磁気バブルデータベース計算機EDCの言語インタフェース

EDCL: RELATIONAL QUERY LANGUAGES FOR THE ETL DATABASE COMPUTER (EDC)

弓場敏嗣　古川康一　植村俊亮
TOSHITSUGU YUBA　KOICHI FURUKAWA　SYUNSUKE UEMURA
電子技術総合研究所
ELECTOROTECHNICAL LABORATORY

1 はじめに

電総研で開発中の磁気バブル記憶を用いたデータベース計算機（EDC；ETL Database Computer）の関係モデル質向言語について報告する。EDCの開発目標の一つは並列に配置した磁気バブル記憶装置を用いての関係モデル質向言語の高速処理であり，EDC用の言語としてそうした関係モデル質向言語を設計，製作する必要がある。

データベース用の関係モデル質向言語としては既に幾つかの提案がある[1]。なかでも著名なのはSEQUEL[2][3]，SQUARE[4]とQBE[5]でありいずれもIBM社で提案され開発されたあるいは開発中の言語である。これらはユーザ・インタフェース言語としての立場からのみ配慮がなされ，実際の使用にあたっての心理学的実験結果なども報告されている[6]。計算機アーキテクチャ即ちデータベース計算機の開発にともなうユーザ・インタフェース言語としての提案はフロリダ大学で開発中のデータベース計算機CASSM用の高水準言語CASDALがある[7]。トロント大学のRAP[8]，ユタ大学のRARES[9]等ではデータベース計算機の高水準機械語の提案はあるがユーザ・インタフェース言語は未だ提案されていない。

EDCLは5層の階層構造を持つEDCの言語系である。本稿ではそれらの階層の内，関係指向と組指向の関係モデル・インタフェースの2つのレベルについてその言語仕様を提案する。また種々の質向文の類型とその記述例を示しSEQUEL，SQUAREなどとの比較検討を行なう。

2 EDCの言語系

2．1 EDCLの階層構造

EDCLはユーザ・インタフェース（F），関係指向と組指向の関係モデル・インタフェース（EとD），高水準データベース命令（機械語）の多重プロセッサ水準（C），同じく単一プロセッサ水準（B）の5層からなる。これらは各々抽象マシン・インタフェースと考えることもできる。ついでながらAインタフェースは磁気バブル記憶装置に対する操作命令である。以下に各水準の保持する機能，特色について述べる。

（1）Fインタフェース

表現は制限された自然言語（擬似自然語）を用い，基本操作は射影（projection），制限（restriction），分割（division）に相当するものを持つ。結合（join）は物理的"関係（relation）"の概念がないので不要である。またデータ・モデル独立でもある。"view"，"conceptual schema"等[1]で呼ばれる論理属性指向で，ユーザの記述する擬似自然語に埋め込まれる引数はこの論理属性によって与えられる。

擬似自然語は幾つかの基本文型を用意しその引数部分を結び付けるといったQBE的方式が手軽である。擬似自然語によるデータベース用質向の表現は最近の自然言語処理研究の成果を採用して高度なユーザ・インタフェースを設定することも可能である。EDCLではこのインタフェースの仕様は固まっていない。

（2）Eインタフェース

表現は非手順的で基本操作は関係から関係と

つくることであり，結合概念が重要な役割を果す。表現・記述の簡単さ，読解性，均衡性，非冗長性を意図するが，インプリメンテーション独立で最適化への考慮はしない。

関係指向で操作対象あるいは操作単位が関係であり，記述にあたっては物理的な関係名，属性名に関する充分な知識が必要である。一時的に必要とされる作業関係（temporary relation）は明示的には存在しない。このインタフェースの言語仕様は本稿で詳しく検討される。

（3）Dインタフェース

Eと同じく表現は非手順的であるが，基本操作は組あるいは組相互関係についての条件を与えそれを満足する組を取出すことである。条件を表現するにあたって変数を用い，Eと同様に表現・記述の簡単さ，読解性，均衡性等を意図している。

組指向で操作対象，操作単位が組であり，物理的に存在する関係名のみ関係は存在する。組に対する条件の与え方等によって制限，結合などが表現され，最適化への配慮が可能となる。記述に使用される組変数（tuple variable）等の変数はインプリメンテーションにあたってそのまま変数として領域がとられるという意味でインプリメンテーション指向である。このインタフェースも後で詳しく検討される。

（4）Cインタフェース

高水準のデータベース操作機能を与える手順的なインタフェースである。通常の読出し，書込みの他，並列動作，連想探索，幾つかの関数命令等を基本命令として持つ。データ・モデル独立であり，並列動作する磁気バブル・データ・モジュール群を管理する制御プロセッサの持つデータベース操作機能である。

このインタフェースでの論理記憶空間のイメージは帯（band），ブロック（block）であり，そこに置かれたデータに対して連想アドレッシングや属性名による記号アドレッシングが可能である。

（5）Bインタフェース

このインタフェースは単一の磁気バブル・データ・モジュール内での高水準データベース操作機能を与える。内容的にはCインタフェースから並列操作を除き単一モジュール操作と見なしたものと同等であり，各モジュール・プロセッサ上に置かれる。

2.2 Eインタフェースの言語仕様

表2.1にE言語の構文規則を示す。言語仕様の複雑さの目安が一般に書換え規則の数であるとすれば，E言語はSEQUEL，SQUAREに比べて簡単化されている。

（1）文

〈表示〉，〈創製〉，…はそれぞれ“対象として指定された関係を表示，創製，…する”と読む。〈創製〉は組の容器としての関係をディレクトリ管理に登録する。組の属性名，型等も同時に指定される。〈削除〉はディレクトリ管理への登録を抹消する。〈挿入〉される関係，データは登録されている属性形式，型等と整合していなくてはならない。

（2）関係と属性

〈関係〉は射影部分，制限条件および他の関係との結合条件によって構成される。〈素関係〉を修飾する括弧内の属性指定は射影，制限条件指定は制限，さらに【…】内の結合条件指定は結合の各関係モデル基本操作を示す。結合条件の両辺に置かれる関係名が同一である場合を自己結合（self-join）と呼ぶ。素関係を修飾する

表2.1 E言語の構文規則

(1) ① 〈文〉 ::= 〈表示〉|〈創製〉|〈削除〉|〈挿入〉

② 〈表示〉::= List {〈関係〉|〈関数〉}[,…]

③ 〈創製〉::= Create 〈関係名〉(〈属性名〉(〈型〉[,〈定数〉])[,…])[,…]

④ 〈削除〉::= Delete 〈関係〉[,…]

⑤ 〈挿入〉::= 〈関係名〉← {〈関係〉|〈関数〉} | 〈関係名〉←Sort-{↑|↓}(〈属性〉;〈関係〉)

(2) ⑥ 〈関係〉::= 〈素関係〉[([{〈属性〉|〈関数〉}[,…]][:〈制限条件〉[(&|)…]])][【〈結合条件〉[(&|)…]】〈関係〉][{/|\}〈関係〉[{/|\}…]]

⑦ 〈素関係〉::= 〈関係名〉[時刻] | 〈素関係〉/〈属性〉[↑|↓]

⑧ 〈属性式〉::= 〈属性〉[↑|↓] | {〈属性〉|〈定数〉}{+|-|*|/}〈属性式〉

⑨ 〈属性〉::= 〈属性名〉

(3) ⑩ 〈制限条件〉::= {〈属性式〉|〈関数〉}{=|≠|<|≤|>|≥} {〈属性式〉|〈関数〉|〈定数〉|〈記号列〉} | {〈属性式〉|〈関数〉}{∈|∉} {〈関係〉|{〈定数〉[,…]}}

⑪ 〈結合条件〉::= {〈属性式〉|〈関数〉}{=|≠} {〈属性式〉|〈関数〉}

(4) ⑫ 〈関数〉::= min(〈関係〉)|max(〈関係〉)|average(〈関係〉)|sum(〈属性〉)|count(〈関係〉)

⑬ * ::= その〈関係名〉，〈属性〉について互て異なる組の値を持つ（ユニーク性）。

⑭ ＋ ::= 異なる素関係に属する同表現の属性名を区別する。

⑮ ’ ::= 分割された関係内の相異なる要素を区別する。

⑯ 〈関係名〉::= 物理的に存在する関係に付けられた記号名。

⑰ 〈属性名〉::= 物理的に存在する属性を指定する記号名。

⑱ 〈型〉 ::= 文字，整数，実数，2進数等。

⑲ 〈データ〉::= 文字列，2進符号列等。

注）メタ記号はBNF記法に従うものとする。またカンマはandを用いてもよい。

括弧内の関数指定（"関数射影"）は被修飾関係が分割された関係の場合のみ許される。また同時に指定される属性はその分割された部分関係内で同一の値を持つときのみ定義される。

〈素関係〉/〈属性〉は分割を示し分割された部分関係の個々を表現する。分割は属性の値による類別のみが行われるものとし，別に分割条件を与える自由度の大きい"条件分割"は許さない。分割された部分関係相互間の結合が存在することに注意されたい。〈属性式〉は既登録の属性名と定数からなる式である。

（3）制限条件と結合条件

〈制限条件〉は素関係の各組要素に対する取捨条件を与える。式中，関数が現われるのは関数射影と同じく被修飾関係が分割された関係である場合に限る。〈結合条件〉は［…］の両側に置かれた関係を結合する条件を与える。比較子の左右に置かれる属性式は各々対応する両側の関係の属性から構成される。関数が現われるのは分割された関係の個々の特性値を結合条件とする場合である。

（4）関数，その他

〈関数〉には minimum, maximum, average, sum, count の5つがある。引数として与えられる関係の射影部分は1項のみ指定されるものとする。関係として分割された関係が指定された場合，各部分関係毎に関数操作は行われる。

属性あるいは関係名を修飾する＊印は，その属性あるいはその関係に関して重複した組を持たないこと（ユニーク性）を表現する。

2.3 Dインタフェースの言語仕様

表2.2にD言語の構文規則を示す．複雑さの程度はE言語のそれとほぼ同程度である．

（1）文

〈表示〉，〈削除〉，〈挿入〉の対象が組である．表示の場合はその他に関数，作業変数を対象としうるが組との混在は許さない．挿入では関数の他，所謂データが対象となりうる．where 以下の条件部分で操作対象が拘束される．また組を直接修飾する制限条件によっても拘束を受ける．削除では変数条件式中に複数の変数を含んではいけない．削除の判定は1つの組または分割された部分関係を単位としてその中で閉じているものに限定される．

E言語での創製文はD言語では存在しない．但し一時的に存在する作業関係の創製，削除文があり，作業関係の属性形式，記憶領域等の管理はこのインタフェースで行われる．

（2）組と属性

〈組〉は射影部分，制限条件によって構成される．〈素組〉を修飾する括弧内の属性は射影を，制限条件指定は制限を各々示す。E言語での結合条件の指定は〈変数条件〉として与えられることに注意されたい。結合，自己結合等は変数条件によって陰に表現される。素組の前に置かれる all, any は各々全ての組，任意の1つの組を指定し，省略の場合は all と仮定される。unique は組単位のユニーク性を保証する。

組，分割された部分関係は共に変数によって指定され，各々組変数（tuple variable），分割変数（division variable）と呼ぶ。これら変数の存在領域は〈変数定義〉によって規定される。但し，その領域からの個々の要素の取出し方についての指定はなく非手順的に表現される。

〈分割変数〉を修飾する射影部分に属性と関数が混在する場合，属性の値は分割された部分

表2.2 D言語の構文規則

(1) ① 〈文〉 ::= 〈表示〉|〈削除〉|〈挿入〉|〈作業関係の創製・削除〉

② 〈表示〉::= List {〈組〉 / 〈作業変数〉 / 〈関数〉} [,…] where 〈変数定義〉[,…][,〈変数名〉[,…]]

③ 〈削除〉::= Delete 〈組〉[,…] where（②のwhere以下と同じ）

④ 〈挿入〉::= {〈関係名〉 / 〈作業変数〉}[,…] ← {〈組〉 / 〈関数〉}[,…] where（②のwhere以下）

⑤ 〈作業関係の創製・削除〉::= {Create / Purge}-work 〈関係名〉

⑥ 〈ソート〉::= Sort-{A / D}(〈属性〉;〈関係名〉,〈関係名〉)

(2) ⑦ 〈組〉::= {unique / [all] / any} 〈素組〉[{[{〈属性〉 / 〈関数〉}[,…]]}][:〈制限条件〉][{A / D}-到]

⑧ 〈素組〉::= 〈組変数〉|〈分割変数〉

⑨ 〈属性式〉::= 〈属性〉|{〈属性〉 / 〈定数〉}{+ / - / * / /}〈属性式〉

⑩ 〈属性〉::= 〈属性名〉

(3) ⑪ 〈制限条件〉::= {〈属性式〉 / 〈関数〉}{= / ≠ / < / ≦ / > / ≧}{〈属性式〉 / 〈関数〉 / 〈定数〉 / 〈定数列〉} | {〈属性式〉 / 〈関数〉}{in / 含む}{〈定数〉 / 〈定数列〉}[,…]

(4) ⑫ 〈変数定義〉::= 〈組〉[,…] ∈ {〈分割変数〉 / 〈関係名〉}[/〈属性〉]

⑬ 〈変数条件〉::= {〈組〉 / 〈関数〉 / 〈定数〉 / 〈定数列〉 / 〈作業変数〉}{= / ≠ / < / ≦ / > / ≧} {〈組〉 / 〈関数〉 / 〈定数〉 / 〈定数列〉 / 〈作業変数〉} | 〈組〉{∈ / ∉ / ⊇ / ⊃}{〈組〉 / 〈関係名〉 / (〈定数〉 / 〈定数列〉)}[,…])

(5) ⑭ 〈関数〉::= min(〈組〉) | max(〈組〉) | average(〈組〉) | sum(〈組〉) | count(〈組〉)

⑮ 〈関係名〉::= 物理的に存在する関係につけられた記号名．

⑯ 〈属性名〉::= 物理的に存在する属性を指定する記号名

⑰ 〈組変数〉，〈分割変数〉，〈作業変数〉::= 記号名によって参照される変数

注）E言語における＊，↑等の印はない。

関係内で全て等しいときに限り定義される。射影部分に関数が許されるのは分割変数を修飾する場合に限られる。

（3）制限条件

E言語と異って制限条件式中に関係名は現われない。また関数が現われるのは分割変数を修飾する場合に限られる。一般に制限条件は変数条件部分にも置くことができるが、関係を修飾する表現は許されない。制限条件を満さないとき素組の値は空と考える。

（4）変数定義と変数条件

組あるいは分割された部分関係を指す変数の存在領域は〈変数定義〉によって規定される。$\{\overset{\in}{\alpha}\}$の右辺で指定される集合が関係名か分割変数である場合左辺は組変数である。同じく分割された関係の場合左辺は分割変数である。

1つの文に現われる全ての組変数、分割変数はここで定義される。なお〈作業変数〉の定義は挿入文の左辺で行われ、以下で実行される文で使用される。

〈変数条件〉は変数間の相互関係の他、制限条件をここに置くこともできる。組に対する包含関係を与える表現では組の集合を表す分割変数を〈組〉が示す場合があることに注意。

（5）関数、その他

引数として与えられる組の射影部分は1項のみに限る。countの場合はその部分の指定を無視する。また引数で指定された組変数、分割変数の存在領域は変数定義によって与える。

〈関係名〉、〈属性名〉は物理的に存在する関係、属性を指定する記号名である。

3 E言語とD言語の特徴

3.1 E言語とD言語の対比

E言語とD言語の最も大きい異いはEが関係概念指向であるのに対しDは組概念指向である点である。Eでは変数の概念がなく、全て論理関係を構成することによって所望の組集合を表現する。従って構成過程での論理関係は物理関係には無配慮のまま記述される。

一方、Dでは組変数、分割変数、作業変数といった変数が重要な役割を果す。操作対象はこれら変数で与えられ、異なる組要素間の相互関係も変数条件として記述される。変数はインプリメンテーションにおいて実際に記憶領域上に確保される。また記述に現われる関係は全て物理関係であり、必要に応じて作業関係が使用される。

表現形式については E は関係代数的であるのに対し、D は関係論理的である。E は関係名、属性名等の置かれる位置情報が意味をもつポジショナル型であるのに対し D はキーワード型である。E の表現は記述の簡単さ、読解性を目的とし、操作対象・単位が論理関係でインプリメンテーションや最適化への配慮はない。D の表現は記述の簡単さに加えて表現の自由度を大きくし記述能力を強めている。属性とその関係名との対応も明示的に表し、E でのように位置関係や印等の便宜的手段に依らない。またインプリメンテーション指向であることは上述の変数、作業関係の導入の他、制限条件、変数定義、変数条件等の与え方による処理効率の異いなどに反映されている。

関係操作に関しては、E では射影、制限、結合、分割などの基本操作により物理関係から所望の論理関係を表現し、論理関係の集合演算も明示的に記述できる。一方、D では射影、制限は組変数、分割変数に対して、結合はそれら変数間の変数条件によって、分割は分割変数の変数定義によって与えられる。関係の集合演算等も変数定義、変数条件によって表現し、ユニーク操作以外は明示的記述手段を持たない。

表 3.1 E言語とD言語の対比

	E 言 語	D 言 語
概 念	関係指向	組指向
（変数）	なし	組変数、分割変数等
（作業関係）	なし	明示的に存在
表現形式	関係代数的	関係論理的
（型）	ポジショナル型	キーワード型
（結合条件）	明示的に存在	変数条件で陰に与える
インプリメンテーション	配慮なし	本来指向している。
（最適化）	配慮なし	余地有り、効率考慮

3.2 SEQUEL, SQUARE との比較

SEQUEL と SQUARE はユーザ・インタフェースの関係モデル質問言語としてそれなりの評価を受けている。[1] SEQUEL は中途半端ながらも自然語に近い表現形式を採用し、非プログラマのユーザに

とってはSQUAREより使い易いものとなっている[6]。SQUAREはfree variableを使用する場合を除いては記述は簡明であり，特に入る構造の質問に対する表現はコンパクトである。こうした利点を持つ一方，次のような欠点もある。

① SEQUELは擬似自然語表現を指向しているがそのことによって記述の曖昧さを招いている。

② 組変数に対応するcorrelation variable，computed variable（SEQUEL），free variable（SQUARE）の導入が表現の統一性，読解性を損っている。こうした変数は可能な限りそれらを使わない表現に代替されるべき必要悪的存在として扱われている。

③ SEQUELでは関数の定義域が文全体によって規定され，関数自体が自己閉包的でない。

④ SEQUELの文の係り方やSQUAREの条件式の与え方にみられるように，暗黙の約束事によって読解性が損われている。

⑤ 両言語とも関係論理を基礎としており，E言語のような結合操作は明示的には存在しない。またSQUAREは分割の明示的記述手段もない。

⑥ SQUAREは表示操作の対象が論理関係である場合と個々の組要素である場合があり，E言語とD言語の操作対象が混在している。

⑦ SEQUELは操作対象が組である点ではD言語と同じくし，入る構造の質問文を許し作業関係を認めない点でE言語と同じくする。インプリメンテーションへの配慮が中途半端である。

これらの欠点に対しEDCLは次のように考えて設計されている。

① 階層性を導入し例えば擬似自然語表現についてはE，D言語では考慮しない。

② 組変数はD言語で積極的に導入し他の変数と合せて表現の統一性，インプリメンテーションへの配慮をも行っている。

③ 関数の定義域はE言語では引数で与えられる関係であり，D言語では引数で与えられる変数の定義として明確に与えられる。

④ 操作対象を各々明確に限定し，条件式の与え方についても明示的表現を行う。

⑤ 結合操作はユーザ・インタフェースでは本来不要であるが，階層化された質問言語のあるレベルでは必要な概念であろう。

⑥,⑦ データベースの質問言語はユーザへの配慮，データ・モデルの表現への配慮，インプリメンテーションへの配慮の各々に対して適応しうる階層を持つべきである。

結局の所，SEQUEL，SQUAREは階層性への配慮がなくEDCLのF，E，Dの各言語機能と表現形式を脈絡なくピックアップして採用した言語といえる。Fインタフェースではユーザへの配慮をEインタフェースでは関係モデル表現の簡潔さと確かさを，さらにDインタフェースではインプリメンテーションへの配慮を全一義的に考えている。

4 記述例

本節で考える例の扱う関係および属性形式は次の通りと仮定する。

```
EMP:(EMPNO,NAME,DNO,SNO,JOB,SAL,ADDR)
DEPT:(DEPTNO,NAME,LOC)
SALES:(ITEM,TYPE,SNO,DNO,VOL)
```

この例で関係EMP（従業員ファイル）の属性DNO（部番号）と関係DEPT（部ファイル）のDEPTNO（部番号）は同じものである。EMPのNAME（従業員名）とDEPTのNAME（部名）は別物，更にMGR（管理者）の型はEMPNOと同じであることに注意されたい。

以下では射影と制限（Q1～6），結合と自己結合（Q7～16），関数（Q17～18），分割（Q19～23），分割と結合の混在（Q24～26）について質問文と記述例を示す。

4.1 射影と制限

［Q1］従業員ファイルから仕事が店員であるものの名前を全て示せ。

```
(E)  List EMP(NAME:JOB='CLERK')
(D1) List all x(NAME:JOB='CLERK')|x∈EMP
(D2) List x(NAME)|x(:JOB='CLERK')∈EMP
(D3) List x|x(NAME:JOB='CLERK')∈EMP
(S)  SELECT NAME FROM EMP WHERE JOB='CLERK'
(Q)  NAME^EMP_JOB('CLERK')
```

E，D，S，Qは各々E言語，D言語，SEQUEL，SQUAREによる記述を示す（以下同じ）。Dでは上記3つの等価な表現が許される。D1，D2，D3は磁気バブル・データ・モジュールからの物理的なデータ転送量を反映させることができるが，D言語インタプリタの内部処理で同一に扱うこともできる。以下の例ではこのような等価な表現のどれか1つを適当に選ぶものとする。なおDで

は関係に対する操作概念は存在しないので，x∈EMP(NAME:JOB='CLERK')という表現は許されない．またx(NAME:…)∈EMPで定義された組変数xがその文中で用いられるとき括弧内の制約付きの組と見なす．

[Q2] 従業員ファイルから仕事が店員で給与が10以上のもの全ての名前と給与を示せ．

```
(E)  List EMP(NAME,SAL:JOB='CLERK',SAL≥10)
(D)  List x(NAME,SAL)|x(:JOB='CLERK',
       SAL≥10)∈EMP
(S)  SELECT NAME,SAL FROM EMP WHERE JOB=
       'CLERK' AND SAL≥10
(Q)  NAME,SAL^EMP_JOB,SAL('CLERK',≥10)
```

Q1，Q2のような純粋に射影と制限だけの操作で充分な場合はどのような言語形式を持込んでも本質的差異はない．SEQUELの表現でも曖昧さ，冗長性は生じないし，SQUAREの表現もその記述様式を受容すると無駄がなくて美しいものとなっている．結局，結合や分割等の表現にどれだけ無理なく適応できるかが関係モデルの質向言語の表現能力，形式美を考える上での要因として重要である．

[Q3] 従業員ファイル中の全ての異なる仕事名を示せ（ユニーク性）．

```
(E)  List EMP(JOB*)
(D)  List unique x(JOB)|x∈EMP
(S)  SELECT JOB FROM EMP
(S2) SELECT UNIQUE JOB FROM EMP
(Q)  JOB^EMP
```

S2はSEQUEL II[3]を示す（以下同じ）．ユニーク操作を明示的に記述しないで，常に対象となる集合に対してユニーク性を保証する方式の欠点として次のことが挙げられる．

① 不必要な場合もユニーク性を保証する処理をするオーバヘッドが大きい．

② 組が重複した状態での処理，平均や算えあげなどではユニーク操作は不要である．

Eでは属性の右肩の＊印（この場合はEMP*(JOB)でも同じ）によって，Dでは組変数を修飾するuniqueによってユニーク操作を明示する．

[Q4] 従業員ファイルから部番号が19，22，41の者の名前を示せ（制限条件に集合）．

```
(E)  List EMP(NAME:DNO∈(19,22,41))
(D)  List x(NAME:DNO∈(19,22,41))|x∈EMP
(S)  SELECT NAME FROM EMP WHERE DNO IN
       (19,22,41)
(Q)  NAME^EMP_DNO('19','22','41')
```

SQUAREではこの表現でDNO∈{19,22,41}であることを暗黙の約束としている．Dでは制限条件を変数条件部分に移した表現も可能である．

[Q5] 従業員ファイルから福田の管理者の名前と給与を示せ（同一関係内での入子構造）．

```
(E)  List EMP(NAME,SAL:EMPNO=EMP(MGR:
       NAME='FUKUDA'))
(D)  1) X ← any x(MGR:NAME='FUKUDA')|x∈EMP
     2) List y(NAME,SAL:EMPNO=X)|y∈EMP
(S)  SELECT NAME,SAL FROM EMP WHERE EMPNO
       = SELECT MGR FROM EMP WHERE NAME=
         'FUKUDA'
(Q)  NAME,SAL^EMP_EMPNO·MGR^EMP_NAME('FUKUDA')
```

Dを除いては同じ関係に対する操作が入子構造になっている．Dでは作業変数（あるいは作業関係）を明示し2つの文で記述される．このような質向に対する記述にSQUAREは適している．

[Q6] 従業員ファイルと部ファイルから銀座にある部に所属する者の名前を全て示せ（異なる関係間の入子構造）．

```
(E)  List EMP(NAME:DNO∈DEPT(DEPTNO:LOC=
       'GINZA'))
(D)  1) WORK ← x(DEPTNO:LOC='GINZA')|
          x∈DEPT
     2) List y(NAME)|y∈EMP,y∈WORK
(S)  SELECT NAME FROM EMP WHERE DNO IN
       SELECT DEPTNO FROM DEPT WHERE
         LOC='GINZA'
(Q)  NAME^EMP_DNO·DEPTNO^DEPT_LOC('GINZA')
```

Dでは作業関係WORKが必要とされる．SQUAREではDNO∈{DEPTNO}が仮定されている(cf.Q5)．

4.2 結合および自己結合

(1) 結合の類型

関係数が2と3の場合の結合図式を図4.1に示す．結合は対応する各関係に属する属性間を結ぶ継手 —⊙— によって表される．同一組の対に複数の継手が考えられる場合や属性対応が1対1でない場合も1本の継手で表現する．

結合図式は関係を節，継手を枝とするグラフ（結合グラフ；join graph）と考えることができる．実際の結合操作はこのグラフ上で2つの節とそれを結ぶ継手を1つの節に縮退させる操作であり，それを繰返せば最終的には1つの節（関係）となる．グラフ状のものをE言語で線形に記述する場合，関係と結合条件中に置かれる属性の対応を明示しなくてはならない．全ての

属性名をユニークに名付けることは現実的でないので，曖昧性が生じる場合に限って両者の対応を t.# EMP等で明示する．1つの文中に現われる区別可能数を例えば3個返としても実用上はそれ程厳しい制約ではないと考えられる．

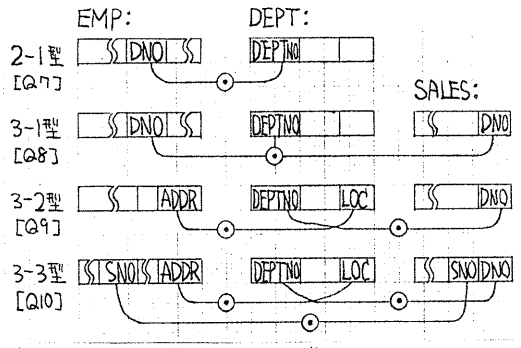

図4.1 結合の類型図式

（2）自己結合の類型

自己結合は1つの関係内の組要素相互間の結合である．1つの関係内の結合すべき組数が2とろの場合の自己結合を図4.1にならって図式化したものが図4.2である．この図式では後に示す例題に対応させて同一組の対に複数の縦手を表示してある．E言語による線形な記述に関しては結合の場合と同じ状況が生じる（Q13）．自己結合については図4.2で示す3つの類型以上に複雑な状況は実用上考慮する必要はないと思われる．

（3）結合と自己結合の混在

上述の各類型を各々混合させて図式表現することができる．結合と自己結合を各々2-1型と1型，3-2型と1型を組合せた例を図4.3に示す．

［Q7］従業員ファイルと部ファイルを用いて従業員名とその所属部名を示せ（2-1型結合）．

```
(E)  List EMP(NAME)[DNO=DEPTNO]DEPT(NAME)
(D)  List x(NAME),y(NAME)|x∈EMP,y∈DEPT,
       x(DNO)=y(DEPTNO)
(S)  SELECT NAME,X FROM Y IN EMP COMPUTE
       X = SELECT NAME FROM DEPT WHERE
           DEPTNO=Y.DNO
(S2) SELECT EMP.NAME,DEPT.NAME FROM EMP,
       DEPT WHERE EMP.DNO=DEPT.DEPTNO
```

Eでは結合条件式の左辺，右辺が各々EMP,DEPTの属性であり，表示対象が1つの結合された論理関係であることに注意されたい．

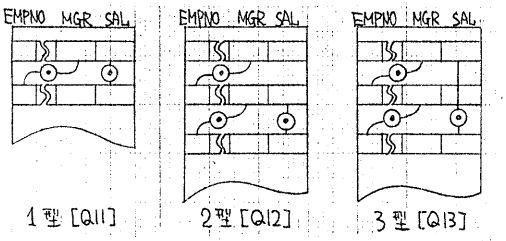

図4.2 自己結合の類型図式

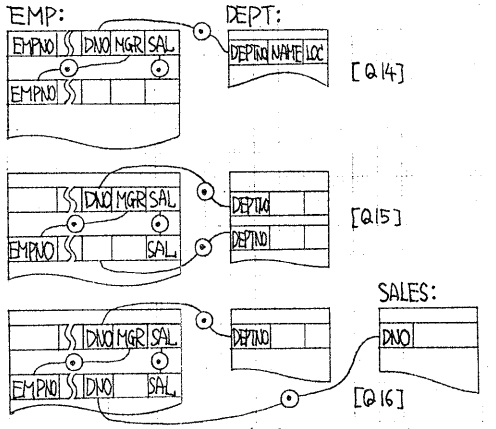

図4.3 結合と自己結合の混在図式

［Q8］福田の所属部名とその部の販売商品名を示せ（3-1型結合）．

```
(E)  List EMP(:NAME='FUKUDA')[DNO=DEPTNO]
       DEPT(NAME)[DEPTNO=DNO]SALES(TYPE)
(D)  1) X ← any x(DNO:NAME='FUKUDA')|x∈EMP
     2) List y(NAME:DEPTNO=X)|y∈DEPT
     3) List z(TYPE:DNO=X)|z∈SALES
```

この例ではEでの結合がDでは射影，制限操作のみに細分されている．

［Q9］住所が部の所在地と同じで給与が10以上の従業員名と，その所属する部の取扱い商品で型が電器に属するものをあげよ（3-2型結合）．

```
(E)  List EMP(NAME:SAL 10)[ADDR=LOC]DEPT
       [DEPTNO=DNO]SALES(ITEM:TYPE=
       'DENKI')
(D)  1) WORK ← x(NAME:SAL 10),y(DEPTNO)|
          x∈EMP,y∈DEPT,x(ADDR)=y(LOC)
     2) List w(NAME),z(ITEM:TYPE='DENKI')
          |w∈WORK,z∈SALES,w(DEPTNO)=z(DNO)
```

Eについては表現の簡潔さが必要だから，線形の結合の場合は端点の関係から記述を始めることが望ましい．この例でDEPTから始めると結合条件式の左右各項と関係との対応が均一でなくなる．Dでは処理の順序で手数が異なるので少ない手数となるよう結合順序を決めるなどの最適化の余地がある．

[Q10] 所在地がある従業員の住所と同じ部に属し，その従業員の所属課番号より大きい課で扱っている商品名を示せ（3-3型結合）。

(E) List EMP↑[ADDR=LOC]DEPT[DEPTNO=DNO, SNO↑<SNO]SALES(ITEM)

(D) 1) WORK ← x(SNO),y(DEPTNO)|x∈EMP, y∈DEPT,x(ADDR)=y(LOC)
2) List z(ITEM)|z∈SALES,w∈WORK, w(DEPTNO)=z(DNO),w(SNO)<z(SNO)

結合の図式表現がサイクルをなす例でEでの記述は↑印を用いて属性の所属する関係との対応を明示しなくてはならない。

[Q11] 自分の管理者より多くの給与をもらっている従業員の名前を示せ（1型自己結合）。

(E) List EMP(NAME)[MGR=EMPNO,SAL>SAL]EMP

(D) List x(NAME)|x,y∈EMP,x(MGR)=y(EMPNO), x(SAL)>y(SAL)

(S) SELECT NAME FROM X IN EMP WHERE SAL> SELECT SAL FROM EMP WHERE EMPNO= X.MGR

(S2) SELECT X.NAME FROM EMP X,EMP Y WHERE X.MGR=Y.EMPNO AND X.SAL>Y.SAL

(Q) $x_{NAME}{}^{\in EMP:\ x_{SAL}>_{SAL}{}^{EMP}{}_{EMPNO}(x_{MGR})}$

Eでは変数概念がない。SEQUELは1変数であるがSEQUELⅡでは2変数表現となっており読解性はよくなっている。

[Q12] 自分の管理者がその管理者の管理者より多くの給与をもらっている従業員名，その管理者と管理者の管理者の給与を示せ（2型自己結合）。

(E) List EMP(NAME)[MGR=EMPNO]EMP(SAL) [MGR=EMPNO,SAL>SAL]EMP(SAL)

(D) List x(NAME),y(SAL),z(SAL)|x,y,z∈EMP, x(MGR)=y(EMPNO),y(MGR)=z(EMPNO), y(SAL)>z(SAL)

Eでは結合図式が組に関して線形なので↑印を用いない記述が可能である。また同じEMP(SAL)の表現が2ケ所に現われるが最初のは管理者の，次のは管理者の管理者の給与を表現している。Dでは結合の例（Q9,Q10）におけるように作業関係を使用する表現も可能である。組変数x,y,zが名々ある人，その人の管理者，管理者の管理者に対応している。

[Q13] 自分の管理者の管理者よりも多くの給与をもらっている従業員名とその給与を示せ（3型自己結合）。

(E) List EMP↑(NAME,SAL)[MGR=EMPNO]EMP [MGR=EMPNO,SAL↑>SAL]EMP

(D) List x(NAME,SAL)|x,y,z∈EMP,x(MGR)= y(EMPNO),y(MGR)=z(EMPNO),x(SAL)> z(SAL)

3-3型結合に対応して自己結合がサイクルをなす例である。Eでは↑印による所属関係の明示が必要となる。

[Q14] 自分の管理者より多くの給与をもらっている従業員名とその所属部名を示せ（自己結合と結合の混在）。

(E) List EMP↑(NAME)[MGR=EMPNO,SAL>SAL] EMP[DNO↑=DEPTNO]DEPT(NAME)

(D) 1) WORK ← x(NAME,DNO)|x,y∈EMP, x(MGR)=y(EMPNO),x(SAL)>y(SAL)
2) List w(NAME),z(NAME)|w∈WORK,z∈DEPT w(DNO)=z(DEPTNO)

Eは↑印を用いない表現も可能である。自己結合と結合が混在する場合，一般には前者を先に処理した方が作業関係の大きさを小さくしうる。EからDに変換するとき"自己結合優先"の処理原則を設けることが妥当であろう。

[Q15] 自分の管理者より多くの給与をもらっている従業員名とその所属部名，およびその管理者の名前と所属部名を示せ（自己結合と結合の混在）。

(E) List DEPT(NAME)[DEPTNO=DNO]EMP(NAME) [MGR=EMPNO,SAL SAL]EMP(NAME)[DNO= DEPTNO]DEPT(NAME)

(D) 1) WORK ← x(NAME,DNO),y(NAME,DNO)| x,y∈EMP,x(MGR)=y(EMPNO),x(SAL)> y(SAL)
2) List w(NAME.1),u(NAME),w(NAME.2), v(NAME)|w∈WORK,u,v∈DEPT,w(DNO.1) =u(DEPTNO),w(DNO.2)=v(DEPTNO)

Eの記述は線形性を考慮してある。自己結合を前方に置くことは可能だがその場合は↑印を必要とする。また，出力形式を制御したい場合はFORMAT文的なものを用意する必要がある。Dで作業関係中の同じ属性名を区別するとき，作業関係に組を挿入したときの位置関係に従って前の方から.1，.2のように区別する。なお作業関係の属性名はそれが創られた元の関係の属性名をそのまま利用する。

[Q16] 自分の管理者より多くの給与をもらっている従業員名とその所属部名，およびその管理者の名前と所属する部の取扱い商品名を示せ（自己結合と結合の混在）。

(E) List DEPT(NAME)[DEPTNO=DNO]EMP(NAME) [MGR=EMPNO,SAL>SAL]EMP(NAME)[DNO= DNO]SALES(ITEM)

Dについては自己結合結果を作業関係に挿入後，3つの関係間の3-2型結合を行えばよい。

4.3 関数

[Q17] 従業員ファイルから部番号5の部の平均給与を求めよ（表示対象に関数）。

(E) List average(EMP(SAL:DNO=5))

(D) List average(x(SAL:DNO=5))|x∈EMP

(S) SELECT AVG(SAL) FROM EMP WHERE DNO=5

(Q) $AVG({}_{SAL}EMP'_{DNO}('5'))$

Eの表示対象は関係の他に関数があり，Dでは組の他に関数と作業変数がある。SEQUELでは関数AVGと操作SELECTの定義域が同一表現されている。なおSQUAREの'印はユニーク操作をしないことを指示する。

[Q18] 部番号5の部で一番給与の多い従業員名を示せ（制限条件に関数）。

(E) List EMP(NAME:SAL>max(EMP(SAL:DNO=5)))

(D) 1) X ← max(x(SAL))|x(:DNO=5)∈EMP
2) List y(NAME)|y(:SAL>X)∈EMP

(S) SELECT NAME FROM EMP WHERE SAL >
SELECT MAX(SAL) FROM EMP WHERE DNO=5

(Q) $x_{NAME}{}^{\in EMP:}\ x_{SAL}{}^{>ALL}{}_{SAL}EMP_{DNO}('5')$

Dでは入子構造の質問は段階を分けて順次記述する。射影部分に関数が現われる例は"各部毎の平均給与を示せ"というように関係を部分関係に分割する操作を含むので次節で与える。

4.4 分割

[Q19] 従業員ファイルから各部の部番号と平均給与を示せ（分割と射影部分に関数）。

(E) List EMP/DNO(DNO,average(EMP/DNO(SAL)))

(D) List g(DNO,average(g(SAL)))|g∈EMP/DNO

(S) SELECT DNO,X FROM Y IN EMP COMPUTE X=
SELECT AVG(SAL) FROM EMP WHERE DNO=
Y.DNO

(S2) SELECT DNO,AVG(SAL) FROM EMP GROUP BY
DNO

(Q) $x_{DNO}{}^{\in EMP,AVG}({}_{SAL}EMP_{DNO}(x_{DNO}))$

Eでは EMP/DNOによって分割された個々の部分関係を意味するのに対し，Dでは分割された状態にある関係（の全体）を意味し個々の部分関係は分割変数gによって与えられる。また，Dで表示対象に組変数，分割変数，関数等が混在することは許されないことに注意されたい（一般には各々定義域が異なり同時に表示対象となることはない）。

SEQUELではcomputed variable，SQUAREではfree variable，更にSEQUEL IIではGROUP-BY操作を用いて記述される。

[Q20] 給与が自分の所属する部の平均より少ない従業員の名前を示せ（分割と制限条件に関数）。

(E) List EMP/DNO(NAME:SAL<average(EMP/DNO
(SAL)))

(D) List x(NAME)|x∈g,g∈EMP/DNO,x(SAL)<
average(g(SAL))

Eの記述でNAMEと関数averageの定義域が同じであることが共にEMP/DNOであることによって示されている。表示対象をEMP(NAME:…)とすると全ての各部の平均より少ない者となる。

[Q21] 平均給与が10未満である部の部番号を示せ（分割と制限条件に関数）。

(E) List EMP/DNO(DNO:average(EMP/DNO(SAL))
<10)

(D) List unique g(DNO)|g∈EMP/DNO,
average(g(SAL))<10

(S) SELECT DNO FROM EMP WHERE 10 >
SELECT AVG(SAL) FROM X IN EMP WHERE
DNO=X.DNO

(S2) SELECT DNO FROM EMP GROUP BY DNO
WHERE AVG(SAL)<10

(Q) $x_{DNO}{}^{\in EMP:}\ AVG({}_{SAL}EMP_{DNO}(x_{DNO}))<10$

[Q22] ある部に所属する従業員の仕事の種類が存在する全ての仕事にわたる場合，その部の部番号を示せ（分割と制限条件に集合）。

(E) List EMP/DNO(DNO*:EMP/DNO(JOB)⊇EMP
(JOB*))

(D) 1) WORK ← unique x(JOB)|x∈EMP
2) List unique g(DNO)|g∈EMP/DNO,
g(JOB)⊇WORK

Dでは変数条件として集合の包含関係の表現を許す。

[Q23] 課の平均給与が他の課の最大給与より大きい課の課番号を示せ（制限条件に分割関係相互間の条件）。

(E) List EMP/SNO(SNO*:average(EMP/SNO
(SAL))>max(EMP/SNO'(SAL))

(D) List unique f(SNO)|f,g∈EMP/SNO,
average(f(SAL))>max(g(SAL))

Q19～Q22までは分割された個々の部分関係に関するものであったが，この例は部分関係同志の条件が現われる。Eでは異なる部分関係を明示する別印が必要であり，分割する属性の右肩に'印を付加する。Dではこの問題は2つの分割変数を導入することで表現上の問題はない。"他の課"の存在範囲を"同じ部"とするとき，制限条件として EMP/SNO(DNO*)=EMP/SNO'(DNO*)を付加すればよい。このとき両辺は1つの値のみを持つことに注意されたい。

4.5 分割と結合の混在

分割された部分関係を従来の組と見なしての結合が可能である。4.2で述べたような結合，自己結合の類型が存在する。このとき結合条件は分割された個々の部分関係を一意に特徴づけるもの，即ち部分関係を組と見なしたとき一意に定まる属性値を持つものに限られる。例えば図4.4の場合では，部番号DNO，各部毎の平均給与，各部毎の従業員数などである。

現実問題としてはこの種の負向は敢て結合を用いて表現しないで個々の関係に独立して段階的に処理すべきかもしれない。しかし，少なくとも理論モデルとしては可能であり存在しうるものである。

[Q24] 従業員ファイルと部ファイルから，各部毎の平均給与と部名を示せ（分割結合）。

```
(E)  List EMP/DNO(average(EMP/DNO(SAL)))
       [DNO=DEPTNO]DEPT(NAME)

(D)  1) WORK ← g(DNO,average(g(SAL)))|
          g∈EMP/DNO
     2) List x(NAME),y(AVERAGE)|x∈DEPT,
          y∈WORK,x(DEPTNO)=y(DNO)

(S)  SELECT NAME,X FROM Y IN DEPT COMPUTE
       X = SELECT AVG(SAL) FROM EMP
             WHERE DNO=Y.DEPTNO

(S2) SELECT DEPT.NAME,AVG(SAL) FROM DEPT,
       EMP GROUP BY DEPT.DEPTNO WHERE
       EMP.DNO=DEPT.DEPTNO

(Q)  x_NAME∈DEPT,AVG(x_SAL∈EMP_DNO(x_DEPTNO))
```

SEQUEL, SEQUELⅡは複雑な負向文では極端に読解性が悪くなる。分割された部分関係が明示的に表現されないので定義域が明確でなく，またGROUP BYやWHEREの修飾対象も曖昧な記述となっている。

[Q25] 部の平均給与が他の部の最大給与より大きい部とその最大給与の部の部番号を示せ（分割自己結合，cf Q23）。

```
(E)  List EMP/DNO(DNO*)[average(EMP/DNO
       (SAL))>max(EMP/DNO(SAL))]EMP/DNO
       (DNO*)

(D)  List unique f(DNO),unique g(DNO)|f,g
       ∈EMP/DNO,average(f(SAL))>max(g(SAL))
```

[Q26] 部毎の平均給与と売上高を示せ（分割相互結合）。

```
(E)  List EMP/DNO(average(EMP/DNO(SAL)))
       [DNO=DNO]SALES/DNO(sum(SALES/DNO
       (VOL)))

(D)  List average(f(SAL)),sum(g(VOL))|
       f∈EMP/DNO,g∈SALES/DNO,unique f(DNO)
       =unique g(DNO)
```

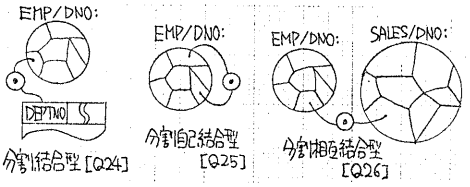

図4.4 分割と結合の混在図式

5 おわりに

幾つかの課題がある。D言語についてはその中間言語的意義（インプリメンテーション指向）を考慮して手順的にする（例えば結合はfor…do…;といった制御文で記述する），E言語では"any"機能の導入，類型図式のQBE的入力，出力形式の制御方式などが挙げられる。条件分割や分割の入子構造など負向の複雑な類型に対し記述能力としてどこまで応えるかは負向言語設計の上で重要な課題であろう。

関係モデルのデータベース負向言語はまだ暗中模索の段階であり，今後とも数多くの提案と開発が試行錯誤的になされる必要があろう。

末筆ながら，本研究の機会を与えられた西野博二パターン情報部長，石井治ソフトウェア部長，新田松樹記憶システム研究室長，またEDC開発グループの口分明男，大表良一，宮川正弘，菅原保雄，SSLの中山の諸氏を始め御討論いただいた関係各位に感謝する。

参考文献

[1]Date,C.J.:An Introduction to Database Systems, 2nd ed. Addison-Wesley Pub., 1977.

[2]Astrahan,M.M. and D.D.Chamberlin: Implementation of a structured English query language, CACM, Vol.18, No.10, Oct. 1975, 580-588.

[3]Chamberlin,D.D., et al: SEQUEL 2:A unified approach to data definition, manipulation and control, IBM J. Res. Dev., 20, 6, Nov. 1976, 560-575.

[4]Boyce,R.F., et al: Specifying queries as relational expressions;the SQUARE data sublanguage, CACM, 18, 11, Nov. 1975, 621-628.

[5]Zloof,M.M.: Query by example, Proc. NCC, 44, May 1975, 431-438.

[6]Reisner,P: Use of psychological experimentation as an aid to development of a query language, IEEE Trans., SE-3, 3, May 1977, 218-229.

[7]Su,S.Y.W. and A.Emam: CASDAL: CASSM's DAta Language, ACM Trans. Database Sys., 3, 1, March 1978, 57-91.

[8]Ozkarahan,E.A.,S.A.Schuster and K.C.Smith: RAP: An associative processor for data base management, Proc. NCC, 44, May 1975, 370-387.

[9]Lin,C.S.,D.C.P.Smith and J.M.Smith: The design of a rotating associative memory for relationa database applications, ACM Trans. Database Sys., 1, 1, March 1976, 54-65.