データベース・システム 67－2
（1988. 9. 22）

ERモデルに基づくデータベース設計支援システムの提案とそのERエディタの実装

西山 智　小花 貞夫

国際電信電話株式会社 上福岡研究所

近年のデータベースの複雑化、大規模化の傾向に伴い、データベース設計の困難さが増大している。これに対し、筆者らは、データベースを矛盾なく、しかも効率的に設計できるように設計者を支援するためのデータベース設計支援システム：DESSERTの構築を行っている。

データベース設計を困難にしている主な要因としては、1)概念設計における概念スキーマの試行錯誤的な決定プロセスが煩雑であること、2)概念設計/論理設計を通した各設計段階間でのスムーズな移行が困難であることが挙げられる。

DESSERTでは、これらの問題に対し、a)概念モデルの表現方法に、ERモデルを拡張したデータ・モデルを採用し、それに基づく視覚言語(Visual Language)による簡易な利用者インタフェースを提供することで、概念スキーマ決定のための試行錯誤を支援する、b)また設計の各段階で必要となるスキーマ変換や設計の正当性検証ツールをはじめとする様々な支援ツールを、同一の動作環境下で提供することにより、各設計段階間の移行を効率よく行えるようにする。

本稿では、DESSERTの構想について論じるとともに、その構成要素のうち、概念スキーマ決定の試行錯誤を支援するERエディタの実装について報告する。

Proposal on Computer Aided Database Design Support System based on Entity-Relationship Model and its Entity-Relationship Model Editor

Satoshi NISHIYAMA　Sadao OBANA

KDD Kamifukuoka R&D Labs. 2-1-15, Ohara, Kamifukuoka-shi, Saitama, 356

Trends of complexity and large scaling in database systems cause difficulty in their design processes. Under this circumstance, a computer aided database design support system called DESSERT (Database Engineering Support System based on Entity-Relationship model Technique), which supports design processes by human designers effectively and consistently, is proposed.

Some of the major reasons for difficulty in the design processes are ; 1) troublesome cut-and-try processes in the conceptual schema design, and 2) effective and smooth progression from one design step to the next, e.g., from the conceptual design step to the logical design step etc.

In the DESSERT, in order to solve the problems above, a) extended data model based on Entity-Relationship model is adopted for representing conceptual schema and a visual language for the user interface is provided, and b) all the supporting tools needed in conceptual / logical design steps are available under the same computational environment.

In this paper, a conception of DESSERT is discussed. Furthermore the Entity-Relationship Model Editor for supporting cut-and-try processes in the conceptual schema design, which is under development, is mentioned.

1. はじめに

近年、データベースの利用は極めて多様化、複雑化、大規模化してきており、それに伴ってデータベースの設計構築も困難となりつつある。その主な原因として、1)大規模データベースの概念スキーマ設計が極めて煩雑であること、2)概念設計/論理設計において様々な設計段階が存在し、かつある設計段階から次の段階への移行には、スキーマ/操作の変換や変換後の正当性検証等が必要であり、ある設計段階での設計変更が前設計段階に波及する場合のコストが大きいことなどが挙げられる。

これまでに、概念スキーマ設計を支援するための、概念モデルや概念スキーマ自動生成手法などが多数提案されてきた[4][8]など。しかし、概念スキーマ設計の際に設計者が行う試行錯誤を支援するツールはなかった。また、ある設計段階から次の段階への設計の移行を支援するために多くの手法も提案されており、その内いくつかはツールとして実装されている。しかしながら、これらのツールの多くは、データベース設計のある特定の段階に着目しており、全設計段階に対して計算機支援環境を提供するものではない。

そこで、これらの問題を解決し、データベースを矛盾なく、効率的に設計できるように設計者支援するため、筆者らは実体/関連モデル(ERモデル)[1]に基づくデータベース設計支援システムDESSERT (Database Engneering Support System based on Entity-Relationship model Technique)[2]の構想をたてた。

DESSERTでは、概念スキーマの表現方法としてERモデルを拡張したデータ・モデルを採用し、それに基づく視覚言語(Visual Language)による簡易な利用者インタフェースを提供することで、概念スキーマ設計のための試行錯誤を支援する。また、概念設計や論理設計の各段階でのスキーマ変換や設計の正当性検証ツールをはじめとする様々な支援ツールを、同一の動作環境下で提供することにより、設計の各段階間の移行を効率よく行えるようにする。

本稿ではDESSERTの構想について論じるとともに、その構成要素のうち概念スキーマ設計を支援するERエディタ[3]の実装について述べる。

2. データベース設計の現状

2.1 データベース設計の流れ

一般にデータベースの設計は、図1に示すように大きく(1)概念設計、(2)論理設計、(3)物理設計の3段階に分けることができる。概念設計ではデータベースの利用者(または応用プログラム)が必要とするデータの概念的あるいは抽象的な構造(概念スキーマ)とそれに対する操作(概念操作)を規定する。論理設計では概念スキーマと概念操作からデータベース管理システム(DBMS)固有のデータモデルで表現したデータ構造(論理スキーマ)とそれに対する操作(論理操作)を決定する。物理設計では、論理スキーマと論理操作からそれぞれディスク等物理的格納媒体における格納構造(物理スキーマ)とそれに対するアクセス手順(物理操作)を決定する。設計のある段階で要求仕様に矛盾が生じた場合、前の段階に戻って再設計を行う。

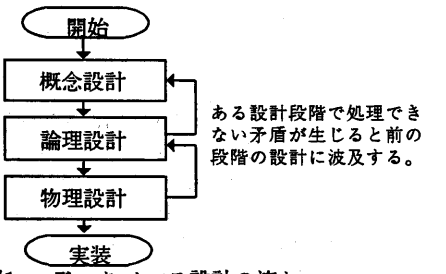


図1　データベース設計の流れ

2.2 データベース設計における問題点

今日の複雑化、大規模化したデータベースの設計では、以下のような問題点が生じている。

1) 実際のデータベース設計においては、実体、属性、関連などの概念に加えて汎化/専化、集約(aggregation)、導出データ、履歴データなどの概念も必要となる場合が多い。
2) 概念設計において、要求仕様からどれが実体、関連、及び属性であるかといった概念スキーマを決定することが困難である。
3) 概念スキーマは、通常複数の設計者により設計が行われる。各設計者は通常異なるビューを持っており、各設計者の概念スキーマ(局所概念スキーマ)から全体概念スキーマを統合することは困難である。また、統合後のビューは元の各設計者のものとは異なるため、各設計者による概念操作(局所概念操作)も統合後のビューにあわせて全体概念操作に変更する必要がある。
4) 設計者は各設計段階で、異なるデータ・モデルやビューによりスキーマを設計しなおす必要がある。このため、各設計段階で前段階の設計結果に基づいて設計を行うことや、その設計結果が前段階の設計(あるいは要求仕様)を満たしているかを検証すること等次の設計段階への設計の移行が困難である。

2.3 これまでのデータベース設計手法

これまでに、多くの設計手法が提案されてきた。例えば、2.2節1)の問題点に対してはERモデルに汎化/専化などの概念を拡張したEERモデル[4]やセマンティックデータモデル等の概念モデルが提案されている。また、3)の問題点に対しては局所概念スキーマから全体概念スキーマへの統合を支援する手法がいくつか提案されている[5]など。さらに概念スキー

マから特定のDBMS(例えばRDB)の論理スキーマを自動生成する手法などもある[6][7][9]など。問題点2)については、①各属性間の関数従属性や多値従属性から概念スキーマを合成する手法、②自然言語による要求仕様から概念スキーマを合成する手法[8]などが提案されているが、①は、全ての関数従属性や多値従属性をもれなく記述することが困難であり、また②についても、その要求仕様に現れる意味を理解して行うものではないため、完全な自動生成はできないなど、まだ問題がある。また、設計者による試行錯誤的な概念スキーマ決定プロセスの支援を行うものはない。問題点4)については、これらの設計手法の多くは単独の手法として提案されており統合的にデータベース設計支援を行うものは少ない。設計者は依然として設計各段階で様々な手法や環境に基づくツールを使用する必要がある。

3. データベース設計支援環境の基本的考え方

2.3節で述べたように、2.2節の問題点2)と4)についてはこれまでに有効な手法が殆ど提案されていない。また、1)の概念モデルについても実際のデータベース設計においては、これまで提案されたモデルでは不十分な場合もある。

そこで、筆者らはDESSERTの構築にあたってこれらの問題点に対して以下の方針をたてた。

① 問題点1)概念モデルについて

設計に用いる概念モデルは、可能な限り様々な概念を表現できる概念モデルを用いる。概念設計段階でより多くの概念を扱える概念モデルを用いることで、より実世界に近いスキーマで設計を進めることができる。このようなモデルを用いることにより、スキーマが概念モデルを満たしていることの検証や、論理設計への変換が困難になるが、ツールを用いてそれらを支援する。

② 問題点2)概念スキーマ設計の支援について

現在要求仕様から概念設計を自動生成するにはまだ解決すべき問題が多い。現状で、効率的なデータベース設計支援を行うためには「設計者による試行錯誤的な設計」を前提とし、その設計をツールにより支援する環境が有効と考えられる。このためにこのツールはマルチウインドウとか視覚言語といった良好な利用者インタフェースを持ち、かつ設計誤りを対話的に設計者に指摘する機能等が必要である。

③ 問題点4)各設計段階間の移行について

各設計段階間での移行を容易にするために、設計各段階で設計の変換ツールや検証ツールを整備し、これらのツールを同一の環境のもとで動作する必要がある。ここで検証ツールとしてはスキーマ/操作間の検証ツールや、ある設計段階の設計結果が前の段階の結果を満たしているかの検証ツールなどが考えられる。

また、これらのツールは単に同じ計算機環境の上で動作させるだけではなく、例えば、最初の設計段階でのツールに対する入力変更が自動的に後の段階のツールに伝播するといったように各設計段階でのツールを有機的に関連付けることで、設計誤りが前の段階の設計に波及した場合も容易に設計修正ができる。

4. DESSERT

筆者らは3節で述べたデータベース設計支援環境を実現するためにDESSERT(Database Engeering Support System based on Entity-Relationship model Technique)の構築を行っている。DESSERTでは3節の考え方に従い、設計者による概念設計を設計入力としている。

また、近年汎用DBMSを用いることにより、物理設計の大半をDBMSが肩代わりできることを考慮し、当面、DESSERTでは概念設計と論理設計を対象としている。

4.1 DESSERTの構成

DESSERTとして必要なツールを図2に示す。

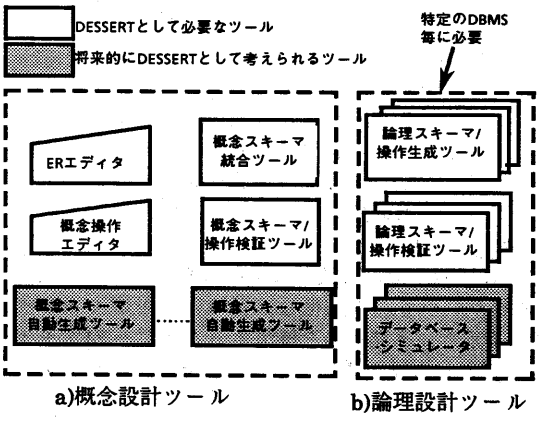


図2 DESSERTのツール構成

a)概念設計段階

- 概念スキーマ・エディタ(ERエディタ)
  概念スキーマを図形的に作成・編集するエディタ機能を提供する。さらに概念スキーマ自身の一貫性検証を行う。
- 概念操作エディタ
  概念操作を図形的に作成・編集するエディタ機能を提供する。
- 概念スキーマ/操作検証ツール
  定義された概念操作を用いて、設計(統

合)の結果得られた概念スキーマの正当性を検証する。

・ 概念スキーマ統合ツール

個々の設計者が設計した局所概念スキーマを統合し、全体概念スキーマを生成する。さらに、各局所概念操作を全体概念スキーマに対する全体概念操作に変換する。

b)論理設計段階

・ 論理スキーマ/操作生成ツール

概念設計段階で得られた概念スキーマ/概念操作から特定のDBMSに対する論理スキーマ/論理操作を生成する。複数のDBMSを支援する場合各々のDBMSに対して必要となる。

・ 論理スキーマ/操作検証ツール

生成された論理操作を用いて、論理スキーマの検証を行う。

また、将来DESSERTの拡張としては、以下のようなツールが考えられる。

a)概念設計段階

・ 概念スキーマを自動生成するツール

2.3節で述べたような自然言語等による要求仕様から概念スキーマ/概念操作を自動生成するツールなど。

b)論理設計段階

・ データベース・シミュレータ

テストデータを投入し実際にそれらに対して論理操作を行い、論理スキーマ/操作の正当性や論理的なデータのアクセス量などのチェックを行う。

4.2概念モデル

DESSERTでは、設計に用いる概念モデルとしてERモデルを用いる。但し、正確にはモデルの記述能力不足という問題点を解消するために、EERモデルに新たに履歴と導出関係の概念を追加したモデルを用いる。

図3に本モデルによる概念スキーマの図形表現を示す。

①実体/弱実体/関連

実体/弱実体は図3(a)/(b)のように表される。実体間の関連は2項関連/多項関連にかかわらず全て(c)のように表される。

②履歴実体

履歴を持つ実体は実体/弱実体に履歴データを表す記号を右下に追加して(d)のように表現される。但し、DESSERTでは当面履歴実体として、実体全体が履歴を持つものに制限し、各属性は履歴を持たない。属性は複数の値を持つことは可能であるが、それらの値の間に順序は定義できない。これは、もし、属性に履歴を持たせると、例えば属性aの3番目の値と属性bの5番目の値は同じ履歴であるといった履歴属性間の値と値に関係が生じ、DESSERTで使用している概念モデルの図形表現では表現が困難になるためである。

履歴実体に対してCURRENT(履歴実体):最新の履歴を指定、OLD(履歴実体):最も古い履歴を指定、PREV(履歴実体に対する関数式):1つ前の履歴を指定、NEXT(履歴実体に対する関数式):1つ新しい履歴を指定、などの集約(aggregate)関数を定義する。また履歴実体はキーの他に履歴キーを持つことができる。この場合、操作時にキーに加えて履歴キーを指定することにより普通の実体と同様に扱うことが可能となる。

③スーパータイプ/サブタイプ

ある実体E1が他の実体E2の部分集合である場合にスーパータイプ/サブタイプの関係と定義し、E1をE2のサブタイプ、E2をE1のスーパータイプと呼ぶ。サブタイプはスーパータイプの全ての属性を継承し、さらに、独自の属性を持つことができる。スーパータイプ/サブタイプは図3(e)のように表される。EERモデルでは、サブタイプを構成する部分集合を作る条件は任意である。しかし、DESSERTでは実装の簡略化のために、サブタイプはただ1つのスーパータイプを持ち、部分集合を作る条件はその特定の1属性の値に関する条件とした。

④汎化/専化

EERモデルでは同一のスーパータイプEを持つサブタイプE1..Enで、部分集合E1..Enに重複がなく、かつE1..Enの総和がEに等しい場合、EとE1..Enは汎化/専化の関係にあると定義する。従って汎化/専化はスーパータイプ/サブタイプの関係の特殊な場合と言える。汎化/専化関係は図3(f)のように表される。DESSERTではサブタイプ/スーパータイプの場合と同様に、専化の条件は特定の1属性の値に関する条件とし、また、専化実体はただ1つの汎化実体を持つ。

⑤実体/関連間

各実体間に存在する関連は各実体と図3(g)のように実線で結ばれる。弱実体が実体と弱関連の関係にある場合、弱実体と関連の間は、通常の関連と区別するために(h)のように矢印付き実線で示す。

関連の結合性は(i)のように示す。カーディナリティが1ではない(Nの)場合、関連の対応する隅を塗りつぶすことで示す。また、その相手実体が必ず存在すると限らない場合、実線に丸印を付加して示す。

⑥導出関係

実体間の導出関係は、円および実線、矢印付き実線を組み合せて図3(j)のように示す。DESSERTでは導出関係については概念スキーマ上実体間に何らかの関係があることを示すだけにとどめ、その関係内容はスキーマ/操作の検証には用いない。これは、実体間の導出関係は様々なものが考えられ、現時点では形式記述が困難であるためである。

⑦属性

EERモデルでは属性に関する図形表現も定義されているが、DESSERTでは属性に関する情報は各オブジェクト(実体/弱実体/関連)に属すると考え、属性に対する図形表現は概念スキーマ表現としては用いない。

EERモデルを用いて概念操作を表現するためには、実体等に対する条件付けや出力すべき項目の表示等若干異なる図形表現を用いる必要がある。この概念操作のためのEERモデル図形表現については、考察でふれる。

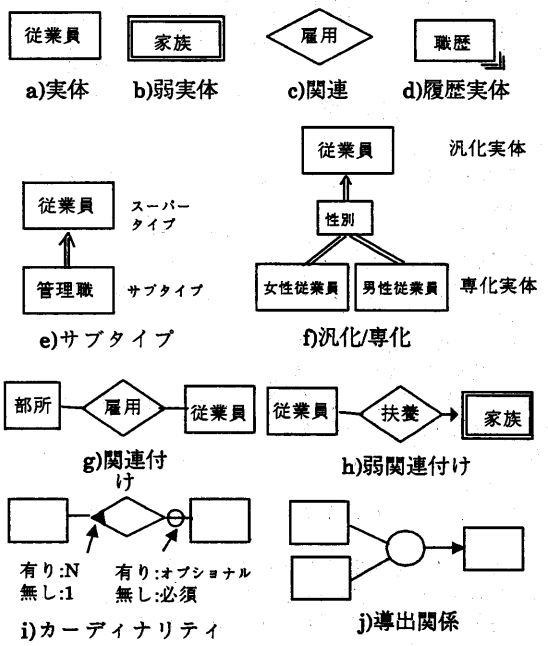


図3 EERモデル

4.3 DESSERTによる データベース設計の流れ

DESSERTによるデータベース設計は図4に示すように、大きく3つの段階に分けられる。

(1) 各データベース設計者による局所概念設計

設計対象のデータベースに対する要求仕様は、局所概念スキーマと局所概念操作の2つとして入力される。設計者はDESSERT上のERエディタを用いて局所概念スキーマの設計入力を行う。また、概念操作エディタを用いて局所概念操作の設計入力を行う。入力された局所概念操作と局所概念スキーマが相互に矛盾しないかどうかを概念スキーマ/操作検証ツールを用いて検証を行う。検証の結果矛盾が存在する場合は、局所概念操作あるいは局所概念スキーマの修正を行う。この段階の概念設計は、与えられた要求仕様を矛盾なく局所概念スキーマと局所概念操作として表現することである。

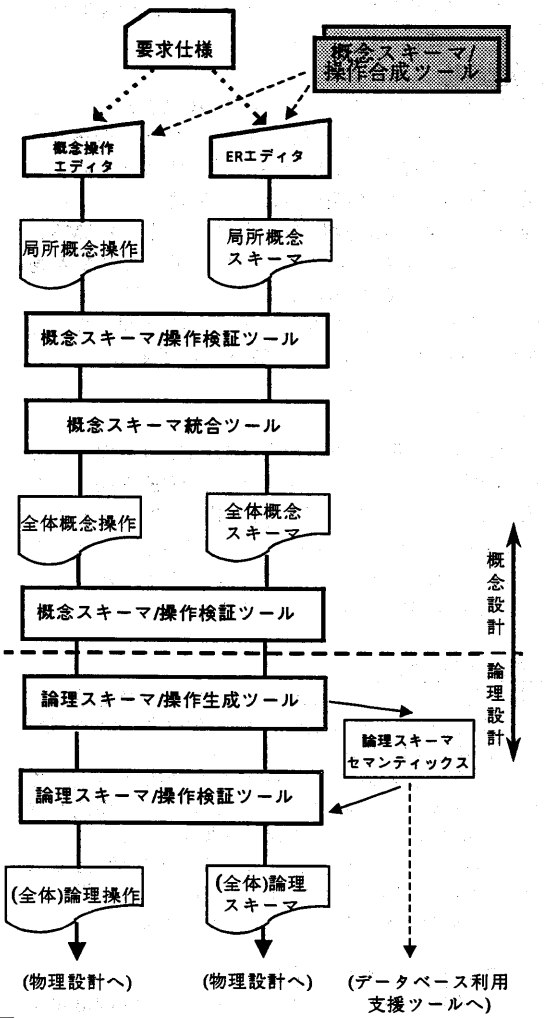


図4 DESSERTでのデータベース設計の流れ

(2) 局所概念設計から全体概念設計の統合

各設計者による局所概念スキーマを概念スキーマ統合ツールを用いて全体概念スキーマに統合を行う。この統合により、各局所概念スキーマから全体概念スキーマへの対応関係が定まる。概念スキーマ統合ツールはこの対応関係を用いて各局所概念操作を全体概念操作に変換を行う。最後に概念スキーマ/操作検証ツールにより全体概念操作が全体概念ス

キーマで実現可能であることを検証し、全体概念スキーマへの統合が完了する。

(3) 論理設計

全体概念スキーマから特定のDBMSを対象とした(全体)論理スキーマを生成する。また、全体概念スキーマから論理スキーマへの対応関係を用いて全体概念操作から(全体)論理操作への変換を行う。この論理スキーマ、論理操作の生成には特定のDBMSに対応した論理スキーマ生成ツールを用いる必要がある。さらに、特定のDBMSに対応した論理スキーマ/操作検証ツールにより論理スキーマ-論理操作間の無矛盾検証が完了すれば、論理設計が終了する。

5. ERエディタ

筆者らは、移植性を考慮してUNIX(汎用OS)とX-Window(汎用ウインドウシステム)をDESSERTの動作環境とした。これまでにDESSERTのツールのうち、ERエディタのプロトタイプを作成し、その評価をもとにERエディタをこの環境に実装を行っている。その詳細について以下に述べる。

5.1 ERエディタの目的

4節で述べたように、ERエディタは、概念設計において概念スキーマ決定のために設計者が行う試行錯誤的設計を支援する。このために、

1) 図形表現による概念スキーマを対話的に作成、変更、ドキュメント化するためのエディタ機能を提供し、概念スキーマの作図の支援を行う。
2) 作成した概念スキーマ自身の一貫性検証(例えば各実体/関連の名前の一意性など)により、概念スキーマ設計段階での誤りの検出を行う。

5.2 プロトタイプとその評価

当初、高機能パーソナル・コンピュータPC9800シリーズ(OS:MS-DOS)と汎用ウインドウ・システム(MS-Window)を用いて、ERエディタのプロトタイプを作成した。このプロトタイプの評価の結果、概念スキーマの修正が容易である、設計誤り(例えば実体間に関連が存在しない等)が発生しにくい等、ERエディタの有用性については実証できた。しかし、その利用者インタフェースの善し悪しがERエディタの実用性に大きく影響することも分かった。具体的には、マルチウインドウ機能、オブジェクト移動/複写時の位置決め機能等がプロトタイプに加えて必要とされた。

5.3 設計方針

5.1節及び5.2節から生じる要求事項に対して以下の方針で設計を行った。

a) X-Windowの上にERエディタのウインドウを生成し、その中でマルチウインドウを実現する。
b) エディタ操作は可能な限り、まず対象を選択し、その後操作を指定する方法に統一する。
c) 設計結果を外部ファイルに出力したり、外部ファイルから読み込んだりする機能を実現する。この外部ファイルはDESSERTで作成される他の支援ツールの入出力になることを考慮し、汎用性のあるものとする。
d) 対話的に設計を進めることを考慮し、設計結果のスキーマに対する検証は2段階に分けて行う。すなわち、設計中の各操作に対してはモデル違反等のみをエラーとする。従って例えば関連が単独で存在することも可能とする。全ての項目の検証は、設計者の指示により起動されることとする。

5.4 ERエディタによる概念スキーマ設計

5.3節の方針に基づきERエディタは図5に示すような操作メニューを提供する。

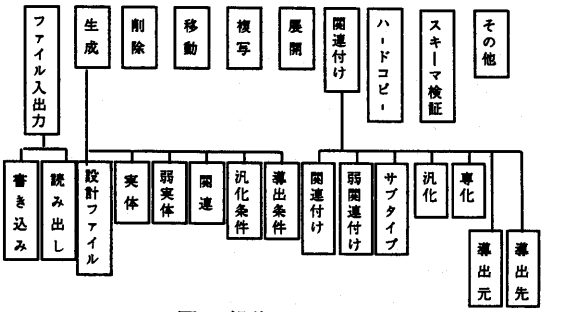


図5 操作メニュー

これらのメニューを用いて例えば各オブジェクト(実体/弱実体/関連)に対しては以下のような操作が可能である。

- 各オブジェクトの生成/展開/削除/移動/複写
- 実体間の階層関係(スーパータイプ/サブタイプ、専化/汎化)の設定(自動的に各実体間の情報の継承が行われる)
- 実体間の導出関係の設定
- 実体/関連間の関係付け/削除
- 各オブジェクトに、名前、コメント、属性、キー情報等必要な情報を設定
- 属性に、名前、型、コメント等各属性に関する情報を設定

また、設計者の指示により設計したスキーマに対して

① 各実体/弱実体/関連の名前が一意性、
② 各実体のキー定義がなされていること、
③ 各関連のカーディナリティが定義されていること、
④ 各実体/関連間の役割が定義されていること、
⑤ 孤立した実体/弱実体/関連が存在しないこと、
⑥ 各実体/弱実体/関連内の属性の名前が一意性、
⑦ 各属性に定義域が全て定義されていること、

等の検証項目で検証が行われる。但し、概念スキーマ検証機能のうち、アイコン操作時に検出できる概念モデルに対する違反、例えば

① 実体/関連間でキーとして用いられている属性に対する実体側での削除/キー指定の取消し

② スーパータイプや汎化実体を含まないサブタイプや専化実体単独での他のウインドウへの移動/複写

③ サブタイプや専化実体でスーパータイプや汎化実体から継承されてきた属性に対する変更/削除

等については、操作のたびにチェックを行う。

図6にERエディタによる概念スキーマ設計例を示す。

6. 考察

(1) 各ツールの実現

①概念操作エディタ

概念操作の入力方法としては、概念スキーマ上で対話的に指示する方法と、概念スキーマとは独立に操作の流れに基づいた図形表現で入力する方法が考えられる。前者の方法は、例えば1つの実体が同じ関連で2つの役割を持つ場合などで操作の入力が難しいと考えられるため、DESSERTでは、後者の方法が望ましいと考えられる。概念操作エディタは、後者、すなわち概念操作のための図形表現を用いて概念操作の設計入力を支援する。この図形表現については(2)で考察する。

②概念スキーマ/操作検証ツール

概念スキーマ/操作検証ツールは、与えられた操作が、そのスキーマ上で実行可能であるかを検証する。検証項目として、スキーマに対して操作に用いられる実体/関連が正しいこと、実体/関連間の役割が正しいこと、指定の属性が実体/関連に含まれること、条件の属性に対する値がその属性の定義域に含まれること、等が挙げられる。

③概念スキーマ統合ツール

概念スキーマの統合では、各局所概念スキーマにおいて単に各オブジェクトの名前とか属性が異なる場合にとどまらず、あるスキーマでは実体であるものが、他のスキーマでは関連であったり、また関連で結ばれた複数の実体であったりする場合も扱う必要がある。この統合の自動化は現時点では困難であると考えられる。従って、ERエディタと同様に設計者が対話的に統合を進めるための支援ツールが望ましい。

④論理スキーマ/操作生成ツール

概念スキーマから論理スキーマを生成する手法は、DBMSに対していくつか検討されている[6][7][9]。DESSERTで新たに加えた履歴実体についても、例えばRDBに対しては、陽に指定する履歴キーに加えて、履歴毎に1ずつ増加する履歴管理用の属性を自動的に実体に追加することで、4.2節②で述べた集約関数が実現できる。

⑤論理スキーマ/操作検証ツール

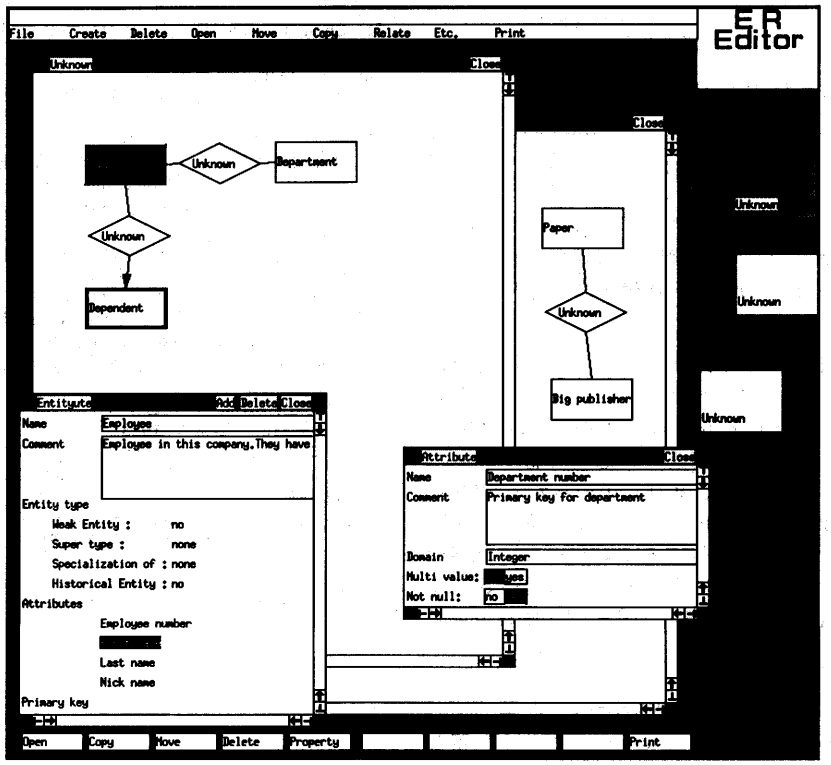


図6 ERエディタによる設計例

概念設計から論理設計が適切に生成されているか検証を行う。すなわち、生成された論理操作が論理スキーマ上で実行可能であることを検証する。また、概念スキーマから論理スキーマに変換する際に論理スキーマから欠落したセマンティクスに関する情報から、その論理操作に意味が存在するかを検証することも考えられる。

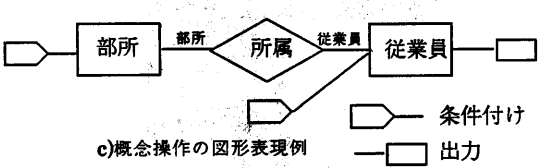
(2) 概念操作の図形表現

DESSERTでは概念スキーマと概念操作で要求仕様をなす。概念操作の入力には、検索条件などを表現する必要があり、概念スキーマの図形表現をそのまま用いることはできない。概念操作の図形表現が必要となる。この表現として一例を図7に示す。操作で検索条件が付く実体または関連は、「条件付け」を示す図形表現が付加され、実際の条件は、その「条件付け」のなかに記述される。出力する属性の存在する実体または関連に対しても「出力」の図形表現が付く。

a)概念スキーマの例

```
SELECT   氏名,年齢
FROM     従業員
WHERE    年齢＞50
  AND    部所(部所)/所属/従業員(従業員)
  AND    部所名＝'大阪支店'
```

b)概念操作のSQL的記述例



c)概念操作の図形表現例

図7 EERモデルによる概念操作記述例

(3) 属性の定義域について

考察(1)の②③④では、概念モデルでの属性の定義域を検証したり、論理スキーマに変換したりする必要がある。概念モデルでは、その定義域は例えば、「曜日」といった、任意の概念が可能である。従って、例えば検証では、条件に用いられる値(例えば'文化の日')がその定義域を満たしているかの判断を自動的に行うことは困難である。従って、現実的には設計者によりその定義域が形式的に記述されること(例えば"one of ('日曜日','月曜日'...'土曜日')")が必要になる。

(4) その他

概念スキーマから論理スキーマへの変換では、概念スキーマの持つ一部のセマンティクスが欠落してしまう。例えばRDBでは、テーブル間の関連は明示的には存在しない。論理操作を用いてデータベース検索を行う際に、設計支援で得られた論理スキーマと概念スキーマ間の対応関係を用いてこれらのセマンティクスを補うことで、データベース利用者に対する支援が行える。

7. おわりに

本稿では、実体/関連モデルに基づくデータベース設計支援システムDESSERTの構想とその概念スキーマ設計支援ツールであるERエディタの実装について述べた。DESSERTは、概念モデルにERモデルに基づくデータ・モデルを用いて、それに基づく視覚言語による良好な利用者インタフェースを提供することで、設計者の概念スキーマ決定を支援する。また、概念設計/論理設計で必要とされる支援ツールを統合的な環境のもとで使用可能とし、各設計段階間の移行を容易にした。これらの手法により、DESSERTでは効率的なデータベース設計が行える。

現在DESSERTでは、ERエディタの実装を行っている。今後は、DESSERTの他のツールの構築もあわせて進めていく予定である。最後に日頃御指導頂くKDD上福岡研究所 小野所長、浦野次長、鈴木コンピュータ通信研究室長に感謝します。

参考文献

[1]:Chen, P., "The Entity-relationship model - Toward a unified view of data", ACM TODS 1, 1, Mar., 1976.
[2]:西山,小花,"ERモデルに基づくデータベース設計支援システムDESSERTの提案", 第37回情処全大, Sept, 1988.
[3]:西山,小花,"データベース設計支援システムDESSERTのERエディタ", 第37回情処全大, Sept, 1988.
[4]:Elmasri, R., et al., "The category concept: An extention to the entity-relationship model", Data Knowl. Eng. 1, 1, 1985.
[5]:Batini, C., et al., "A comparative Analysis of Metodologies for Database Schema Integration", ACM Computing Surveys 18, 4, Dec., 1986
[6]:Teorey, T., et al., "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", ACM Computing Surveys 18, 2, June, 1986.
[7]:近藤他,"ERモデルに基づくデータベース設計支援ツール",第27回情処全大, 1983
[8]:川口,溝口他,"データベースの論理設計を支援する知的インタビューシステム",情処学会研究会報告AI-48,1986
[9]:小花,浦野,"Entity-Relationshipモデルに基づくデータベース・スキーマ設計法",第23回情処全大, 1981