The ERwin インサイダー™ 2

& BPwin インサイダー™

チップス,裏ワザ,記事

ERwin & BPwin
ユーザーへ

Contributions by ERwin & BPwin Users from Around the World

Edited and Complied by Ben Ettlinger

監訳 松本 聰

Volume 1 Number 2

September 2000

日本語版

2001 年 1 月

編集者の覚え書き

わずかな予算と限りある資源を考えれば、「**ERwin & BPwin インサイダー**」の第 1 版の出版は、大成功だった。

この版は、ERwin の広い輪を作り、世界中に電子的に送られた。我々のよき友である、日本 ERwin ユーザー会の会長 松本聰は、すべてのページの日本語翻訳を指揮した。これによって、「**ERwin インサイダー**」は日本の何千ものユーザーに利用可能となった。正確には、この版がどの程度世界を駆け巡ったかはわからない。しかし、電子メールによって、多くの人たちがこれを受け取ったであろう。

しかし、我々はこの版が、現在のバージョンである ERwin 3.5.2 に関しては、最後であると思っている。ハード・コピーおよび電子的プリントによって ERwin 4.0 ベータ・プログラムにフルスイングするべきである。私は、次の版が ERwin4.0 への入門についてとくに強調され、新しい ERwin4.0 の特徴についてのチップスおよび裏ワザのすべてを示すものであってほしいと真剣に思っている。これを実行するためには、あなたがた ERwin ユーザーからの支援が本当に必要である。インサイダーによる専門知識の共有は、あなた自身を含む ERwin ユーザーにすべて役立つと思っている。

第 1 版の成功および我々がユーザーと CA から受け取ったフィード・バックに基づいて、ERwin インサイダーのスコープをちょっと広げることができた。一般的なチップス、裏ワザおよびモデルリングの問題に加えて、この版では ERwin3.5.2 の議論を越える 2 つの記事を見つけるであろう。この中で「OR の準備ができている？」は、オブジェクト・リレーショナルについて語っている。これは市場が向かっている方向を示し、ERwin がバージョン 4 以降示すであろう方向づけでもある。

最終記事は、論理データモデリングへのソフトウェア・ベンダーの認識の明らかな不足に対するちょっとしたうっぷんである。

Ben Ettlinger

Editor
Data Administrator
New York Power Authority
President, NYEMUG
New York Enterprise Modeling User Group
ben.ettlinger@nypa.gov

松本 聰

翻訳・監修
JNT システム(株)
President, JEMUG
Japan Enterprise Modeling User Group

Table of Contents

Erwin Tips & Tricks

SQL Server 7.0 デフォルトオブジェクトの割り当て

Michael Levine Guelph, Ontario Canada
訳）株式会社インテック　　桶谷 貴弘

ERwin 3.5.2 SP2は適切にデフォルトオブジェクトをSQL Server 7.0にエクスポートする。
しかし, SQLデータベースのカラムにはデフォルトオブジェクトが割り当てられない。
問題の原因はsp_bindefaultプロシジャを適用することにおいて間違った構文にある:ERwin は必要に応じて引用符におけるデフォ
ルトの名前を囲まない。
この問題は,手作業で、必要な単一引用符を追加することで回避できる。
デフォルトオブジェクトの代わりにデフォルト制約条件を使用した方がいいだろう。
ERwinメニュー サーバー|デフォルト/初期値|生成オプションでデフォルト制約条件の使用を指定するできる（sp_bindefault
の代わりにデフォルトを選ぶ）。
制約条件は正常に動作するようになる。

開くまたは保存に時間を要するモデルマートモデルの扱い

Terence J. Fitzpatrick, CA（ウェブボードより）
訳）株式会社インテック　　桶谷 貴弘

1.)利用者が，メタデータの「いくつか」を見る必要があるだけならば，どのようにサブジェクトエリアエディタ(モデルマートの)と
レポートブラウザを使用するのかを示す。
最初に全体のモデルを開かないで，サブジェクトエリアエディタで利用者にモデルマートから開くためにいくつかのエンティ
ティを選択させて，作成できる。
レポートブラウザは開いていないモデルのメタデータにもアクセスできる。
2.)
まれなケースを除いて，モデルを開いた状態で，一日中オンラインで作業すべきでない。
モデルマートからモデル(または，それの一部)を開いて，次に，名前を付けて保存(モデルマートのものでなく，ERwinのメ
ニュー)を使用して，ローカルのドライブにER1ファイルを保存しなさい。
モデルで作業。
セーブしてモデルマートに戻る準備ができたら，モデルマートに再接続して，保存しなさい。
モデルマートの「スナップショット」でそれらを簡単に実行できる。
モデルマートが、ERwinモデルのあらゆるメタデータ情報をリレーショナルデータベースに格納することを忘れないようにしな
さい。
モデルを「開く」ということは、75テーブルのモデルマートデータベースから列を読んで，それらを解釈しERwinが表示すること
を意味する。

シソーラスと辞書

Ben Ettlinger, New York Power Authority
訳）株式会社インテック　　桶谷 貴弘

ERwinのスペルチェッカ, 辞書またはシソーラスを待つことはできないか?
ERwinで動作し、ベンダの要求として、いかなる他のWindowsのデスクトップアプリケーションでも動作する辞書ツールをダウン
ロードできる。
いったんダウンロードした後, セットアップを実行しなさい, あなたがしなければならないことは単語を強調表示させcntl Lを
押すことがすべてである。
ほんの一瞬でツールは、強調表示した単語が入力された状態でwww.dictionary.comを開く。
Cntl Mは, dictionary.com, thesaurus.com, Web検索, またはamazon.com(100%フリーなものは何もない)にリンクするため
に選択でオプションメニューを開くだろう。
www.cleverkeys.comでダウンロードしなさい。

ドメイン

Whit Owens, CA ERwin ウェブボードより）
訳）株式会社インテック　　桶谷 貴弘

私はドメインの誤用をよく見かける。
ERwin ドメインは'一般的な属性/カラム'として造られるべきである。
あなたの組織が'Class Words'を使用するなら, これらはドメインの非常に良い候補になる。
理想的には、ドメインは、いくつかの属性を含むことをサポートすべきである。
名前, 名,姓,住所,エリアコード,開始日は、すべての良い可能性である。
標準規格, ドメインディクショナリエディタの'属性に継承される名前'でのERwin Macrosの利用が非常に役に立つ場合がある,
Macro<%Lower(%OwnerEntity %AttDomain)>は, エンティティ名とドメイン名を使用して小文字属性を作成するだろう。
属性かカラムが与えられているどんなプロパティもドメインに与えることができて, 現在, ドメインが与えられているすべての
プロパティで属性を作成するために独立属性ブラウザからエンティティまでこれらのドメインをドラッグできる。
現在, 標準規格を使用するのは最初から何かを作成するよりも簡単である。

エラーメッセージ

Linda Jeney（ERwin ウェブボードより）
訳）株式会社インテック　　桶谷 貴弘

「オブジェクト名 ‘ pbcattbl'は無効です」
この誤りはデフォルトの設定のために発生する。
メニューのクライアント |対象クライアントを選びなし'を指定しなさい。
あなたが現在PowerBuilderに設定しているなら(デフォルトの設定の理由を私に尋ねないように--愚かな理由だが、ラジオボタンの最初がPBである…)…フォワードエンジニアリングでは問題にならないが、完全比較のエクスポートフェーズの生成で問題の原因になる。

定義情報の多量インポート

出典 :WWW.infoadvisor.com
訳)株式会社シーエーシー　真野 正

（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がそれらは、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ます。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]まず、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しなさい。**

＜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しなさい。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しなさい。何があってもオブジェクト名を変えないように指導しなさい。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なさい。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しなさい。**

- **オリジナルモデルでデータベーススキーマを生成し、保存しなさい。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**

しなさい。

- テーブル名とカラム名に対して、連結機能を使用して SQL を生成しなさい。

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しなさい。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）

- 誰も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だろう。

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、”論理モデルのみ”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でしょう。

「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もデラーの言わんとすることがより鮮明になるであろう。

以上

属性名の大量変更

出典 :WWW.infoadvisor.com
訳)株式会社シーエーシー　真野 正

－ERwin モデルを ERX ファイルに Export（保存）する。
－ERX ファイルを Workpad で開く。
（訳者註．通常は秀丸などの text エディターが便利でしょう）
－Domain Properties セクションを探す
（訳者註．/*,,,,,,,,,,
CREATE TABLE DOMAIN_PROP_VALUE,,,,,,,,,　で始まるセクションです。）
－このセクションをカット＆ペーストで別の Workpad シートにコピーし、Attributes.txt という名前で保存する
－Excel で Attributes.txt ファイルを開く。
区切り文字としてカンマを指定。
－Excel 上ワークシート上に新しいカラムを作成する。
－”CD_ANM”というプロパティタイプ上の属性名を変更してそのカラムに移行する。"_" を space"　"に変換するために、SUBSTITUTE 機能を利用する。それ以外のプロパティタイプを持つ行は、単純コピーする。
－そのカラムをコピーする
－カラム名としてペーストする
－Excel ワークシートを text ファイルとして保存する
－text ファイルを Workpad で開く
－テキストをコピーする
－旧セクションを新セクションで置き換え新しい ERX ファイルとする

エンティティ名

－Entity セクションについて、同上の操作を行い名前を変更する。

二つのセクションの変更が終わったら、ERX ファイルを ERwin に import する。

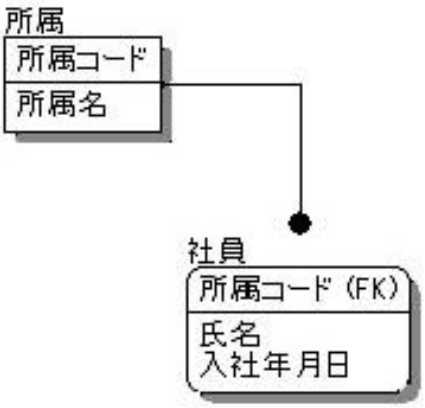
コーテーションやカンマ（DECIMAL(9,9)のフィールドのように）の取り扱いに注意しなければならない。

Excel を良く知っていれば、30 分くらいでできるでしょう。

（訳者註.）
ERX ファイルを EXCEL にコピーして､属性名やエンティティ名を変換するよりも直接 text エディターで行ったほうが早いのではないか。

EXCEL ファイルを CSV に変換出力する際にダブルコーティション(")の扱いがうまくいきません。??

◆　サンプルモデル



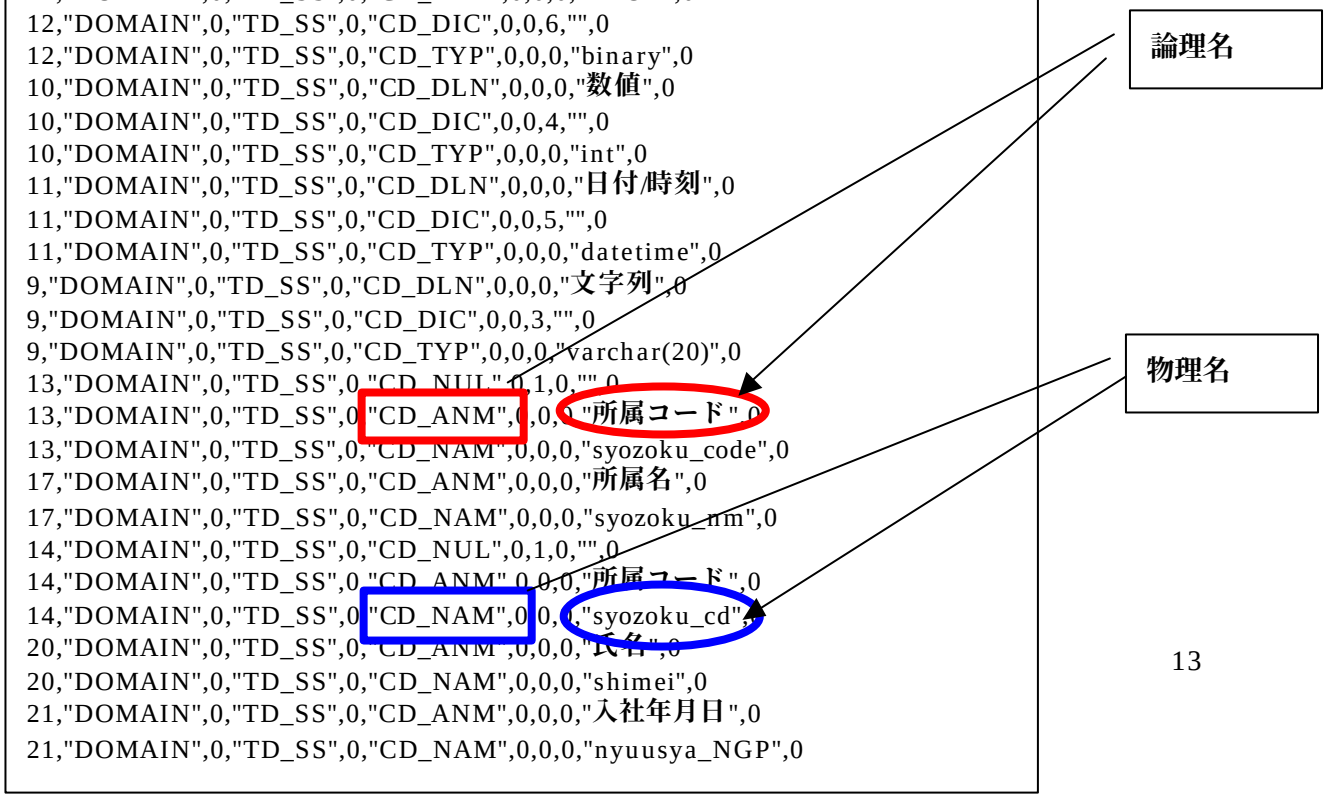
◆ Entity セクション

```
/*
CREATE TABLE ERWIN_ENTITY
     (ENTITY_ID              INTEGER NOT NULL,
      ENTITY_TYPE             CHAR(6) NOT NULL,
      DIAG_ID                INTEGER NOT NULL,
      ENTITY_NAME               VARCHAR(254),
       TABLE_NAME             VARCHAR(254),
      ENTITY_NOTE             INTEGER,
     ENTITY_DEFINITION         INTEGER
      VIEW_WITH_CHECK           INTEGER
      ENTITY_FLAGS            INTEGER
);
*/
```

エンティティ名

◆Domain Properties セクション

```
/*
CREATE TABLE DOMAIN_PROP_VALUE
     (DOMAIN_ID             INTEGER NOT NULL,
      DOMAIN_TYPE            CHAR(6) NOT NULL,
       DIAG_ID              INTEGER NOT NULL,
      TARGET_SYSTEM          CHAR(6) NOT NULL,
      DOMAINPV_SEQ          INTEGER NOT NULL,
      DOMAINPV_TYPE          CHAR(6) NOT NULL,
      DOMAINPV_TEXT_VAL        INTEGER,
      DOMAINPV_INT_VAL         INTEGER,
      DOMAINPV_ID_VAL         INTEGER,
      DOMAINPV_STR_VAL        VARCHAR(254),
      DOMAINPV_INHERIT         INTEGER NOT NULL
);
*/
7,"DOMAIN",0,"TD_SS",0,"CD_TYP",0,0,0,"char(18)",0
7,"DOMAIN",0,"TD_SS",0,"CD_NUL",0,0,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"%AttName",0
7,"DOMAIN",0,"TD_SS",0,"CD_PLB",0,0,0,"%AttName:",0
7,"DOMAIN",0,"TD_SS",0,"CD_PHD",0,0,0,"%AttName",0
7,"DOMAIN",0,"TD_SS",0,"CD_CMT",26937744,0,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_DMC",26938184,0,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_DMN",0,0,0,"%DomainName",0
7,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"%AttDomain",0
7,"DOMAIN",0,"TD_SS",0,"CD_PCM",26937576,0,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_PLP",0,23,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_PHP",0,25,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_PJT",0,23,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_PCS",0,26,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_PBM",0,110,0,"",0
7,"DOMAIN",0,"TD_SS",0,"CD_VSB",0,121,0,"",0
8,"DOMAIN",0,"TD_SS",0,"CD_DLN",0,0,0,"<未定義>",0
8,"DOMAIN",0,"TD_SS",0,"CD_DIC",0,0,2,"",0
12,"DOMAIN",0,"TD_SS",0,"CD_DLN",0,0,0,"BLOB",0
12,"DOMAIN",0,"TD_SS",0,"CD_DIC",0,0,6,"",0
12,"DOMAIN",0,"TD_SS",0,"CD_TYP",0,0,0,"binary",0
10,"DOMAIN",0,"TD_SS",0,"CD_DLN",0,0,0,"数値",0
10,"DOMAIN",0,"TD_SS",0,"CD_DIC",0,0,4,"",0
10,"DOMAIN",0,"TD_SS",0,"CD_TYP",0,0,0,"int",0
11,"DOMAIN",0,"TD_SS",0,"CD_DLN",0,0,0,"日付/時刻",0
11,"DOMAIN",0,"TD_SS",0,"CD_DIC",0,0,5,"",0
11,"DOMAIN",0,"TD_SS",0,"CD_TYP",0,0,0,"datetime",0
9,"DOMAIN",0,"TD_SS",0,"CD_DLN",0,0,0,"文字列",0
9,"DOMAIN",0,"TD_SS",0,"CD_DIC",0,0,3,"",0
9,"DOMAIN",0,"TD_SS",0,"CD_TYP",0,0,0,"varchar(20)",0
13,"DOMAIN",0,"TD_SS",0,"CD_NUL",0,1,0,"",0
13,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"所属コード",0
13,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"syozoku_code",0
17,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"所属名",0
17,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"syozoku_nm",0
14,"DOMAIN",0,"TD_SS",0,"CD_NUL",0,1,0,"",0
14,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"所属コード",0
14,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"syozoku_cd",0
20,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"氏名",0
20,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"shimei",0
21,"DOMAIN",0,"TD_SS",0,"CD_ANM",0,0,0,"入社年月日",0
21,"DOMAIN",0,"TD_SS",0,"CD_NAM",0,0,0,"nyuusya_NGP",0
```



印刷のテクニック

Submitted by Karen Lopez,
List Mistress for the Erwin Web Board www.infoadvisors.com
訳)株式会社日本総合研究所　細川 努、鈴村 幸太郎

サブジェクトエリアを使ってそれぞれの論理・物理表示にレイアウトの設定をすれば、たくさんの印刷レイアウトを保存することができます。

論理表示と物理表示が分かれているので二つの表示を切り替えてもレイアウトをなおす必要はありません。こうすれば、WebPublisher***を使うときにも適当なレイアウトを簡単にきりかえることができます。**

* WebPublisher **は国内ではサポートされておりません。**

サブジェクトエリアのヘッダーに、マクロワードを入れることができます。それを使ってそれぞれのダイアグラムに応じて、印刷時に自動的に記述させることができます。たとえばヘッダーを

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

のように設定すると、

ファイル名　――　論理／ベンダーのメンテナンス

のようになります。フッターにページ数や、印刷時刻を設定すれば、印刷したダイアグラムについての情報を知りたいときに役に立ちます。

ヘッダーとフッター

Submitted by: Karen Lopez,
List Mistress for the Erwin Web Board www.infoadvisors.com
訳)株式会社日本総合研究所　細川 努、鈴村 幸太郎

ダイアグラムのヘッダーやフッターを普通の方法で変更しようとすると、それぞれのサブジェクトエリアについて設定しなければならないので煩雑で大変です。

ソフトウエアの標準のデータモデルのバージョン情報(1.2, 2.1, 2.2 RFC1, etc.).はリリースのたびに毎回フッターに設定していたとします。バージョン変更のたびに、５０以上のサブジェクトエリアにもフッターを設定しなおさなければいけません。そこで、下のテクニックを使えば一度に変更することができます。

1. MainSubjectArea を選択する
2. 基準となる情報を入れるテキストボックスを挿入します。（私の場合、” Version x.x, Copyright 2000 Mycompany, Inc.”のような感じです。）
3. テキストボックスをダイアグラムの左上に持っていきます。
4. このテキストボックスを表示させたいすべてのサブジェクトエリアに入れてください。一度この操作をすれば、新しいサブジェクトエリアを作ったときにも左上にこのテキストボックスが入ります。

テキストボックスの情報を一箇所訂正すれば、すべての場所で変更されています。私はこの方法をバージョン番号や、CopyRight を入れるときに使っています。

物理ロール名

Submitted by: Ann Peoples,
Wachovia, Inc
訳)株式会社日本総合研究所　細川 努、鈴村 幸太郎

以下の場合を想定してみる。

物理ビューにおいて、物理のみとして作られたテーブルがあり、他のテーブルとリレーションシップがたくさんある。また、ひとつのテーブルと多重関連を含んでいる。（このテーブルは、多くの型のコードを含んだ参照テーブルであった。）主キーのカラムは（たとえば、REF_ID**）外部キーとして他のテーブルに同じ名前のままでコピーされている。多重関連をもっているので、外部キーはリネームが必要となるが、物理側では、ロール名の機能がない。**

解決策：テーブルの物理のみのステータスをはずし、論理ビューにおいて、ロール名を割り当てる。名前が変更されたら、再び、テーブルのステータスを物理のみのステータスに戻す。

リレーションシップ名の偶然の一致の解決

Submitted by: Ann Peoples,
Wachovia, Inc
訳)株式会社イートン 松岡 修司

次の状況を与えられたとしよう。ドメイン(あるいはコード)テーブルが論理モデルに「論理のみ」として存在した。それらのテーブルの外部キーは、STAT_REF_ID、DT_TYP_REF_ID**などの名前へ物理モデルで改名された。参照されるテーブルは「物理のみ」として加えられた。**

最初に、参照されるテーブルと他のテーブルのリレーションシップは引かれなかった。それは物理的なビューのリレーションシップとして決定された。

論理的なビューの外部キーはデュプリケーションの回避のために「論理的のみ」としてマークされた。

上記の問題に記述されたプロセスは、次のようである。

それらが既に存在すると ERwin **が判断したので、物理名は許可されなかった。**

しかしながら、名前はレポートに現われず、ERwin **内のいかなる場所にも見ることができなかった。**

解決策：論理的なビューの影響を受けたリレーションシップを削除し、それらを後ろに加えなさい。

Oracle 8i の接続／SQL*Net

Seen on the Erwin WebBoard
訳）株式会社イートン 松岡 修司

問題点：Oracle8.1.5 **クライアント・ソフトウエアで、**ERwin Server **をインストールした。オラクル接続**(Oracle Connection)**コマンドは実行され、**Oracle Connection **ダイアログがオープンした。**user/pw/connect string **を接続し、**connect **ボタンをクリックした。しかし、何も起こらない。**

解決策：Oracle8i **は** SQL*Net **を全くインストールしない。また、接続する** 8i **によって要求される** Net8 **は** ERwin **によって代案とは認められない。**Erwin **から** Oracle 8i **に接続する** SQL*Net **をインストールし** config **しなければならない。**

DB2 Index Defaults

Submitted by: Theo van Westrienen, Martinair Holland,
theo.van.westrienen@nl.martinair.com
訳)株式会社イートン 松岡 修司

問題点：DB2 のインデックスをデフォルトでセットできるか？

解決策：物理の DB2 特性のためのデフォルトは ERwin で実施しなくても DB2 の中でそれ自身でセットすることができる。

インデックス・タイプは DB2 の内にセットすることができる。それを ERwin においてブランクにしておけば、DB2 は生成時にデフォルトを適用する。

USING ブロックは、DB2 の内の標準デフォルトを持っている。Erwin にブランクを残した時、プライマリおよびセカンダリの量は 12 および 12(OS/390 のための DB2 v5 の中で)だろう。（訳者：よくわかりません）

DB2 には、他のキーワードのための多くのデフォルト値がある。DB2 リファレンスマニュアルにこれらを見つけることができる。

私はさらにターゲット・サーバー・セッティング(Target Server Settings)においてインデックス名マクロを使用する。しかし、「テーブル UDP」が何を意味するか理解しない。テーブル・ネーム・マクロ(Table Name Macro)は UDP ではない。個別のタブ上でテーブル・エディタ(Table Editor)にテーブル用の UDP を見つけることができる。

テーブルスペース名は物理プロパティ・タブ上のテーブル・エディタ(Table Editor)に入力することができる。さらにここでデータ・ベース名を入力することができる。これを行うためには、物理ビューでテーブルを右クリックすると、テーブル・エディタまたは物理プロパティを選択する。私は、UDP はここでは意味をなさないと思う。なぜなら、UDP（多くはマクロと結びついたもの）の使用は、実際の物理プロパティ・エディタよりも仕事を増やす。

容量測定エディタ(Volumetric Editor)では、さらにテーブルにテーブルスペース／データ・ベース名をアサインする可能性がある。ストレージ・コーナー（下の右）に、物理オブジェクト(Physical Object)エディタを入力するためのボタンをクリックする。このように入力する変更を逃す傾向があるので、私はこのオプションを使用しない。

私は、問題なしで過去 18 ヶ月、何百ものテーブルスペース DDL の生成を ERwin を使用して行っている。

インデックス・タイプをフィル・アウトしない場合、適切なオプションを備えた完全比較（もしくはモデルの更新）を行うことにより、適用された DB2 がセットした値を得ることができる。 テーブルスペースとデータ・ベースについては、物理オブジェクト・エディタで DBsync オプションを使用してもよい。

SQL Server でのユーザ定義データ型

A question from the Erwin WebBoard

訳)株式会社イートン 松岡 修司

質問)

いくつかのデータ・ベース・プラットフォーム（例えば、SQL Server、Sybase）へのフォワード・エンジニアリングを行うとき、相互に働く 2 つのオプションがある。
スキーマの下で、sp-addtype と呼ばれるチェックボックス、ユーザ・データ型と呼ばれるチェックボックスである。これがチェックされた場合、ERwin はストアド・プロシジャーを生成する。その「カタログ」ユーザはすべて SQL Server の IAB の中でデータ型を定義される。その後、sql の中のデータ型を生成するとともに、これらのデータ型に関連したカラムを参照する。たとえば、実際の物理特性（例えば varchar(20)）の代わりに。こんなことをする人がいるか？

答)

Spatial データ・オプションを持った Sybase を使用する場合、Spatial の関連するデータ・カラムを ddl において、ポイント、Polygon、ラインなどとして定義することは必要である。Sybase の Spatial サーバーはこれらの「ユーザ・データ型」を適切な Spatial の情報を保持

する追加のカラムあるいはテーブルを作る手掛かりとして使用する。

Magnificent Macros

訳）ドコモ・システムズ株式会社 川瀬 智子

ウムラウト（訳者注：ドイツ語のウムラウト符号 ¨ をつけたａ，ｕ，ｏのこと）を論理モデルから物理モデルに反映する
論理モデルにおいてドイツ語のウムラウトを使用したときは、以下のマクロをつかって物理モデルを生成することができる。

Entities -> Tables
このマクロは、ドイツ語のウムラウトを置換し 18 文字以内（我々が使っているデータベースでの最大文字数）にカットしたエンティティ名称を生成する。

Attributes -> Columns
このマクロは、ドイツ語のウムラウトを置換し１８文字以内（我々が使っているデータベースでの最大文字数）にカットした属性名称を生成する。

注意："β"のようなウムラウト以外の特殊文字でも同じように置換することができるが、我々の論理モデルにおいてはウムラウト以外の特殊文字は使用していない。

ブラウザの種類や言語設定によってはこのマクロでウムラウトが正しく表示されないことがあるかもしれないが、共通的な考え方を理解してほしい。

Macro for Domains
Tips&Tricks **の** Domain **の章をみてください。**

Modeling Issues

訳）ドコモ・システムズ株式会社 川瀬 智子

アドレスで最も一般的なものは、住所エンティティの属性や、社員エンティティの住所属性としてモデル化されるものである。
これらの属性で最も頻繁に使用されるのは、州(State)、市(City)、通り(Street)、番地(Address)、etc である。
一方、ほかの国々においては、アドレスをこのようなよくあるフォーマットでは表現することのできない場所が数多く存在する。
これは、国際的な郵便用住所（また、他のタイプのアドレス）をサポートすることのできる柔軟なモデルについてのひとつの提案である。

以下に示す基本モデルから始めてみて、それから、個別のニーズに合うように変えてください。

表記法：Information Engineering.
3 個のエンティティタイプ：
(1)アドレス（PK はアドレス ID が最小限であるが、顧客 ID＋アドレスタイプ(bill-to, ship-to, etc.)のような複合アドレスでもよいし、またその他のものでもよい）
(2)アドレスライン（PK はアドレスエンティティの PK＋通番。属性として、Text 属性とアドレスラインタイプエンティティの PK を外部キーとして含む）
(3)アドレスラインタイプ（PK はアドレスラインタイプ ID。属性として description 属性などがあるろう）
2 個のリレーションシップ：
(1)含む(Containing)：アドレスエンティティとアドレスラインエンティティに１対多で定義する。アドレスエンティティ側は必須で、アドレスラインエンティティ側は任意である。
(2)定義する(Defines)：アドレスラインタイプエンティティとアドレスラインに１対多で定義する。アドレスライン側は任意で、アドレスラインタイプ側は必須である。

アドレスラインタイプの値のドメインは、street, PO Box, City, State, Province, Region, Country, Attention, Department, Floor, Mailstop, などである。
この基本モデルは１つのアドレスエンティティに対し、通番属性に通番を設定することでアドレスラインエンティティを必要な数（複数）分、対応させることを許している。
この基本モデルの拡張の１つは、国ごとの明確な規則を定義することである。規則というのは、国（国は追加のエンティティタイプ）ごとにどういう順番でどのアドレスラインタイプが現れるかという規則である。
（訳者注：例えばアドレスラインタイプとして郵便番号、都道府県、市、区、町、番地を考えたときに、日本なら住所は１：郵便番号、２：都道府県、３市、４：区... という順番であるという定義を行うことである）
以上によって、正しいモデルを使いモデルのデータとして表現することで、データ構造体を使用

するよりもむしろ、うまく表現が可能である。
アドレスタイプ(bill-to, ship-to, email, IP, intra-company, mailstop, etc)をどのように関連づけるかは、どのようにアドレスを管理したいかに依存する。

OR への準備はできているかい？

Ben Ettlinger
Data Administrator
New York Power Authority
President
New York Enterprise Modeling User Group
訳)松本 聰
JNT システム(株)
IRM グループ シニアコンサルタント
President
Japan Enterprise Modeling User Group

1 つのコーナーは、C. J. Date の「第 3 の宣言」の重要性である。
(http://www.firstsql.com/dbdebunk/cjd1a.htm)

もう一つのコーナーは Joe Di Santis であり、オブジェクト指向データベースの提案者である。
(http://www.databasetrends.com/99.december/12.per.thechanging.html)

一方、データ・ベース思想の指導者はコーナーの外であり、中心の輪は現実に対処しなければならない我々論理データ・ベース・デザイナーである。現実には、あなたが Date およびかれの「第 3 の宣言」の支持者でなくても、また彼がオブジェクト指向データベースが大嫌いなことによらず、Object Relational は、ハイブリッドなオブジェクト・リレーショナルのパラダイムは、あなたの側にあり、我々はその環境の中でモデル化するために準備をしたほうがよい。

Oracle、DB2 および SQL Server を含む主なデータ・ベース・プレーヤーはすべて、新しい製品を用意した。それは、オブジェクト。リレーショナル・パラダイムであり、新しいオブジェクト・データ・ベースをリレーショナル製品の中に包含するものである。いくつかの製品は、オブジェクト・リレーショナル・パラダイムを「拡張リレーショナル」であるとしている。それはそれらがオブジェクトよりもリレーショナルであることを強調している。

オブジェクト・リレーショナルとは何か。 オブジェクト・リレーショナル・データベース管理システム（ORDMS）の公式の定義はない。しかしながら、ORDBMS は、SQL 標準の拡張である SQL3 および 4 で定義されるだろう。

どのように ORDBMS はデータモデリングに影響するか。

IDC Bulletin #14821E - August 1997 において Steve McClure は、非常に適切にそれを要約している。「SQL3 標準の引用によれば、ORDBMS はテーブルに OO 指向をを加えることを試みるデータ・モデルである。」すべての永続（オブジェクト）情報(データ・ベース)はすべてテーブルの中にある。しかし、表形式エントリーのうちのいくつかはアブストラクト・データ・タイプ（抽象データ型：ADT）と名付けられ、より豊富なデータ構造を持つことができる。抽象データ型は基本的なアルファニューメリック・データ・タイプを組み合わせることによって構築されるデータ・タイプである。抽象データ型のサポートは魅力的である。なぜなら、新しいデータ型に関連したオペレーションやファンクションは新しいデータ型のコンテンツによるレコードのインデッ

クスや格納や検索に使用できるからである。（例えば、マルチメディア）

これらの新しい概念は、埋め込みあるいはサブテーブルおよびユーザ定義データ型を含んでいる。1 つは、UML の中で必ずしもモデル化する必要がない。これは SQL3 標準の中で使用される用語である。またオラクルではネストテーブルおよびアレイ（配列）と呼ばれる。UML、統一モデリング言語は、オブジェクト指向パラダイムのために一般に認められたグラフィックの表現言語である。リレーショナル・データベースのコンテクスト内で、新しいデータ・ベースが構築するのは単にリレーショナル・モデルの拡張となり、われわれが使用してきたモデルリングツールを使用して、モデル化することができる。モデルリングツールがデータ・ベース・ベンダーと歩調を合わせ、シンボルを含み、あるいは同種のものとしてリレーショナルファミリーの構築に採用される。

ちょっと、1 ステップ戻ろう。それは、働いている環境に依存しており、互いの不可欠な部分か、でない場合があるアプリケーション開発の特別の概念を明瞭に区別するためにこの時点で、重要である。オブジェクト指向、オブジェクト指向データ・ベース、オブジェクト・リレーショナル・データベースがあり、コンポーネント・ベース開発、それは時々オブジェクト開発と呼ばれるものがある。

Webopedia に定義されるように、「オブジェクト指向は、それがどのようにして使用されているかに依存する異なるものを意味することができる、ポピュラーな愛用語である。オブジェクト指向プログラミングは、再使用可能なオブジェクトを作成する機能と、データ構造を組み合わせる特別のタイプのプログラミングに言及する。」[1]

純粋リレーショナル・アプリケーション環境では、関数型言語が使用される。これは、データと機能が明確に個別の対話するユニットとして存在することを本質的に意味する。この環境のデータ部分はリレーショナル・データベースによってコントロールされる。オブジェクト指向アプリケーション環境では、データおよび機能が、単一のユニットに組み合わせられる。このユニットはオブジェクト指向データ・ベースである。オブジェクト・リレーショナル・データベースは、リレーショナル・データベースの十分な機能性と、オブジェクト指向構築を組み合わせたハイブリッドである。

そうでなければ、オブジェクトあるいはコンポーネントベースの開発という用語は、システムを統合するために使用される。第 1 に異なるタイプのプレゼンテーション、プログラミング、インターフェースオブジェクトに適合する。そして、単一もしくは複数アプリケーションで再使用される。純粋オブジェクト指向システムは、プレゼンテーション、プログラミングおよびデータを含むすべてのレベルに再使用可能なオブジェクトを含む。

他方、オブジェクトまたはコンポーネント、オブジェクト指向開発を実行することは、データ・ベース・プラットフォームに関係なということを理解するのは重要である。

データ・ベースは 4 つレベルの中で一番下の位置にある。レベルのうちのいかなるものあるいはすべてがコンポーネント・アーキテクチャーを利用することができる。（下の図を見よ）

データ・ベース層の上に、データ・ベースとビジネス・ロジック層を含んでいるプログラム・コード間のコミュニケーションを提供するデータ・インターフェース層がある。

[1] www.webopedia.com

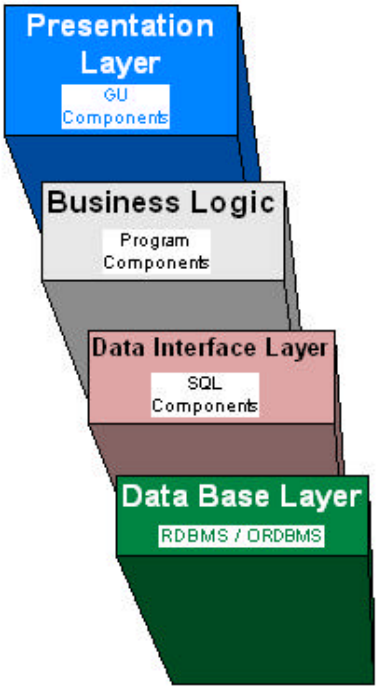


図 1

トップの層が、GUI（アプリケーション･フロント・エンドである）で構成される一方、ビジネス・ロジック層はプログラム・コードから構成される。オブジェクト・コンポーネントはいかなるあるいはすべてのレベルででも存在することができる。また、データ・クラス（ボトム・データ・ベース・レベルから描ける）はこれらのコンポーネントに含まれている。クラス、それはデータ・ベース層のインスタンス化であるが、データおよびメソッドもしくはオペレーションの両方を組み合わせる。（プロセス）メソッドまたはオペレーションは、データ・ベース・プラットフォームがリレーショナル・オブジェクトである場合、自身はデータ・ベース層にある。プラットフォームが厳密にリレーショナルな場合、メソッドもしくはオペレーションはビジネス・ロジック・レベルに存在する。

UML はオブジェクト開発のための一般に認められた言語である。この点から、常に上記の層は、オブジェクト開発が実行される、適切な UML ダイアグラムを使用することができる。UML は 1 セットの図形で構成される。例えばユースケース図、アクティビティ図など。データ・ベース層に最も適切なダイアグラムはクラス図（ER ダイアグラムの UML 等価物）である。

これは ERwin ユーザーにとって何を意味するか？ もし、OODB で設計するとどうするか？ UML 設計をするか、ERwin を使うか？ オブジェクト･リレーショナル環境で ERwin を使うことができるか？

すべてのアプリケーションは、オブジェクト・リレーショナルに導かれた特徴を必ずしも必要としない。最も伝統的な金融のアプリケーションは、新しい内容を要求しない場合がある。マルチメディアを使用する地理情報システム（Geographical Information Systems：GIS）データ・ベースおよびアプリケーションのような、より新しいアプリケーションで大きな支援をする。Oracle 8i、DB2 7.0、Sybase 12.0 および SQL Serever 7.0 への接続というオブジェクト・リレーショナル・パラダイムの実現は、バージョン 4.0 に不運にもアドレスされないだろう。しかしな

がら、バージョン 4.5 リリースがこれらのデータ・ベース・プラットフォーム、および新しいデータ・ベース・バージョンの全体の問題をアドレスするということを示している。事実、4.0 のリリースの直後に現われるという楽観論がある。最終的な結論は、4.5 が新しいものをアドレスしなければならないということである、しかし、どのようにして? どのようにしてネストのテーブルおよび変数配列（アレイ）、オペレーションあるいはメソッドを表わすか、Distinct あるいはユーザ定義データ型、ROW TYPE を表現するか?

ISO および ANSI による開発中の SQL3 標準も IDEF1X 標準（1993 年以来変更されたようには見えない）も、リレーショナル・オブジェクト構築の視覚的な表現には対処していない。

もし、現状の IDEF1X の形式が ORDBMS へ適切に使用できない場合、それを忘れてしまい UML のクラス図を使用するか。できますか？ しかし、私は警告したい。UML を学習することは簡単ではない。UML はすべてのための肝要な要素ではない。多くのエキスパートはデータモデリングのための UML の有効性に疑問を持っている。

Dave Hay、経験を積んだモデラー、「Data Model Patterns- Conventions of Thought」(Dorset House)の著者―オラクル ER 記法の熱烈なファンは、次のことを主張する「「オブジェクト指向分析」そんなものは存在しない。」、単にオブジェクト指向の設計、そして「UML はビジネスマンとビジネスを分析するのにふさわしくない。」[2]

私は Dave Hay に同意する。UML のクラス図はデータモデルを表現するには理想的ではない。

Terry Halpin、Object Role Modeling の発明者でありマイクロソフトの Visio 部門のデータ・ベース設計の技術的なリーダーは、次のように述べる「ER モデリングは、属性を持っており、リレーションシップに参加するエンティティでアプリケーション領域を見る。
この見方は全く直感的である。また、オブジェクト指向のアプリケーションをモデル化の UML の最近の上昇にもかかわらず、ER はなおデータ・ベース・アプリケーションのための最もポピュラーなデータをモデル化するアプローチである。」[3]

それでは、OR 環境中で ER モデルリング言語を進めることはどういうことなのか?

皮肉にも、オブジェクト・リレーショナル・データベースをモデルリングするために旧 Logic Works のツールだった OR-Compass は、「O」と「R」を融合した 製品として用意を整えていた。バージョン 1.01（それは結局ただ一つのリリースだったが）では IDEF1X を拡張し、ビュー（その後 ERwin に加えられた）とテーブル・メソッドとタイプ・テーブルの 3 つの記法拡張を導入した。OR-Compass は、物理モデルだけが与えられ、構築のために新しいものをすべて組込むことになっていた。そして、ERwin との統合の話さえ出ていた。

インフォミックスは、最初の主なオブジェクト・リレーショナル・データ・ベース・ベンダーだった、OR-Compass の努力はすべてその方角へ集中された。
数年前にユーザ会で尋ねられた時、なぜか他のベンダーは関心がなかった。答えは、インフォミックスが製品を既にそこに持っていたということだった、他のベンダー持ってなかった。デシジョンはしたがって、インフォミックスについていくために行われた。製品の運命を決める決定だった。そして、Logic Works から CA にいたるどこかで中止された。

---

[2] Entity Relationship Modeling from an ORM Perspective, Dr. Terry Halpin, 12/99, The Journal of Conceptual Modeling

[3] ibid

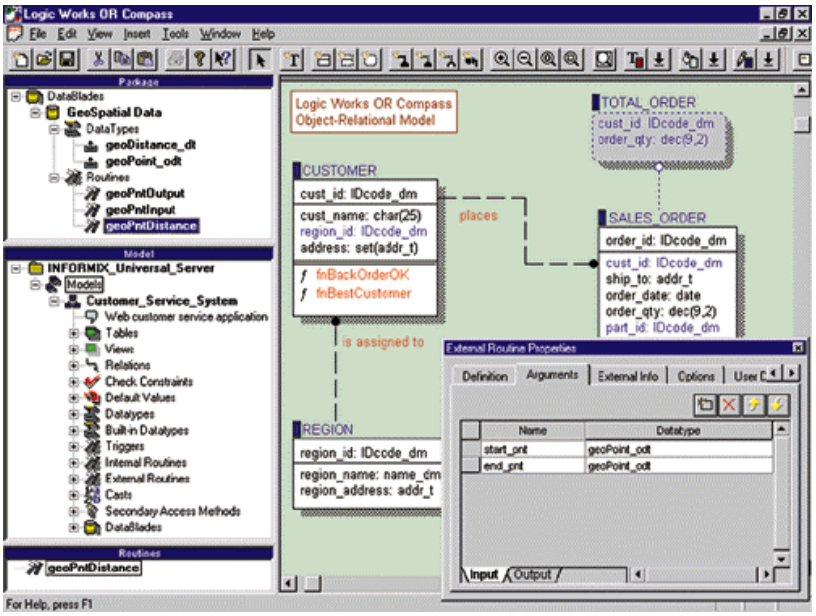


図 2 - OR-Compass のスクリーン・ショット

どのような拡張が ERwin に加えられるか。近い将来の ERwin が、現バージョン中で既に導入された関連するオブジェクト拡張をモデル化するデータに備えなければならない。ユーザに定義オペレーター、拡張可能なオプティマイザーあるいは関連を持たないインターフェースのような拡張である。

私たちは最も精通している、オラクル 8i の拡張についてマークする。他のベンダーは類似した拡張を導入した。

- **Object Types（オブジェクト・タイプ）**：抽象データ型あるいはユーザ定義データ型と呼ばれるもの、これらはグルーピングあるいはアグリゲート、同じあるいは異なるタイプの属性である。例えば、アグリゲートの電話番号は、国コード、市外局番、交換および拡張を含むことができる。住所のアグリゲートは通り、都市、州、国および郵便番号を含むことができる。
- **Collection Types（コレクション・タイプ）**: Oracle 8i は 2 つのコレクション・タイプ、VARRAY およびネスト・テーブルを持っている。VARRAY はマルチ・ヴァリューのあるカラムである、ネスト・テーブルは別のテーブル内に埋め込まれたテーブルである。
- **Pointers（ポインタ）**：データ・ベースあるいはもちろんフラット・ファイルのいずれかに、他のどこかを指すデータ・タイプ(REF、OID)を持っている。(Join の必要を除去する。)
- **Object Views（オブジェクト・ビュー）**：上記すべてのビュー。
- **Methods（メソッド）**：これらは直接テーブルに関係しているオペレーション、手続きあるいは振る舞い（objectese）である。 (これはトリガと同じものではない。) メソッドは、タスクを実行するためにテーブルに含まれ、データと一緒に働く。
- **Large Objects (LOBs)**: これは、BLOB、CLOB、NCLOB、BFILE のデータ型を含む。

既存の標準に基づき、OR-Compass に導入され、いままで見慣れたもの、それは次の図を描くのに合理的である。オブジェクト・モデリング記法と同じように、オペレーション・セグメント

は、エンティティ/テーブルの底に加えられる。エンティティ/テーブルのボックスのより下のラインは、箱の一番上のラインがキーおよび非キー属性を分離するのと同じ方法でカラムおよび属性からメソッドを分ける。エンティティおよびテーブルによって使用されるメソッドはすべて、この下のセクションに現われる。括弧(すなわちコロンに続くこと)中で、メソッド名の右に、メソッドによって使用される属性/カラムのリストになる。

運用上のセクション中の属性およびカラムを見る能力を付ける/付けないのオプションがさらにあるべきである。それは、現在利用可能な他方のディスプレイ・オプション、オプションに似ている。（データ型、NULL など）運用上のセクションをすべて隠す能力。

図 3 - オペレーション・エリアを持ったエンティティ

テーブルにオペレーションを現実に付ける能力は ERwin 3.5.2 に存在する。オペレーションはテーブル・レベルストアード・プロシジャーと見ることができる。ERwin3.5.2 テーブル・エディターの中でテーブルで生成され関係することができる。

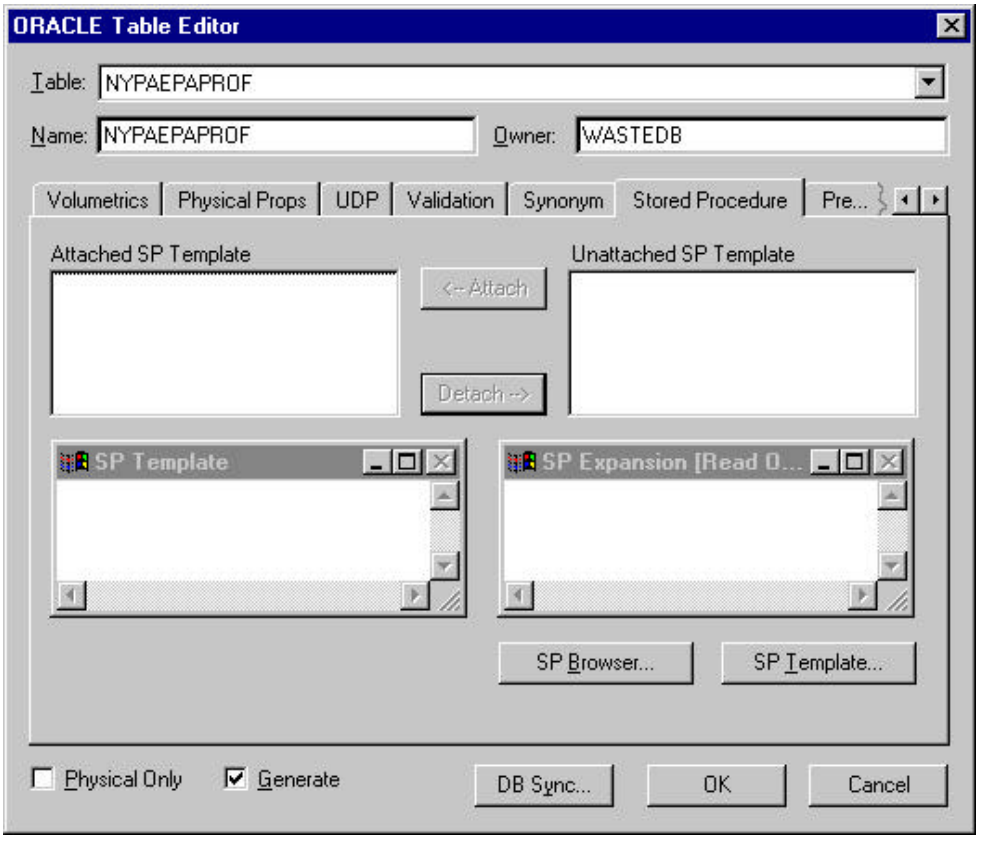


図 4 - ERwin のストアード・プロシージャ・エディタ

ストアド・プロシジャー・エディターが同様にちょうどメソッドのエディターでありうるというちょっとしたの考えは、単にちょっとした考えである。メソッドは、Java、C および PL/SQL などの様々な言語で書くことができる。新しいホルダー・タブは ERwin の内にメソッドの構築を許すために作成されなければならない。あるいは、他のどこかにあるコードに付けられる。

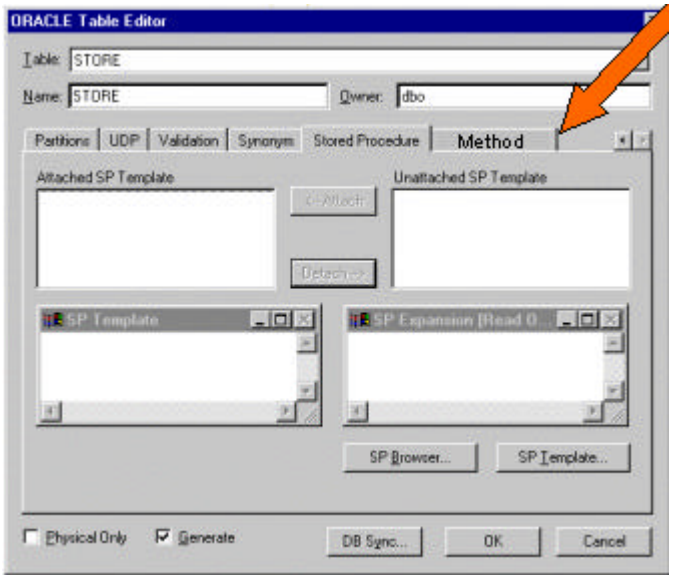


図 5

バージョン 4.0 では、新しい描画オブジェクトの使用により偽りのオペレーション・エリアを作ることができる。エンティティの下にオペレーション・エリア・ボックスを作成し、テキストを書き入れて、次に、エンティティを含んだ描画オブジェクトを「グループ化する。」グループの特徴は、Power Point の特徴と同様である。それは、 ユーザーが様々な描画オブジェクトを描くことができ、それを 1 つのオブジェクトにグループ化することができる。もちろん、描画オブジェクトに機能性はない。非常に原始的であるが、視覚的な結果は得られる。モデル化する見地から、ほとんどの新しいデータ・タイプに対する準備をすることが目の前にある。それらを利用可能なデータ・タイプのパレットに加えなければならない。バックエンドで、ERwin は何かを生成しなければならない。DLL がフォワード・エンジニアリングのために要求される。

REF と OID を含むポインターは単にデータ型リストへの追加でありうる。明白に、すべてのオブジェクト拡張のように、ERwin は適切な DDL を生成しなければならなかった。コレクション（別名：アレイ（配列）およびネストテーブル)を表わすことはもう少し巧妙である。 コレクションは、IDEF1X 用語で言えば、別のテーブルへ埋め込まれた 1～N の非依存関係テーブルである。埋め込まれたテーブルは、関係の「多」側に明白に存在する。

属性をグループ化する概念は、IDEF1X に従事する人に全く合わないわけではない。FIPS 184[4]、トマス・ブルース[5],にグループ属性に関する任意のものを置くことができなかった。「要素と呼ばれる他の属性のコレクション」として、またそれらを表わす方法の紹介としてグループ属性を定義することができる。[6] 実際、アレイおよびネストテーブルで、アレイと呼ばれる 1 つのデータ・タイプおよびネストと呼ばれる 1 つのデータ・タイプを単純に視覚的に表わすことができる。アレイあるいはネストテーブルと命名されるデータ・タイプを選ぶ場合、属性エディターはそのデータ・タイプに属性を登録してくれるようにユーザにたずねるコレクション・サブウィンドーを表にすることができる。

コレクション、データ・タイプを備えた属性、記述などである。この解決策はダイアグラムを単純にするが、実際には視覚的に全体図を与えるものではない。

アレイ対ネストテーブル

「ネストテーブルは、サイズとして無制限であり希薄になりうる。したがって、ネストテーブル内の要素は DELETE 手続きを使用して、削除することができる。可変サイズアレイは最大のサイズを持っており、データ・ベースに格納された時それらの秩序および添字を維持する。可変サイズアレイは、アプリケーションがバッチ・アレイスタイルのデータを処理するバッチ・オペレーションに適している。」

(Oracle 8i Web Design, Oracle Press/Osborn/Mcgraw Hill, 2000)

「VARRAY の内のオブジェクトは、最適の実行を実行し、マスター・レコードを直接蓄える。ネ

---

[4] Federal Information Processing Standards Publication 184, 1993 December 21. FIPS 184 is the Federal standards document which defines IDEF1X.

[5] pps. 102, 119, 135 Designing Quality Databases with IDEF1X Information Models by Thomas Bruce 1992 Dorset House ISBN 0-932633-18-8

[6] ibid 136-138

ストテーブル内のオブジェクトは、マスター・テーブルと無関係に現実に格納され、したがって最適な検索のためのインデックスを要求する。」
(Oracle 8 Design Using UML Object Modeling, Oracle Press/Osborn/Mcgraw Hill, 1999)
これ以上の議論は、この記事の範囲外である。

これは、ブルースの考えに多少似ている、ネストテーブルおよびアレイはエンティティ/テーブル・スタイル・ボックスの境界線によって表わされることを連想させられる。

アレイの名前には、「A_」が付けられ、ネスト・テーブルの名前には「N_」が付けられる。この表現法はビューの表現と同一である。（ビューに「V_」が付けられるように）コレクションがどのテーブルに属するか示すリレーションシップのラインは標準のIDEF1Xのリレーションシップのシンボルではあってはならない。従って丸（これが異なるタイプのリレーションシップであることを示している）を備えた実線のライン（依存性を示す）のような新しいシンボルの導入である。

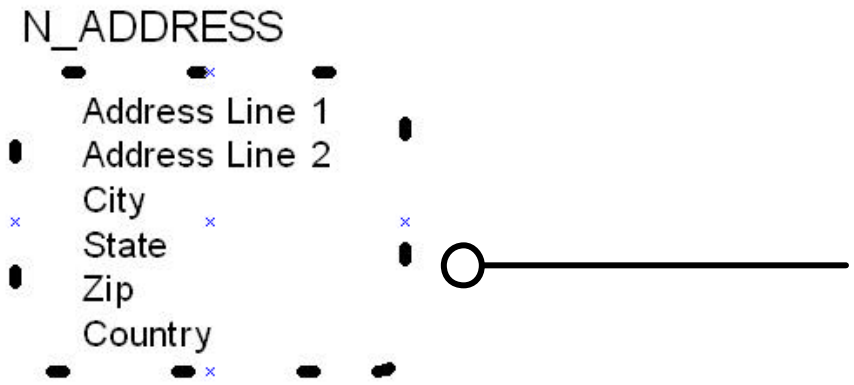


図 6 - コレクション

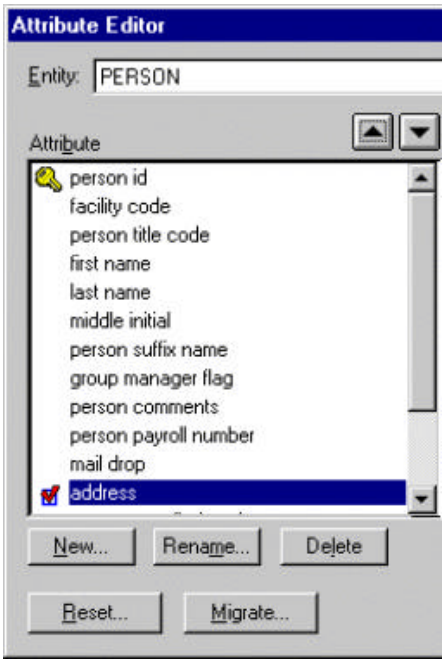


図 7 - Collapsed コレクション

もしアレイあるいはネストのテーブルがモデルの中で一度使用されれば、この解決は受け入れ可能だろう。もしアドレス・アレイがデータ・ベースで多数使用されるなら、ダイアグラムは、散らかり Busy な形になるだろう。代替の解決は拡張可能なカラムのグルーピングを導入する。その後、アレイあるいはネストテーブルは他のカラムおよびデータ・タイプとして collapsed 形式で現わされる。

collapsed カラムに隣接している、指標になる＋や↓や🖱は、それを示すエクスプローラのように、このカラムは拡張することができるネストテーブルであることを示す。

拡張オプション上のスイッチングは埋め込まれたテーブルの十分なディスプレイに帰着するだろう。

図 8 - 拡張コレクション

コレクションのために必要になった属性/カラムが頻繁に用いられているグループの非常に有効で効率的なツールとしてドメイン・ディクショナリ・エディターを使用することができる。ディクショナリ・エディターは指定者とグルーピングを作ることを拡張することができる。たとえば、モデラーがテーブルにコピーされた時埋め込まれたテーブルになる一つとしてのグループを指定することを可能にするような、一般的かあるいは DBMS ホルダー・タブの下の箱かラジオ・ボタンあるいはチェック・ボックスである。

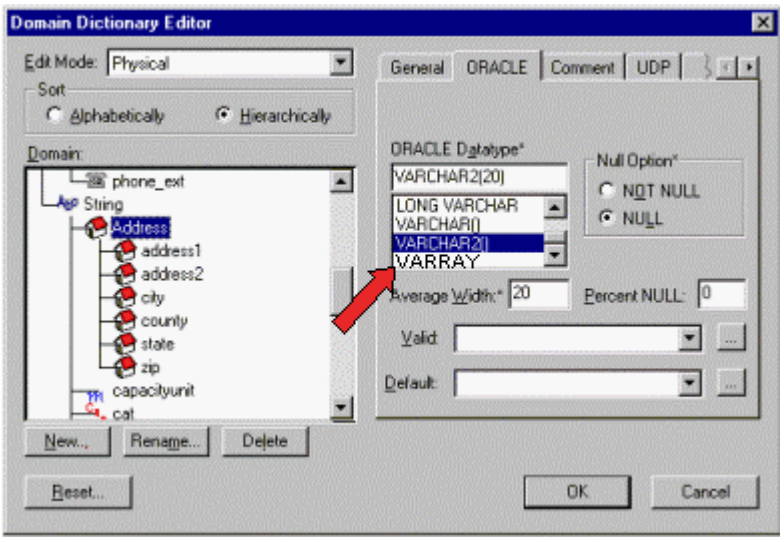


図 9 ドメイン・ディクショナリ・エディタ

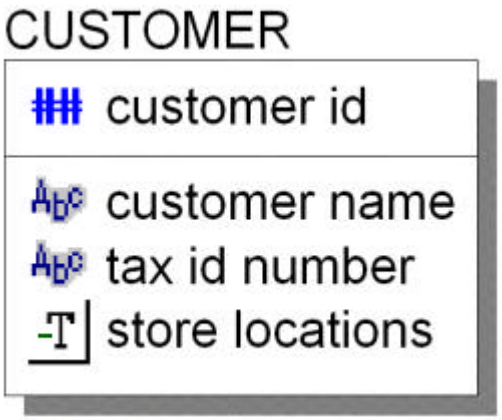


図 10 Collapsed コレクション

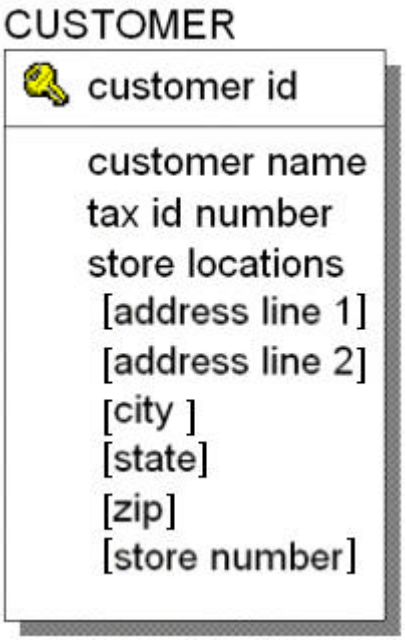


図 11 拡張コレクション

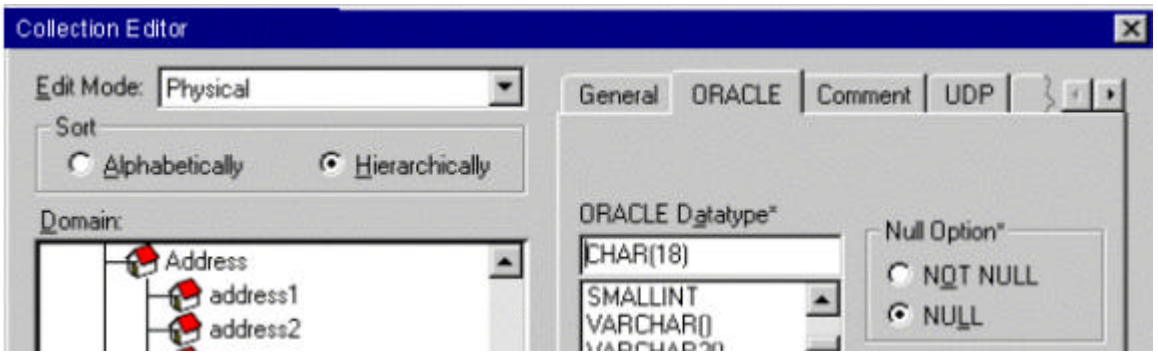


図 12 コレクション・エディタ

コレクション・エディターの追加は適切な場合がある。その場合、ERwin はある形式でコレクションを作成することを用意しなければならない。

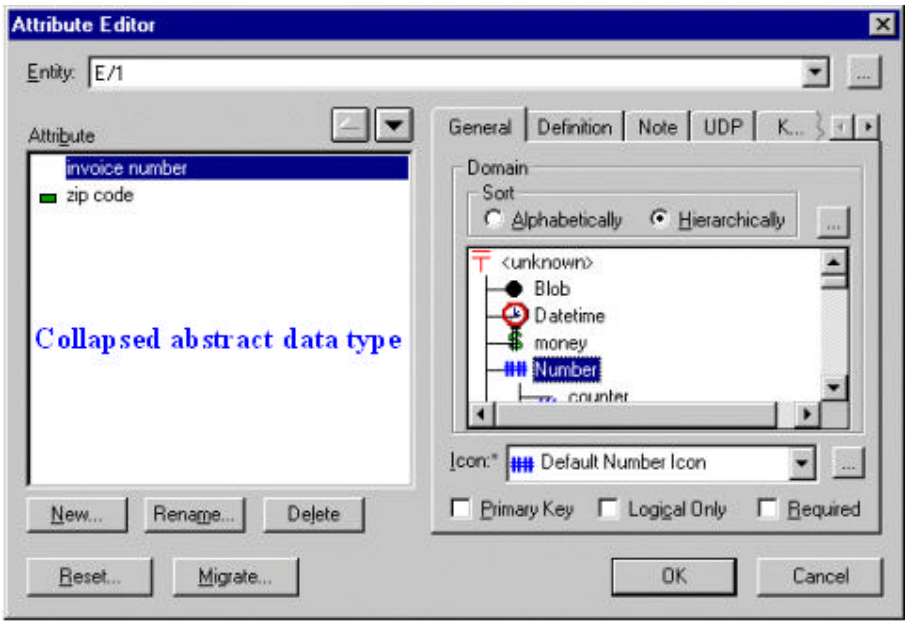


図 13 Collapsed 抽象データ型

オブジェクトタイプ、それはユーザのニーズに適合するために抽象データ型を作成することができると言う形で知られている。

4 桁を加えた郵便番号（Zip code）あるいは国コード、市外局番、ローカル番号からなる電話番号、拡張番号はこれがどこで有用になりうるかのちょうど良い 2 つの例である。コレクションに似た方法でこれらを扱うことができる。下記の図は、抽象データ型を表示することができることを示す。

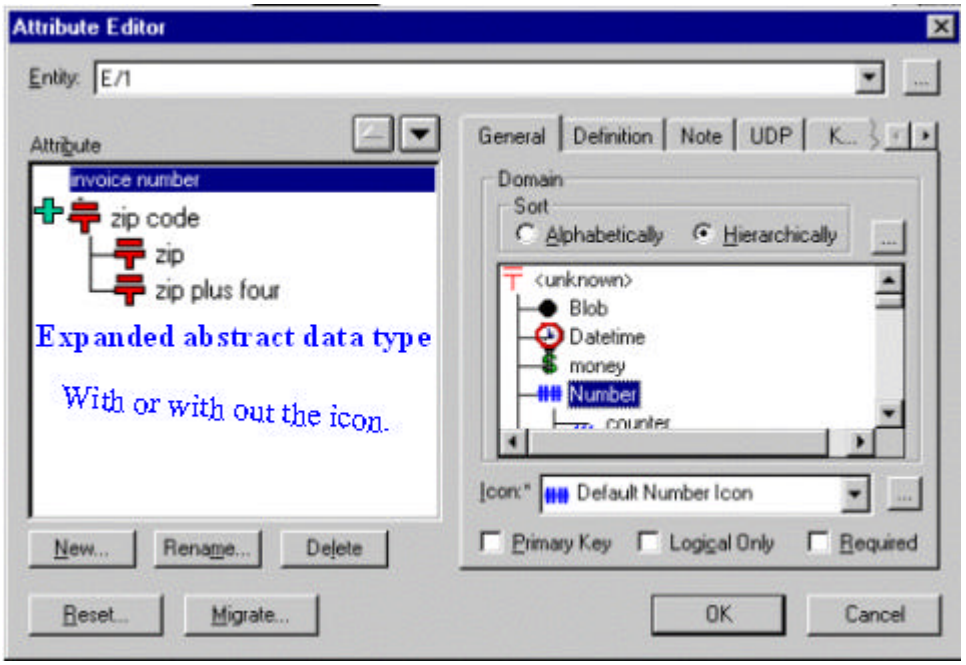


図 14 拡張抽象データ型

エディタの種類は ERwin のドメイン・エディターに現在でも見つけることができる。レベルを作成する能力は属性エディターに存在する。モデルの中でのデータタイプの表現は、コレクションを表わした図 15、16 に多少似ている。例を示すと、collapsed および拡張抽象データ型を示す

ことができる。

図 15 Collapsed 抽象データ型

図 16 拡張抽象データ型

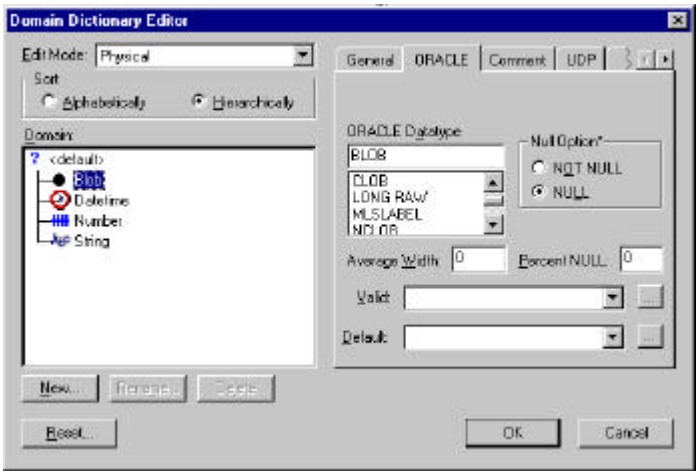


図 17 LOB

大きなオブジェクト・データ・タイプ(LOB)は ERwin にとって未経験のものではない。バージ

ョン 3.5.2 では、様々な LOB に既に備えている。図 17 参照。

ポインターは IT の歴史から借りられた章である。それは「déjà vu（既視感）」に至る所で似ている。チェイン･ファイルやもっと近辺ではネットワーク・データ・ベースの記憶を戻す。現在、ORDBMS の Oracle では、2 種類のポインターがある。REF と IOD である。REF は、他のテーブルあるいはデータ・ベースの外のフラット・ファイルのカラムまで参照できる。例えば、ある種類の Blob の格納が、別のテーブル・スペースまたはフラット・ファイルに置かれる場合、それらが使用される。REF はまれにしか参照が付けられなかった Blob を指す場合があり、非常に大きなマルチメディア・クリップはテーブルまたはフラット・ファイルに格納される場合がある。

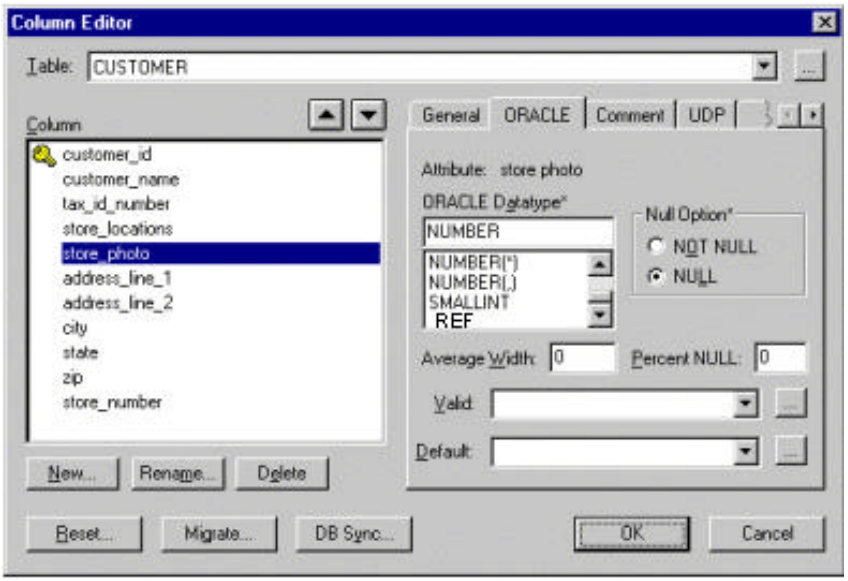


図 18 REF

純粋オブジェクト・データ・ベースでは、OID あるいはオブジェクト ID が、キーに替わって使用される。これらは Number を生成し、一度だけ使用され、再び使われることがないデータベースである。ORDBMS は、従来のキーあるいは OID を使用する選択を提示する。

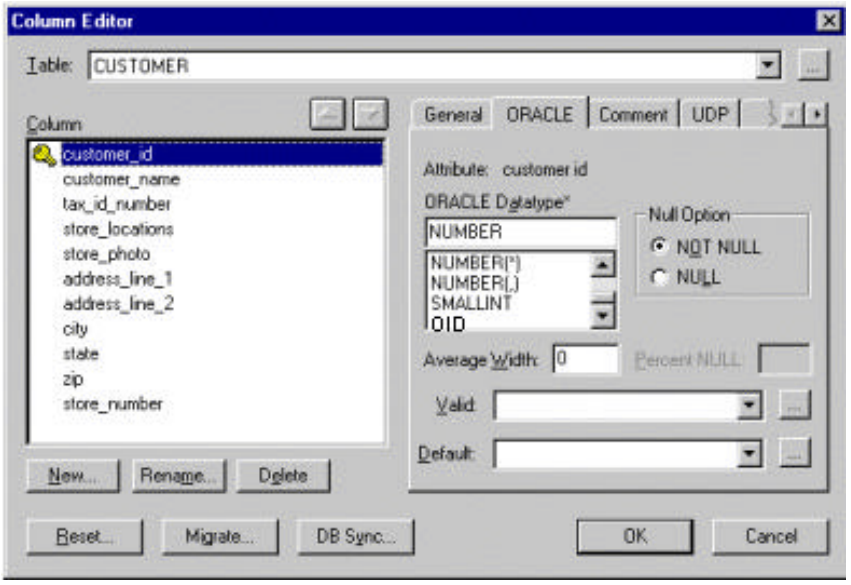


図 19 OID

再び言うが、OID には長所および欠点両方がある。「OID は、多くのアプリケーション、特に分散処理およびレプリケーションにとって重要な、全体的にユニークな識別子を提供する。しかしながら、ストレージ・リクアイアメントおよびローディング・タイムのオーバーヘッドが増加することにも注意が必要である。」[7]

ERwin の将来のリリースに影響するを持つ、オブジェクト･リレーショナルとは関係ない、他のデータベースの特徴がある。例えば、拡張可能インデックスである。この特徴は、インデックスを付ける方法をデータ・ベース・ツール・セットの一部として必須ではなく、データ・ベースを拡張するために使用する。

コンプレックス・データ・タイプはコンプレックス・インデックシング方法をもたらす。テーブルをクラスターに分ける能力は、ユーザによって要求されたように見える別の特徴である。これらの問題は、この記事の範囲外である。

ERwin4.5 以降、IDEF1X と IE はもっと「オブジェクト」を採用するだろうか。CA の UML ツール Paradigm Plus は、「IDEF1X-ity」になるのでしょうか。UML と IDEF1X がマージされ IDEFML または UMDEF1X になるのかな? ERwin は Parawin または Erwindigm になるのかな?

OO へのシフトは確かにある。しかし、IDEF1X および他の ER 記法が残るということは、現状では明らかでない。しかし、現状の範囲では、オブジェクト・メタ･モデルはオブジェクト・リレーショナル・パラダイムの形式をとった伝統的なリレーショナル構造の融合によって構築される。付加的にオブジェクト概念および構築は、オブジェクト・リレーショナル・パラダイムが引き続くことを示している。少なくとも ERwin が最新のデータ・ベース技術と歩調を合わせる予定であるなら、この問題はバージョン 4.5 にアドレスされなければならない。

[7] Getting to Know Oracle 8I (Document #A76962-01), 1999 Oracle Corporation

Vendors Be Normal

Ben Ettlinger
Data Administrator
New York Power Authority
President
New York Enterprise Modeling User Group
訳)株式会社日本総合研究所　細川 努、鈴村 幸太郎

[8]硬い殻に閉じこもっているということは、フラストレーションだ。多くのソフトウエアベンダーが「われわれの製品は十分で、もう他のソフトウエアは必要ないですよ」と考えていると思っているかもしれない。

少なくともベンダーのデータベース構造を見たときにそう思うであろう。たとえどんなソフトウエアが企業の IT インフラをサポートきたとしても、孤島に立っているようなものである。重厚壮大な ERP ソリューションを導入した企業でさえ、ビジネスプロセスにおいて DATAMART や DWH や他のシステムに情報を提供するためにデータを引き出すことは必要である。ベンダーが提供するインターフェイスは、ユーザーにとって不十分で、あなたが求めているものでない事が多い。

データベースに継承されているビジネスルールを完全に理解するために、データの内部構造を見て、どのようにデータが蓄積されているか見ることができるようになりたいだろうか。内部まで見たくないのかもしれないが、すべてのひとは、データベースの中に引き継がれているビジネスルールとデータ要求が、実際のビジネスのデータモデルに対して当てはまるかどうか、興味を持っているはずである。

私はアメリカの中でもっとも大きな公共事業を持つ州のデータ管理者として、ベンダーにソリューションを提供されたような被雇用者に対して、いつも実際に存在するビジネスルールとデータ要求を比較させてくださいと頼んできた。

そのような比較は、データの過不足を決めることを簡単にしてくれる。

ソフトウエアベンダーの意向にかかわらず、あなたのサイトにいったんアプリケーションが構築されれば、データベースをを一瞬にしてリバースすることができる。

どんなデータモデリングツールでも何百ものテーブルをほんの数分で ER 図にリバースすることができるほど有能である。最近私は CA-ERWin を使って Oracle データベースに作られた 400 のテーブルを 5 分かからずにリバースエンジニアリングした。

ベンダーがデータベースの中身を見ることを禁止しなければ、あなたは一瞬でこの図を作ることができ、フラストレーションなんてないだろう。

私は、今までに、ベンダーのソリューションにいくつかの正規性の類似性を発見した。

---

[8] *This article appeared in the Aug. 4, 2000 edition of the DM Magazine Weekly on-line newsletter.*

決して、第4正規形について話しているのではない。第1正規形や第2正規形はどうだろうか？ 多くのベンダーは、Codd は「鱈」かマサチューセッツの住人だと考えているようだ。物理実装では多少の非正規化を（たとえ控えめでも）必要とするということは理解できない。

しかし、これらの非正規化がされていても、少なくともデータベースを正規化して眺めるということは難しくはない。しかしながら、正規化がまったくなされていないのならば、データと関連したルールを理解することは、困難な作業になるだろう。

DBMS の関係制約の実装がいいかげんであるのに出くわすことがある。テーブルの間の"リレーションシップ"の多くは見せ掛けの外部キーという意味を含んで使われている。これらの見せ掛けの外部キーは同じ内容を含んだ二つ以上のテーブルに二つの対応する項目を配置することで発生する。二つのテーブルの間のコネクションは、二つのカラムのデータが一致するという事実のために、実際のデータベースでもキーのマッチングの関係を構築すべきであり、外部キーの鏡像である。

そして、アプリケーションのコードにはこれら両方の同期をとることに責任を持たなければならない。これは、データベースの制約を使わずに、見せ掛けのリレーションシップとして妥協した integrity（完全性）を残してしまう。

他のベンダーでは、テーブルの間のリレーションシップを保つためにデータベースのトリガーに頼っている。これは、少しは完全性の確保をデータベースの責任にすることができる。しかし分析の観点からすれば、トリガーで確保された完全性はデータベースの構造を調べたとしても自明でないという問題がある。

この場合、制約が存在するかどうかを知るためには、トリガーのコードやドキュメント（知ってのとおり、すべてのドキュメントがいつも正しいドキュメントとは限らない。）を調べなければならない。リレーションシップにおいてこんなことが正しいことだろうか？

私はまた、トランザクション駆動型のデータ－ベースを見つけた。これは、論理構造やいくらかの正規化をするというより、システムによって完結するトランザクションを基本として、情報を蓄積している構造を持ったデータベースである。これは、他のアプリケーションで利用するためのデータ分析やデータ抽出を非常に難しくする。また、アプリケーションの機能的な設計を完全に理解することができない。

より簡単になっていくものなど一つもない。ベンダーはオブジェクトリレーショナルパラダイムへと動いて、アレイ（array）や入れ子（nest）構造のテーブルや、メソッドなどを取り入れようとしている。データベースの正規化やデータベースの構造の中にあるビジネスルールを理解する機能がだんだんはっきりしないものになっていく。

DHW の神聖な教祖である Ralph Kimball の最近のコラム[9]に、次のような文章をつづっている。これは、私の考えと同じことである。

ベンダー（この場合、ERP ベンダー）がデータとビジネスプロセスの間の壁の透過を難しくな

[9] Data Webhouse in Intelligent Enterprise June 26, 2000

るように作っているということである。

ここで Kimball がいう壁とは以下の言葉に答えがある。

「ERP システムは意思決定のサポートよりもトランザクションのプロセスを強く意識している・・・・フロントエンドやユーザーの必要性を無視するような密室文化である。」

「ERP システムの基本データベーススキーマはばかばかしいくらい複雑である。複雑に絡んでいて、正規化のルールの考慮のかけらもない。」

Kimball は、こう締めくくっている。シームレストランザクションとオープンシステムを求める声が他からでてきて、ERP の重要性は下がっていくだろう。

何度も振り返ってみるが、オープンプラットフォームがもっとも成功してきたものである。すなわち、Microsoft は、環境は守られたが、複雑さのせいで、自然であるあるいは人工的にかにかかわらず結局は、落ちていくだろう。

そして、ここに極みがある。私たちは共通フレームワークとオープンシステムの時代へと動いていくであろう。データベースの更なる正規化、アプリケーションのドキュメントを含む論理データモデルが良い出発点である。

［監訳者のコメント］

Ben が言いたいことは何か、これはユーザー会のメンバーへだけでなく、国内のさまざまなデータベースにかかわっている関係者へのメッセージでもある。

真摯に受け止めてほしい。

私たち Japan Enterprise Modeling User Group の目標ももちろん、ここにあるわけで、Erwin が単なるモデリングのための道具ではなく、情報システムを正しく構築するためのサポートを行える道具として位置付けたいと考えているのである。

日本語版 第 2 版の出版に当たって

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

第 1 版の日本語バージョンを出版したと思ったら、あっという間に第 2 版が出版され、日本 ERwin ユーザー会のメンバーには、「オイオイまたかよ!!」といわれることは必至であろうと思いながら、会長の権限で押し切ってしまった。

しかし、編集者の覚え書きに自分の名前が出ているのは、なんとなく面映ゆい思いでありながら、結構嬉しがっている自分を発見したりもして…。

今回も、Ben の記事が主体となっているが、とくに最終記事「Vender be Normal」には注目してほしい。データベースの設計が日本よりも明らかに進んでいる米国においても、かくやと思わせる重要な記事である。

われわれモデル屋としても襟を正して読むべきであろう。国内においても、ERP ベンダー、コンピュータ・メーカー、パッケージ・ベンダーには充分心してもらいたい事柄である。

また、なつかしい製品の名前も出てきた「OR-Compass」。この製品はバージョン 1.0.1 で開発が終了したのだが、国内では日揮情報システム（当時）と私のところに試供版が送付され、評価を行っていたものである。Ben の記事にもあるように、この製品は当時 OR を開発し、これで市場を席巻しようとした Informix のためにあったものであったが、Informix の失敗と同時に消え去ってしまった。しかし、これが SQL 3,4 の標準化をベースに実現しようとしている。しかし、リレーショナルのモデルよりも更に難しい OR のモデリングはどのように進めるべきか？ これは、ある意味ではユーザー会の課題であるかもしれない。

今回も、ERwin ユーザー会のメンバーが翻訳に協力してくれた。感謝、感謝である。