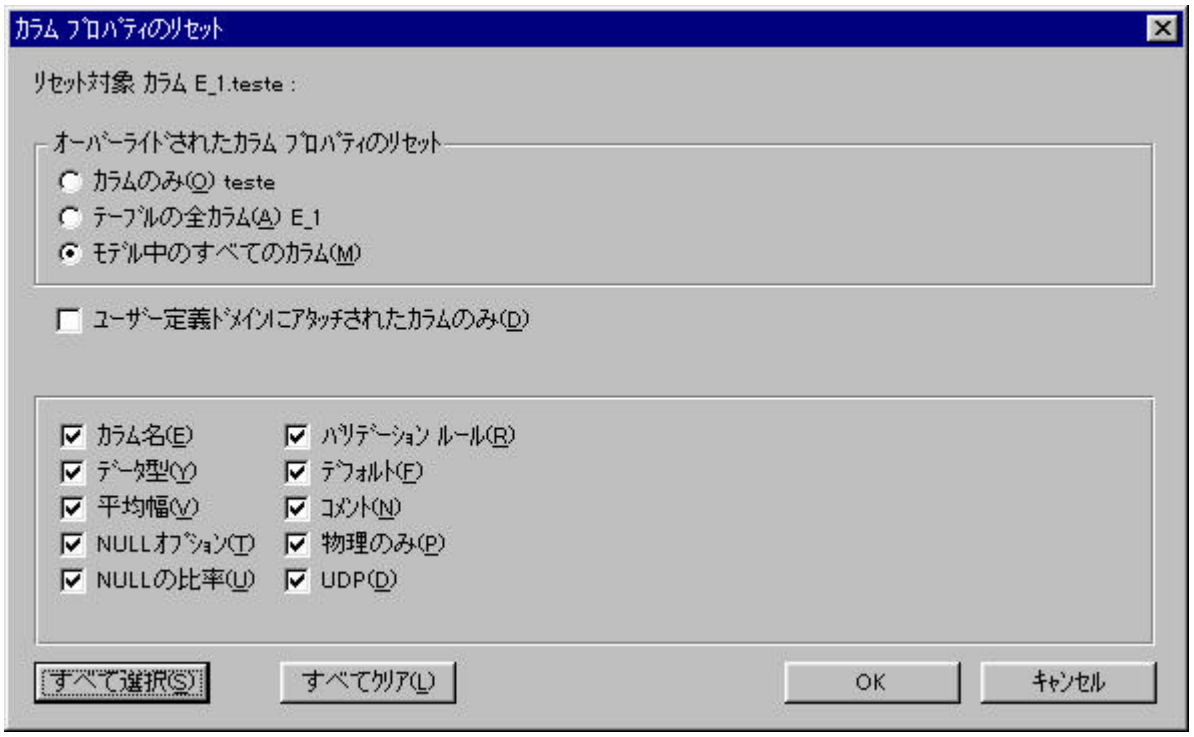
昇順:
WORK,WRK
WORKSHEET,WSHT
WORKSTATION,WSTN

降順:
WORKSTATION,WSTN
WORKSHEET,WSHT
WORK,WRK

2．モデルのカット＆ペースト:

前述したテンプレートモデルに対して、%lookup マクロを使っていないモデルからカット＆ペーストを行う時は、ペーストした結果のすべてにカラム名がﾃﾞﾌｫﾙﾄの%AttName になっている。この問題を解決するには、カラムエディタの物理モデルにおいて、カラムプロパティのリセットを行う必要がある。この操作で、%AttName は、省略されたカラム名に置き換えられる。

物理モデル図において、テーブルを選択し、カラムエディタを起動して、カラムエディタ内で、リセットボタンをクリックする。〝カラムプロパティのリセット”のダイアログで、〝モデル中のすべてのカラム”を選択し、〝すべて選択”ボタンを押す。この操作で、%AttName はすべて省略したカラム名に置き換えられる。



訳：㈱日本総合研究所 鈴村 幸太郎、木下 恵一、細川 努

「ヘッダーとフッダー」

ListMistress, ERwin Users Discussion Group
InfoAdvisors, Inc.
www.infoadvisors.com
Toronto, Ontario Canada

もし通常表示するダイアグラムのヘッダ、フッダ情報を変更しようとすると、それぞれのサブジェクトエリアに移動しヘッダ、フッタを変更する時間にイライラし、以下のトリックを使いたくなることでしょう。

いくつかの背景として、まず私は多くのソフトウエアと同じく（1.2，2.1，2.2 RFC1 など）のようなバージョンナンバー情報をもつ標準的なデータモデルを扱っている。
私はこの情報をページフッタに含めていますが、５０以上にものぼるサブジェクトエリアのヘッダ、フッダをリリースする際にいつも更新しなくてはならなかった。
以下のトリック（裏技））を使うと、今では一度の変更だけで済んでしまう。

1. メインサブジェクトエリアに移動する。
2. 元の情報となるテキストブロックを挿入する。（私の場合は、Versionx.x, Copyright 2000 Mycompany,Inc.）
3. このテキストブロックをダイアグラムの左上に移動する。
4. この情報を出したい全てのサブジェクトエリアにこのテキストブロックを加える。

これは１度だけ行う必要がある。新しいサブジェクトエリアを作成する際にも必要である。メインダイアグラムの左上にこれを設置すると、新しいサブジェクトエリアにそれぞれが現われる。
この情報を更新する必要がある場合に、一個所更新する。するとこの変更は全てに現われます。私はこの手法をバージョンナンバーと著作権注意書に使用している。

訳：協和発酵㈱ 田尻

BPWIN IDEF 外部エージェント

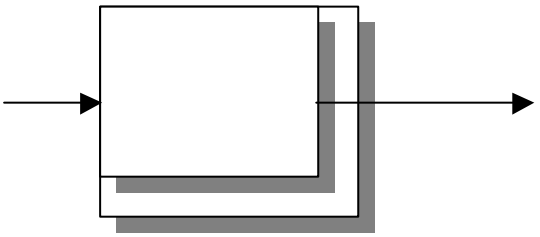
Doug Stone, Information Modeler
NUSCO (NU Service Co.) Berlin, CT.
New York Enterprise Modeling User Group

DFD でいうところの外部エージェントを利用したいと思うが、BPwin で IDEF を使用する際には残念ながらこの機能はサポートされていない。

私たちは DFD を使用することもあるし、また、ICOM などのために IDEF を利用することもある。その場合でも外部エージェントが気に入っている。

われわれが考えたトリックでは２つの Acibity Box を作成し、そのうち小さい方を大きい方の上に重ね、大きいほうの右側がはみ出るように配置することである。

こうすることで、これはシャドウがかかった外部エージェントのように見える。それから我々はこれから実際のアクティビティに矢印を引くのである。



訳：㈱富士総合研究所 一倉

日本語版の出版に当たって

松本 聰
JNT システム㈱ IRM グループ
シニア・コンサルタント
Japan Enterprise Modeling User Group President

思ってみれば、Ben に最初に会ったのは、1997 年に開催された 1st Logic Works User Conference でした。彼の作った非常にこった内容の Power Point のプレゼンテーションを見て、「さすがアメリカのプレゼン」と感心したものでした。当時の私の英語能力は本当にひどいもので、今になってみると、よく遠く New Orleans まで行ったものだと思います。

今年（00 年）、ERwin も CA の製品になってしまい、われわれのユーザー会も表向きは「Japan Enterprise Modeling User Group」となっていますが、アメリカ各地のユーザーグループの多くも、基本的には「ERwin User Group」であることには、変わりがありません。

さて、Ben とはその後、「User Group President」どうし（同士）としてのお付き合いが続いています。

Ben Ettkinger

Ben の若干の紹介。

写真でもわかるように、彼は熱心なユダヤ教徒です。ちょっと見は、ひげの感じなどで取っ付きにくいし、議論をはじめるとちょっとこわいかもしれない。（ある人がそんな感想を持った と聞いたので）しかし、話をすると非常に親切で、まじめに話を聞いてくれる人です。彼のここ 3 年のテーマはこの出版にもあるように「IDEF1X vs UML」で、この 2 つのモデリング言語の共通点、相違点およびそれをどのように活用すべきか、という一点に絞られていました。

ところで、この出版に関しては今年の初めに Ben から Mail があり、ERwin のチップス、トリックス（裏技）を出版するので、協力が欲しいとのことでした。この件について、ユーザー会の面々や日揮情報システム プロダクト販売本部（現 日揮情報ソフトウェア）とも相談し、記事およびチップスとして ICON ライブラリを送付しました。これらは Vol. 1 には間に合いませんでしたが、次ぎの Vol. 2 には掲載されると思っております。

Vol. 1 が CA User Group のホームページに掲載され、私はすぐにこれは翻訳すべきだと思いました。幸い、ユーザー会のメンバーもこれに賛成してくれ、各自担当部分を持って翻訳を行いました。

Ben からも翻訳出版に関しては「問題ない」という返事をもらい、翻訳をはじめたのですが、身の程知らずで一番長い「IDEF1X vs UML」を担当してしまい、多いに反省をしているところです。

なお、日本語版の出版にあたり、ご協力いただいたユーザー会の方々、日揮情報ソフトウェアの皆様、コンピュータ・アソシエーツの皆様に心からお礼を申し上げます。