ERモデルに基づく図式問い合せ言語とその実現法

日本電気㈱　ソフトウェア生産技術研究所　秋口忠三
Chuzo Akiguchi

あらまし
データベースに対する問い合せ処理は計算機の非専門家が行なう情報処理業務としてもっとも重要なものの一つである。図式問い合せ言語はこのような計算機利用者にとって非常に有用であると思われる。本論文ではＥＲモデルに基づく図式問い合せ言語とその実現法について述べる。問い合せ要求はグラフィックス・エディタを用いてＥＲダイアグラム上に記述される。この問い合せ記述は内部的にはまず関数モデルに変換され、最終的には関係代数演算式が生成される。この関係代数演算式が実行されて求める結果が得られる。

１．はじめに

データベースに対する問い合せ処理は計算機の非専門家（エンド・ユーザ）が行なう情報処理業務としてもっとも重要なものの一つである。

エンド・ユーザ用の問い合せ言語としてはリレーショナル・モデルに基づくＳＱＬ[1]やQuery by Example（ＱＢＥ）[2]などが標準言語として認知されている。ＳＱＬはテキスト・ベースの問い合せ語であり、問い合せ要求を出す対象となるデータベース・スキーマの特定部分を名前によって参照する必要がある。インタラクティビティはそれほど高くない。

一方ＱＢＥは、データベース・スキーマ（リレーションの枠組み）を画面上に表示し、これにたいして参照したい部分を直接指示できるようにした、一種の図式問い合せ言語である。インタラクティビティを最大限に活用したこの問い合せ要求の表現方法により、データベースの利用者は問い合せ言語の構文およびデータベースの構造を容易に理解できるようになった。このことがＱＢＥをエンド・ユーザ言語として広く普及させた理由の一つであったと考えられる。

しかしＱＢＥを用いても、リレーショナル・モデルが本来持っている意味論的な欠点を完全にカバーすることはできないと思われる。すなわちリレーショナル・モデルではデータベースを第三正規形のリレーションの集まりとして実現することを基本としているために、多くの実体の連関で構成される事実は一般に複数のリレーションで表現される。従ってこのような事実に関する問い合せでは、一般に複数のリレーションを結合する必要がある。結合すべきリレーションのキーが複数属性から構成されている場合にはこの結合の指定は必ずしも容易とはいえない。

ＥＲモデルが提案された理由の一つはリレーショナル・モデルのこのような欠点を補うためであった[3]。本論文ではＥＲモデルに基づく図式問い合せ言語とその実現法について述べる。ここで提案する図式問い合せ言語は、ＥＲモデルを用いることにより、ＱＢＥの利用容易性を保存しつつ、リレーショナル・モデルに基づく問い合せ言語の欠点を補うことを目的とする。問い合せ要求はグラフィックス・エディタを用いてＥＲダイアグラム上に記述される。この問い合せ記述は内部的にはまず関数モデル[4,5,6]に変換され、最終的には関係代数演算式が生成される。この関係代数演算式が実行されて求める結果が得られる。

エンド・ユーザの立場から見た図式問い合せ言語の利点は次の二点にまとめることができよう。

- データベースの構造が図式で表現されるので理解しやすく、問い合せの構文も単純にできる。
- データベースの構造図式上で問い合せ要求を記述できるのでタイピング量が少なくてすむ。

このような特長を持つ図式問い合せ言語の実現の試みは以前よりあった[7,8,9]。最近のハードウェアの進歩で、ユーザ・フレンドリィなグラフィック・インタフェースを安価に提供できるようになったことにより、データ間の関連までも図式で扱える図式問い合せ言語の有用性は高まるであろう。

２．問い合せ処理の概要

本問い合せ処理システムは、ワークステーションのＵＮＩＸ上で開発中である。データベース管理システムとしてはリレーショナル方式のTroll/USE[10]，グラフィックス・エディタはＩＤＥのＥＲエディタ[11]をベースにしている。

２．１　問い合せ処理環境

問い合せ処理の実行時の環境はＵＮＩＸのファイル・システムの階層的ディレクトリ構造によって図１のように実現される。

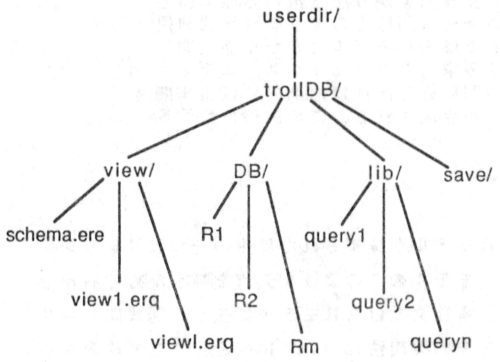

Fig.1　Query Environment

図１　問い合せ処理環境

図１においてtrollDB/は問い合せ環境のルート・ディレクトリである。trollDB/の下には４つのディレクトリ－－view/ , DB/ , lib/, save/ －－が置かれる。view/ ディレクトリの下には、グラフィックス・エディタを用いて作図したＥＲダイアグラムをテキスト形式で表現したファイルが置かれる。データベース・スキーマに対応するＥＲダイアグラム（以後スキーマ・ダイアグラムと呼ぶ）はただひとつ存在し、このファイルにはschema.ereという固定名が付けられる。またスキーマ・ダイアグラムの特定部分を選択することによって作られたビューに対応するＥＲダイアグラム（以後ビュー・ダイアグラムと呼ぶ）は複数個存在し、これらのファイルには.erqのサフィックスが付けられる。

DB/ ディレクトリの下には、Troll/USE のデータベースを構成するリレーションが格納される。これらのリレーションはスキーマ・トランスレータによってスキーマ・ダイアグラムから生成された Troll/USEデータ記述文（リレーショナル・スキーマ）をTroll/USE に入力することによって生成される[6] 。

データベースに対する問い合せ要求はビュー・ダイアグラム上で記述され、それをクエリィ・プロセッサが処理してlib/ディレクトリの下にTroll/USE の問い合せ記述文を生成する。これを Troll/USEが実行することにより所望の結果を得るわけである。問い合せの結果は save/ディレクトリの下に保存することができる。

クエリィ・プロセッサはウィンドウ・システムの下で以下のコマンドを実行することによって起動される。

```
erq [-e query_env] [view_diagram]
```

ここで query_envは問い合せ環境のルート・ディレクトリ、view_diagramは問い合せを実行したいビュー・ダイアグラム名である。

２．２　問い合せ処理時の画面イメージ

クエリィ・プロセッサ起動後の画面イメージを図２に示す。問い合せ処理は、この図に示すようにワークステーションのディスプレイ上に表示された３つのウィンドウを用いて行なわれる。それぞれのウィンドウは次の役割を持つ。

(1) 問い合せウィンドウ

このウィンドウ上ではＥＲダイアグラム用のグラフィックス・エディタが動いている。グラフィックス・エディタの編集機能を使ってビュー・ダイアグラムを定義する。

(2) 問い合せ条件記述ウィンドウ

このウィンドウ上ではＵＮＩＸのｖｉエディタが動いている。属性（円で表される）に対して問い合せ条件や導出属性の定義などを入力するために用いる。

(3) 問い合せ結果表示ウィンドウ

これはクエリィ・プロセッサを起動したプロセスに対応するウィンドウである。このウィンドウに問い合せの結果が表形式で表示される。

２．３　問い合せ要求記述の概要

本図式問い合せ言語では、問い合せ要求を、ＥＲモデルに基づく図式表現とテキスト表現の組み合わせによって、次の２段階で記述する。

(1) ビュー・ダイアグラムを定義

スキーマ・ダイアグラムから問い合せ要求者の興味の対象となる部分を選択・編集する。この段階で導出属性の定義を行なうことができる。導出属性とは他の属性から四則演算や集計演算などによって算出される属性である（３．４節参照）。このようにして定義されたビュー・ダイアグラムは問い合せ処理環境の view/ディレクトリの下に保

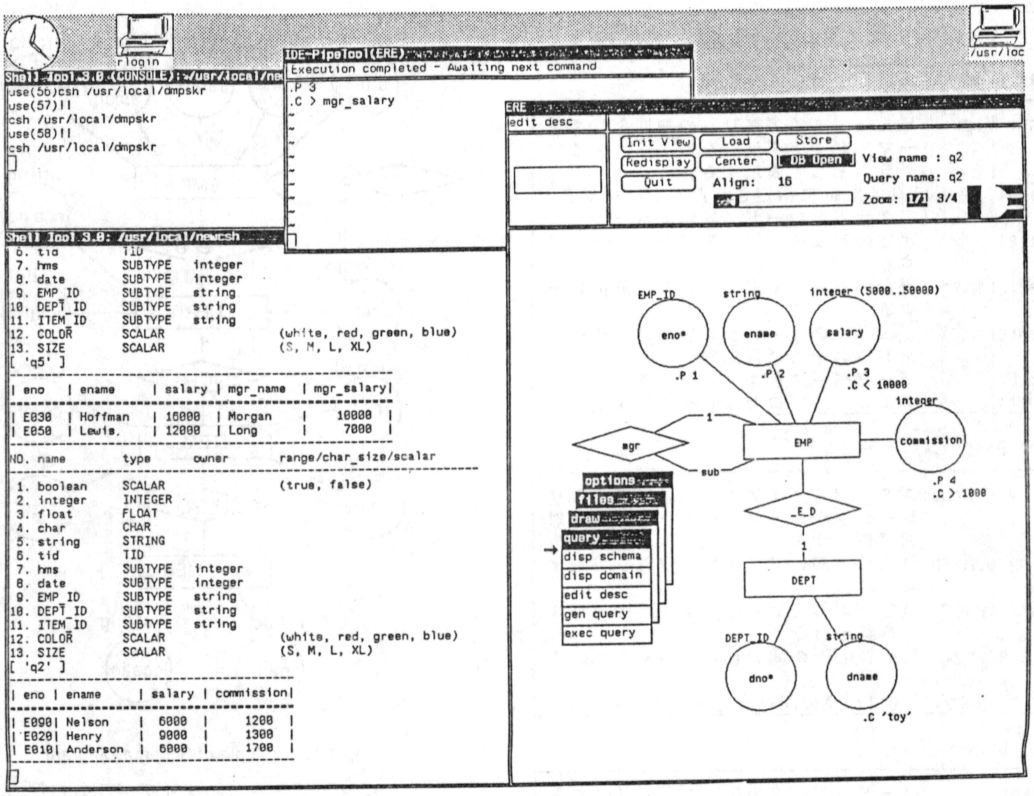

Fig.2 A screen image in query processing

図２　問い合せ処理時の画面イメージ

存し、クエリィ・プロセッサ起動時または問い合せ処理実行中に指定することができる。クエリィプロセッサの起動時にビュー・ダイアグラムの指定がない場合はスキーマ・ダイアグラムがビューとして採用される。

(2) ビュー・ダイアグラム上で問い合せ条件を記述
ビュー・ダイアグラムの属性に対して問い合せ要求を記述する。問い合せ要求の記述は、マウスを用いて属性を指定した後、問い合せ条件記述ウィンドウ上のｖｉエディタにより行なう。問い合せ要求記述については３節で述べる。

２．４　システム・メニュー

問い合せ要求は問い合せウィンドウ上部の固定メニューおよびポップアップ・メニューを選択することによって行なわれる。各々のメニューの項目およびその意味を表１にまとめる。

固定メニューは問い合せの画面環境を整えるためのコマンドから，Ｑｕｅｒｙメニューは問い合せ要求の記述と実行を行なうためのコマンドから成る。Ｄｒａｗメニューはビュー・ダイアグラムの編集モードを変更するために用いる。insert,replace,move,delete,scaleの５つのモードがあり、各モードの下で対応する編集操作を連続して行なうことができる。

３．問い合せ記述

２．３節で、問い合せ要求の記述が、ビュー・ダイアグラムの定義とビュー・ダイアグラム上での問い合せ条件の記述の２段階で行なわれることを示した。本節では、図３のスキーマ・ダイアグラムを用いて、ビュー・ダイアグラムの図式定義の方法と、この要求記述のテキスト表現に当たる部分、すなわち導出属性の定義と問い合せ条件の記述、について説明する。

表1　システム・メニュー一覧

(a) 固定メニュー

Init View:	ビュー・ダイアグラムの初期化。スキーマ・ダイアグラムをビュー・ダイアグラムとして表示する。
Load:	View Nameに指定されたビュー・ダイアグラムをロード・表示する。
Save:	現在のビュー・ダイアグラムをセーブする。
Redisplay:	現在のビュー・ダイアグラムを再表示する。
Center:	ビュー・ダイアグラムを問い合せウィンドウの中央に再配置する。
DB Open:	データベースをオープンする。
Quit:	クエリィ・プロセッサを終了する。

(b) Query メニュー

disp schema:	スキーマ・ダイアグラムに対応するリレーショナル・スキーマをテキスト形式で表示する。
disp domain:	属性の上部にドメインの定義を表示する。
edit desc:	属性に対して問い合せ条件式のテキスト編集を行なう。
gen query:	Troll/USE の問い合せ記述文を生成する。
exec query:	Troll/USEの問い合せ処理を実行する。

(c) Drawメニュー

insert:	ノード／アークの追加のモードにする。
replace:	ノードの変更モードにする。
move:	ノード／アークの移動モードにする。
delete:	ノード／アークの削除モードにする。
scale:	ノード・サイズの変更モードにする。
clear:	問い合せ条件式をすべてクリアする。

導出属性の定義と問い合せ条件の記述は、Ｑｕｅｒｙメニューからedit desc（問い合せ条件のテキスト編集）を選択することによって起動されるｖｉエディタを用いて行なわれる。問い合せ条件はプリント仕様と選択条件の記述とから成る。プリント仕様，選択条件の記述および導出属性の定義の始まりは、それぞれ。.P，.Cおよび.Dで示される。

３．１　ビュー・ダイアグラムの定義

ビュー・ダイアグラムはスキーマ・ダイアグラムに対してノードの削除、追加、および移動のコマンドを実行することによって作られる。スキーマ・ダイアグラムからビュー・ダイアグラムを定義する手順を以下に示す。

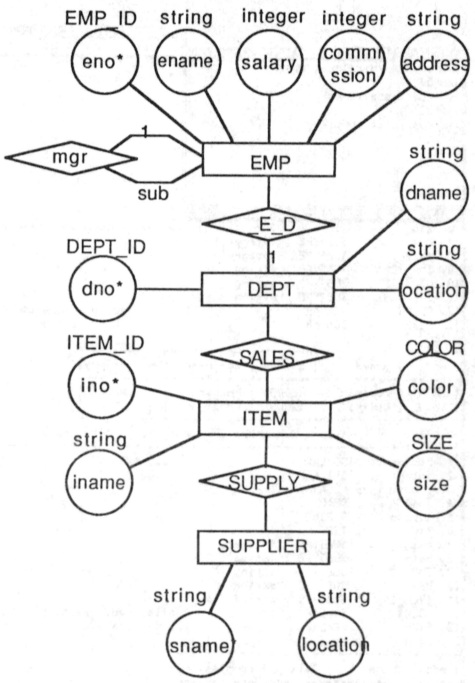

Fig.3 An Example of Schema Diagram

図３　スキーマ・ダイアグラムの例

(1) 固定メニューの Init Viewをマウスでクリックする。これによりスキーマ・ダイアグラムが問い合せウィンドウに表示される。

(2) 編集モードをdeleteにし、必要のない部分をマウスでクリックすることにより削除する。

(3) 編集モードをinsertにし、導出属性のノードを追加する。

(4) 編集モードをmoveにし、ノードをマウスでドラッギングすることにより全体をバランス良く再配置する。

３．２　プリント仕様

問い合せの結果は表形式で表示される。プリント仕様では、その表に含めるべき属性に対して、表の中でのカラム順序、カラム幅、および見出しを与える。

プリント仕様の構文は以下のとうりである。

.P [order] [:width] [header]

ここで、order はカラム順序を表す整数、width はカラム幅を表す整数、headerは見出しを表す文字列であり、それぞれ省略できる。.Pの指定は関係代数の射影に相当する。

３．３　選択条件の記述

ＥＲモデルではデータベースを実体の集合とそれらを結び付ける関連の集合として表現する。選択条件は、これら実体の集合と関連の集合から検索すべき部分集合を選択するための条件式であり、実体集合と関連集合の属性に対する論理式の形で記述する。選択条件はビュー・ダイアグラムの各々の属性とビュー・ダイアグラム全体に対して指定でき、指定された選択条件をすべて満足する実体と関連の集合がデータベースから選択される。すなわち問い合せの選択条件はすべての選択条件式の and結合と解釈される。

各属性に対して与えられる選択条件式は要素的命題を and（＆）とor（｜）、not （～）で連結したものである。要素的命題は、対象とする属性の値と他の属性の値または定数値との比較を取ったもので、その構文は次のとうりである。

```
rel_op ( attr_name ¦ constant )
```

ここで関係演算子（rel_op）としては、＝，～＝，＞，＜，＞＝，＜＝，＄，および！＄がある。関係演算子＄と！＄は文字列型の属性に対して文字列の包含関係を与える。たとえば

```
$ "abc"
```

は対象とする属性の値が文字列abc を含むならば真

```
!$ "abc"
```

は対象とする属性の値が文字列abc に含まれるならば真である。

属性値の範囲の指定を容易にするために次の略記法を用意している。

```
(x,y)  ... >x  & <y
(x,y]  ... >x  & <=y
[x,y)  ... >=x & <y
[x,y]  ... >=x & <=y
~(x,y) ... <=x ¦ >=y
~(x,y] ... <=x ¦ >y
~[x,y) ... <x  ¦ >=y
~[x,y] ... <x  ¦ >y
```

問い合せに対する選択条件は全選択条件式の and結合と解釈されるので、属性に対する選択条件だけでは、例えば図３のビュー・ダイアグラムに対して次の選択条件を表現することはできない。

（toy 部門で働いているかまたは給与が200000円以下）

```
dname = 'toy' ¦ salary <= 200000
```

このように異なる属性に対する選択条件のor結合を必要とする場合には、ビュー・ダイアグラムに対する選択条件として指定する。その要素的命題の構文は次のとうりである。

```
attr_name rel_op ( attr_name ¦ constant )
```

属性に対する選択条件では、要素的命題の関係演算子の左辺を省略できるが、ビュー・ダイアグラムに対する選択条件では省略できない。

ビュー・ダイアグラムに対する選択条件はＱＢＥのコンディション・ボックスに相当する。

３．４　導出属性の定義

ＥＲモデルでは、実体や関連の性質を表すために属性が用いられる。情報を冗長なく蓄積・管理することがデータベースの目的であるから、スキーマ・ダイアグラムには一般に冗長な属性は含まれない。ここで冗長というのは、その属性値が他の属性値から導き出しうるということを意味する。一方特定の問い合せ要求を持つ利用者にとっては、要求する情報が、たとえ冗長であっても、より明示的かつ直接的に与えられるほうがよい。すなわち各々の利用者の要求にあったデータのビューを定義できることが望ましい。

リレーショナル・モデルでは、リレーションに対する問い合わせの結果はまたリレーションであるから、これを仮想的なリレーションとして利用者に提供することで、利用者の要求にあったデータのビューを実現できる。

ＥＲモデルに基づく本問い合せ言語では、ビューダイアグラムの定義時に実体集合や関連集合に対して導出属性を定義する機能を組み込むことにより、この要請に対応しようと考えた。導出属性とは、あるアクセス・パスを辿って参照される属性に対して、算術演算(+,-,*,/,div,mod)や文字列演算(++)、集計演算(account,avg,min,max,sum,sdev) などを適用することにより算出される属性である。

導出属性の定義は次のように行なう。まずビュー・ダイアグラムに新しい属性ノード（円形）を追加し属性名を付ける。これを対応する実体集合あるいは

関連集合に接続する。最後にその導出属性に対して導出仕様を記述する。

導出仕様の記述において、「アクセス・パスを辿る」という操作は関数モデルの枠組みの中でより明確に示すことができる。ここではいくつかの例を示すにとどめ、4．1節で説明を補足する。以下の例は、図3の実体集合 EMP（従業員）に対して定義された導出属性の導出仕様である。

（例1）所得総額

```
.D salary + commission
```

（例2）上司の名前

```
.D mgr.ename
```

（例3）部下の平均給与

```
.D !mgr.salary.avg
```

（例4）所属部門(DEPT) で売っている(SALES) 品物(ITEM)の種類数

```
.D _E_D.SALES.ITEM.account
```

3．5　問い合せ記述の例

図2に問い合せ処理時の画面イメージを示した。この画面に表示された問い合せの意味は次のとうりである。

（問1）toy部門で働いていて、かつ給与(salary)が$10000以下で、かつ歩合(commission) が$1000以上の従業員(EMP) について、従業員番号(eno) 、名前(ename) 、給与、および歩合を表示せよ。

この問い合せは、図2のビュー・ダイアグラムに対してでなく、スキーマ・ダイアグラム（図3）上で記述することもできる。

より複雑な問い合せ条件を持つ問い合せ例、以下の問2と問3、の図式表現をそれぞれ図4と図5に示す。図5の例では導出属性が使われている。

（問2）pencilかcomputerを売っている toy部門またはhardware部門で働いていて、かつ給与が$6000以上$20000以下で、かつ名前が 'L' で始まるか 'n'で終わる従業員について、従業員番号、名前、給与、および所属部門名を表示せよ。

（問3）自分の上司より多い給与を取っている従業員について、従業員番号、給与、上司の名前(mgr_name)、上司の給与(mgr_salary)を表示せよ。

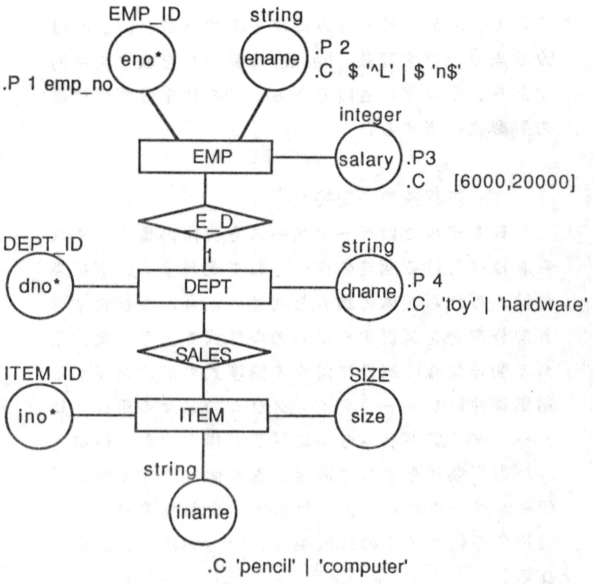

Fig.4 Graphical Query Representation of Q2

図4　問2の図式表現

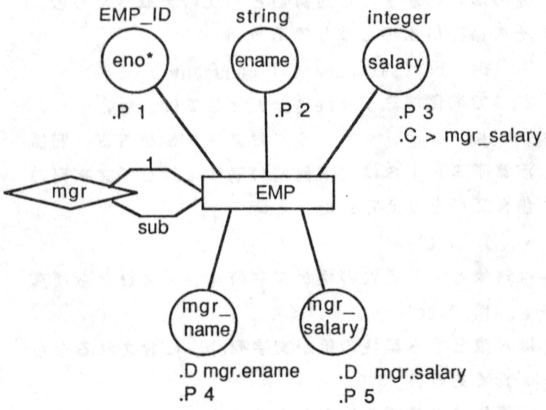

Fig.5 Graphical Representation of Q3

図5　問3の図式表現

これらの問い合せはいずれも概念的には次のように処理される。まず導出属性が定義されていたらその値が算出される。次に問い合せの中で参照されるすべての実体と関連の組み合わせが作られる。これらの各々の組に対して選択条件を満たさないものは削除される。最後にプリント仕様を持つ属性の値が取り出され、指定された順序に従って表示される。

4．問い合せの内部表現

問い合せ要求はビュー・ダイアグラム上で記述される。これが関係代数演算式に変換される過程で、内部的にいったん関数モデルでの表現に変換される。この変換方法はリレーショナル・スキーマの生成方式と同じである[6]。

ビュー・ダイアグラムの各要素（実体、関連、属性）はスキーマ・ダイアグラムの対応する要素に関連づけられなければならないが、この関連づけも関数モデルの上で行なわれる（図6参照）。

4．1　関数データモデル

データベースの関数モデルは、データベースを対象集合と対象集合の間の関数関係（1対1または多対1関係）によって表現するデータ・モデルである。対象は（それが抽象的な存在か具体的な存在かによって）抽象的対象と具体的対象とに分類される。抽象的対象はその存在がデータベースの更新によらずスキーマ定義時に決定される。一方具体的対象の存在はデータベースの更新処理によって出現したり消滅したりする。例えば、従業員番号や年齢、色などは抽象的対象として、従業員や部門は具体的対象としてモデル化される。

関数は二つの対象集合の間で定義される。対象集合Ｏｓから対象集合Ｏｄへの関数Ｆは次のように表される。

Ｆ：Ｏｓ　－＞　Ｏｄ

ＥＲモデルにおける概念が以上の関数モデルでの概念とどのように対応付けられるかを表２に示す。図７は図３のスキーマ・ダイアグラムに対応する関数モデル表現である。

表２　ＥＲモデルと関数モデルの間の対応関係

ＥＲモデル		関数モデル	図３で対応する要素
実体集合		具体的対象	EMP,DEPT,ITEM, ...
関連集合	多項 多対多	具体的対象	SALES,SUPPLY
	多対１ １対１	関数	mgr,_E_D
多項／多対多関連の役割		関数	(SALES/DEPT), (SALES/ITEM), ...
属性		関数	eno,ename,salary,...
定義域		抽象的対象	EMP_ID,string, ...

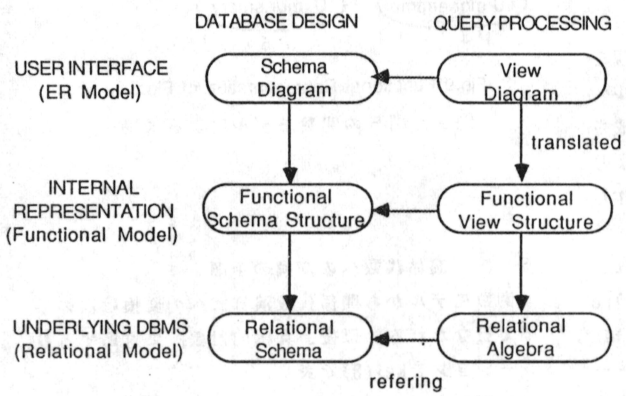

Fig.6 Query processing mechanism

図６　問い合せ処理の方式

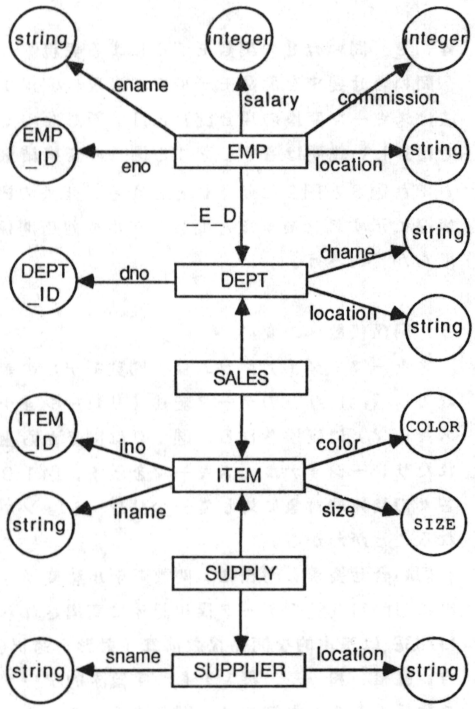

Fig.7 Functional Representation of Fig.3

図７　図３の関数モデルによる表現

関数は二つの対象集合の間の多対１の関係を表すと同時に、アクセス・パスを辿るために使われる。３．４節で導出属性の定義の例をいくつか与えた。この中で、他の属性を参照するために関連集合名(mgr,_E_D)、属性名(ename,salary,commission) 、実体集合名(ITEM) 等が使われている。これらは、実は関数モデルに翻訳したときの関数の参照に相当するものである。

例３の!mgrはいわば mgrの逆関数で各従業員に対してその部下（複数人）を対応付ける。従って !mgr.salaryは部下の給与の集合であり、集計演算 avgによってその平均が求められるわけである。

例４では、関数モデルで対象集合に相当する実体集合(ITEM) や多対多関連集合(SALES) が関数を参照するために使われているが、この理由は対応する役割の名前が省略されているからである。関連集合SALES における実体DEPTの役割と実体ITEMの役割にそれぞれSUBJとOBJ という役割名がついていたとしたら例４の記述は次のようになる。

```
.D _E_D.!SUBJ.OBJ
```

４．２　問い合せの関数モデルによる表現

問い合せ要求を関数モデルでの表現へ変換する方法はスキーマ変換の場合[6] とほぼ同じなのでここでは詳しい説明は省き、図４と図５の変換結果をそれぞれ図８と図９に示すにとどめる。変換の概要は表２に示すＥＲモデルと関数モデルの対応関係より明らかであろう。

５．関係代数への変換

スキーマ・ダイアグラムは、関数モデルの表現を経て、 Troll/USEのデータ記述（リレーショナル・スキーマ）に変換される。図１０に図７から生成されたリレーショナル・スキーマを示す。図１０より、各々の具体的対象に対して一つリレーションが作られることがわかる。

問い合せ要求も同様に、関数モデルに変換され、次に Troll/USEのデータ操作記述に変換される Troll/USE は基本的な関係代数演算（射影、選択、結合、直積、和、差、積）をもつ手続き的データベース言語である。本節では、図８や図９のように表現された問い合せ要求を、図１０のようなリレーショナル・スキーマに対する一連の関係代数式に変換する方法について述べる。

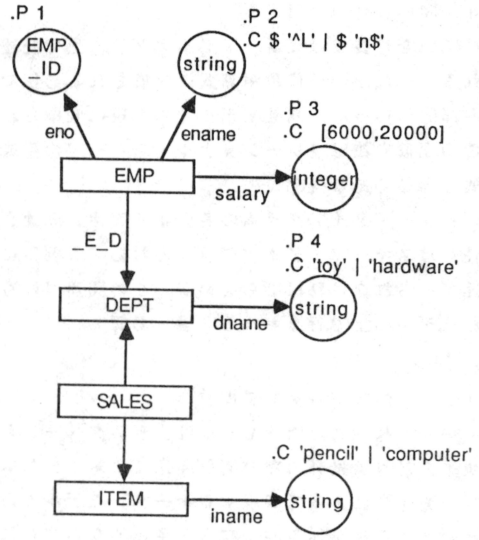

Fig.8 Functional Representation of Fig.4

図８　図４の関数モデルによる表現

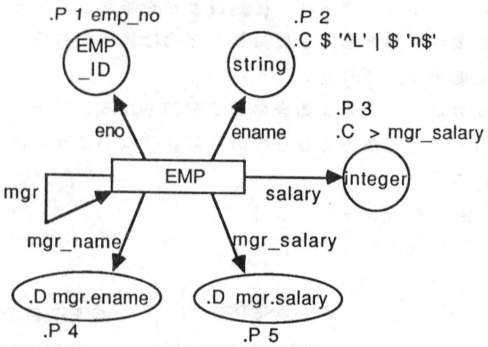

Fig.9 Functional Representation of Fig.5

図９　図５の関数モデルによる表現

５．１　関係代数への変換の手順

関数モデルから関係代数演算式への変換は次の手順で行なわれる。以後、具体的対象B に対応するリレーションをRel(B)で表す。

(1) 問い合せの対象となる具体的対象を求める。プリント仕様か選択条件の少なくとも一方が定義されている抽象的対象への関数をもつ具体的対象が問い合せの対象となる。

```
{Company DB}
{Domain Definition}
{-------------------------------------------------------}
domain  EMP ID:          string;
domain  DEPT ID:         string;
domain  ITEM ID:         string;
domain  COLOR:           scalar(white, red, green, blue);
domain  SIZE:            scalar(S, M, L, XL);

{ Entity Relation }
{-------------------------------------------------------}
relation  EMP          [key  eno] of
    eno                  : EMP ID;
    ename                : string;
    salary               : integer (5000..50000);
    commission           : integer;
    address              : string;
    dno                  : DEPT ID;   { from DEPT }
    mgr_eno              : EMP_ID;    { from EMP }
end;
{-------------------------------------------------------}
relation  DEPT         [key  dno] of
    dno                  : DEPT ID;
    dname                : string;
    location             : string;
end;
{-------------------------------------------------------}
relation  ITEM         [key  ino] of
    ino                  : ITEM ID;
    iname                : string;
    color                : COLOR;
    size                 : SIZE;
end;
{-------------------------------------------------------}
relation  SUPPLIER     [key  sname] of
    sname                : string;
    location             : string;
end;

{ Relationship Relation for M M or n ary (n>2) relationship }
{-------------------------------------------------------}
relation  SALES        [key  dno, ino] of
    dno                  : DEPT ID;   { from DEPT }
    ino                  : ITEM_ID;   { from ITEM }
end;
{-------------------------------------------------------}
relation  SUPPLY       [key  sname, ino] of
    sname                : string;    { from SUPPLIER }
    ino                  : ITEM_ID;   { from ITEM }
end;
```

Fig.10 Relational Schema corresponding to fig.7.

図１０　図７に対応するリレーショナル・スキーマ

(2) ビュー・ダイアグラム上でこれらの具体的対象を関連づけている関数を求める。こうして求められた具体的対象と関数とによってアクセス・パスが作られる。アクセス・パスは閉路を作っていてはならない。つまり閉路を作らないようにビュー・ダイアグラムを定義しなければならない。閉路がある場合はエラー・メッセージが出される。

(3) 導出属性が定義されている具体的対象B に対しては、次のようにその導出属性を含むリレーションを生成する。

(a) 導出属性の定義が演算を含まない場合は、参照している関数に基づいて結合演算を行ない、その結果をRel(B)とする。

(b) 導出属性の定義が演算を含む場合は、その導出属性を含むリレーション Rを定義し、Rel (B) の各タプル tに対して導出属性値 vを求め、その結果を Rに追加する。以後 RをRel (B)とする。

ここまでで、アクセス・パスに含まれるリレーションの集合が求められた。

(4) 選択条件をもつ具体的対象 Bに対応するリレーションRel(B)に対して選択演算を適用する。

(5) アクセス・パス上の具体的対象B1から具体的対象B2への関数 Fに対して、Rel(B1)とRel(B2)をRel(B1)の Fに対応する外来キーとRel(B2)のキーで結合する。

(6) ビュー・ダイアグラムに対する選択条件に基づき、得られたリレーションに選択演算を適用する。

(7) 最後にプリント仕様に基づき射影演算を適用する。

5．2　関係代数への変換の例

以上の変換手順を図８と図９に適用した結果を、それぞれ図１１と図１２に示す。

図の左側の矩形と実線の矢印はアクセス・パスを表す。点線の矢印は関係演算子へのリレーションの入出力を表す。

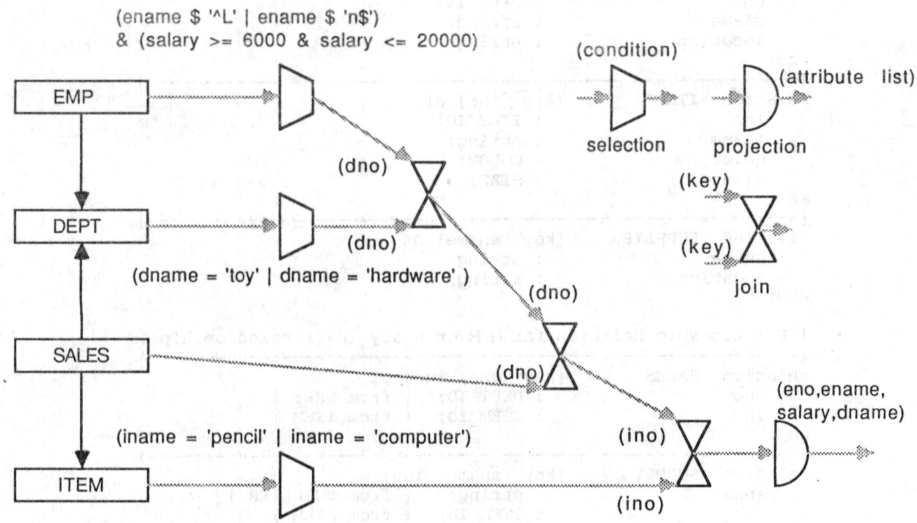

Fig.11 Developing a relational algebra expression from Fig.8

図１１　図８の関係代数式への展開

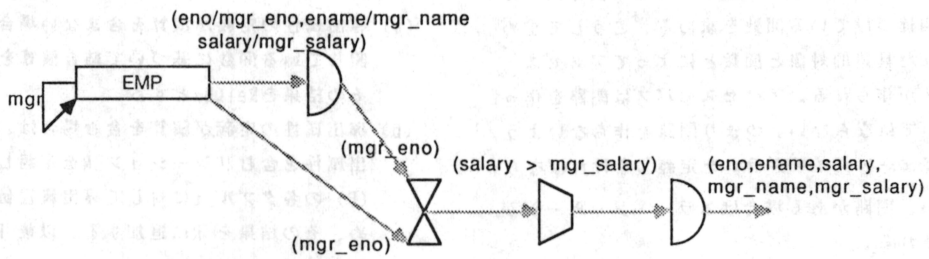

Fig.12 Developing a relational algebra expression from Fig.9

図１２　図９の関係代数式への展開

6．むすび

本論文では、ＥＲモデルを用いた図式問い合せ言語とその実現法について述べた。問い合せ要求をＥＲダイアグラム上で表現し、それを内部的に関数モデルに変換してから関係代数演算式を生成するという方式を提案した。複雑に絡み合ったデータに対して検索処理を行なう手段として、ＥＲモデルを用いた図式問い合せインタフェースは有用であろう。

内部処理用として用いた関数モデルは、リレーショナル・モデルだけでなくＣＯＤＡＳＹＬのネットワーク・モデルや多くの意味論的データ・モデルとの相性もよく、異種データベースを関連づける中間表現として適していると思われる。従って、この方式を用いてＣＯＤＡＳＹＬデータベース上でＥＲモデル・インタフェースを実現することもそれほど困難はないだろう。

問い合せ言語の記述能力については問題を残している。導出属性を定義できるものの、関係代数の割り算、関係論理の全称限定子(ALL)、ＳＱＬのGROUP BY等に対応する機能を持っていない。全称限定子か手続き性の導入を考える必要がある。

エンド・ユーザを対象とする場合、ユーザ・インタフェースの良否が特に重要になる。現在テキスト入力の部分は汎用テキスト・エディタviを用いているが、プリント仕様、選択条件、導出属性の各々に対して、その目的に応じた入力手段を提供するほうがよい。

これらは今後の課題である。

謝辞　本論文は、筆者がカリフォルニア大学サンフランシスコ校に留学中に行なった研究の一部をまとめたものである。留学中に指導を頂いたWasserman教授に深謝する。

参考文献

[1] Chamberlin, D.D, et al., "SEQUEL 2: A Unified Approach to Data Definition, Modification, and Control," *IBM J. Res. Dev.*, Vol.20, No.6, pp.560-575, 1976.

[2] Zloof, M.M., "Query-by-Example: A Data Base Language," *IBM System Journal*, Vol.16, No.4, pp.324-343, 1977.

[3] Chen, P.P., "The Entity-Relationship Model -- Toward a Unified View of Data," *ACM Transaction on Database Systems*, Vol.1, No.1, pp.9-36, 1976.

[4] Shipman, D.W., "The Functional Data Model and the Data Language DAPLEX," *ACM Transaction on Database Systems*, Vol.6, No.1, pp.140-173, 1981.

[5] Buneman, P. and R. Nikhil, "The Functional Data Model and its Uses for Interaction with Database," in *On Conceptual Modelling*, M.L.Brodie, J. Mylopoulos, and J.W. Schmidt (ed.), pp.359-408, Springer-Verlag, New York, 1984.

[6] Akiguchi, C., B. Khan, and A.I. Wasserman, "Optimum Relational Schema Generation From Extended Entity-Relationship Models," *to be appeared.*

[7] Zhang, Z.Q. and A.O. Mendelzon, "A Graphical Query Language for Entity-Relationship Databases," *Proceedings of the Third International Conference on Entity-Relationship Approach*, Anahaim, California, Oct. 5-7, 1983, North-Holland, pp.441-448, 1983.

[8] Ursprung, P. and C.A. Zehnder, "HIQUEL: An Interactive Query Language to Define and Use Hierarchies," ibid., pp.299-314.

[9] McDonald, N. and M. Stonebraker, "CUPID - The Friendly Query Language," *Proceedings of ACM Pacific 75 Conference*, 1985.

[10] Kersten, M.L., A.I. Wasserman, and R.P. van de Riet, "Troll/USE Reference Manual," Laboratory of Medical Information Science, University of California, San Francisco, 1982.

[11] Wasserman, A.I., P.A. Pircher, C.C. Mills, and R. Wagner, "An Integrated Family of Graphical Software Design Tools," *Proc. 2nd Symposium on Practical Software Development Environments*, December, 1986, in press.