データベース・システム　72-15
(1989. 7. 21)

知的設計支援のためのオブジェクト指向データベース
Object-Oriented Databases for Intelligent Design Support

宇田川　佳久　・　荻野　徹　・　石川　洋
Udagawa Yoshihisa　Ogino Toru　Ishikawa Hiroshi

芝　諭　・　市川　照久　・　辻　秀一
Shiba Satoshi　Ichikawa Teruhisa　Tsuji Hidekazu

(三菱電機㈱情報電子研究所)
MITSUBISHI Electric Corp.

あらまし　本文は，設計分野，とりわけ，回路設計で扱われるオブジェクトのモデル化，一貫性管理，検索，再利用について述べている。

Abstract　In this paper, we discuss issues of modeling, maintaining, retrieving and reusing objects in design applications, especially in circuit design.

1. はじめに

データベースは，必要な情報を，必要とするユーザにだけ，タイムリーに提供するための基盤となるものである。その萌芽は1960年代にまでさかのぼることができる。1970年代には，データベース・アーキテクチャの確立，汎用データベース管理システムの開発や利用技術の進歩を見た。データベースは，個人用のファイルを使った場合に比べて

・データの共用を推進できる
・データの標準化を推進できる
・データの冗長性を少なくできる
・データの機密保護に貢献する
・データの一貫性管理が容易になる

といった特徴がある。ファイルは個人によるデータの活用を支援するためのツールであるのに対し，データベースはデータを組織的に活用するためのツールと位置付けることができよう。

データを組織的に活用する，つまり，データを共同利用するためには，データベースが使いやすいものでなければならない。そのためにデータ構造の単純化，操作演算の簡潔化が追及されてきた。リレーショナル・データベースは，簡潔化の流れの時代に提案されたものであり，リレーションの正規化理論はその象徴的なものであると考えられる [4,10,12]。リレーショナル・データベースは商用化され，多くのアプリケーションに適用されてきた。特に，単一の値を操作の単位とするアプリケーション（在庫情報管理，売買情報管理，人事情報管理など）に適用したときに威力を発揮することが知られている。

一方，計算機，とりわけワークステーションとパーソナルコンピュータの普及に伴い，本来個人用に作られてきたデータを共同利用したいという要求が高まってきた。この傾向はＣＡＤ（Computer Aided Design）の分野で著しい。設計の工程ごとに，設計ツールが整備され，設計ツールの協調が重視されるようになってきたことが背景として考えられる [1,6,7,8,9]。

設計ツールが扱う情報は，本文２章で述べるように，図形や色々な単位を持つコード情報などから構成されているという特徴がある。つまり，管理の単位となる情報が一つの値として表現できないわけである。このような背景から，設計支援のためのデータベース（以後，設計データベースと呼ぶ）は，事務用のデータベースとは異なるものが必要になると指摘されている。

本文は，設計支援のためのデータベースと事務用データベースとの比較を試みるとともに，回路図を例にしながら，設計データベースの実現方法と実現結果について報告する。

2. 設計データ管理とオブジェクト指向

この章では，知的設計支援のモデルとして，回路の設計システムを考える。このシステムでのツールの概要を述べるとともに，データ管理の重要性を論ずる。

2.1 知的設計支援

回路の設計手順は，おおむね
(1) 回路の仕様を記述する
(2) 仕様を分析し，回路をブロックに分割する
(3) それぞれのブロックを設計する
という作業に分けることができる。

回路の仕様記述は，入出力の電気的特性，機能記述を行う作業である。機能記述には，タイム・チャート，

状態遷移図などを使うことが多い。

回路をブロックに分割する作業は，次の基準に従って行うとうまく行くことが経験的に知られている。

・すでにＩＣ化されている機能を中心にブロック化する。

・常套的に使われている回路がある場合は，その回路を一つのブロックにする。

すなわち，以前に設計した回路を流用することを考慮しながらブロック分割する，と言い換えることもできよう。

ブロックごとの回路設計は，次の３種類に大別することができる。

(a) 既設計ＩＣに条件設定をし，ブロックの仕様に合うようにする。

(b) 常套回路を手直しする。

(c) 新規に回路設計をする。

今までに開発されてきたＣＡＤシステムは，主に(c)の“新規に回路設計をする”作業を支援するものと位置付けることができる。新規回路の設計は，回路設計のうちで最も手間がかかる作業であるので，これを支援するシステムが役に立つことは衆目の一致するところである。

しかし，既設計回路のＩＣ化が進み，ＣＡＤシステムの普及によりカスタマが設計した回路がコンピュータ内に大量に蓄積されつつある現状を考えると，既設計ＩＣや常套回路の手直しを支援することが益々重要になってきていることも事実であろう。このためには，過去の設計事例を検索し，再利用を支援する設計データベースが必要になる。

２．２　設計データベースの主要機能

２．２．１　複合オブジェクトとしての回路図

回路図は，部品として使われている回路図や結線の状態などの構造に関する情報，回路図の構造をわかり易く表現するための図形情報などによってモデル化される。これらの情報は，通常，複数のレコード型に属するレコードの集まりによって記述される。回路図の構造と図形表現との一貫性を維持するためには，構造情報と図形情報とを手続きによって関連付けることが必要になる。つまり，回路図をモデル化するためには，レコードの集まりと，それらのレコードに対する手続きをまとめて管理するオブジェクト指向のアプローチが適切であると指摘されている。

回路図は単独で存在するものではない。回路図は，より基本的な回路図から組み立てられており，その回路図自身，より上位の回路図を構成するために引用されている。この様にして，回路図は階層構造を構成している。

一つの階層に属する回路図は，何回かの修正作業の末に完成するものであるから，変更履歴を管理する機能が重要になってくる。また，与えられた仕様を満たす回路を実現する方法が複数存在することが多い。変

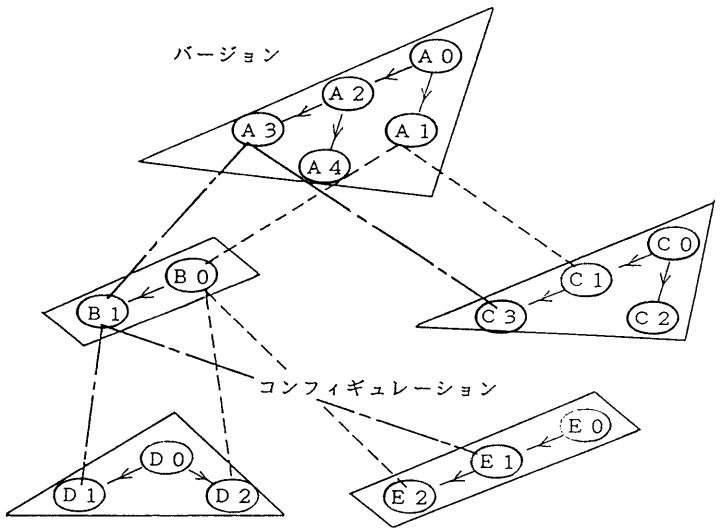

図１．バージョンとコンフィギュレーション

更履歴と複数の実現方法を管理することを総称して，バージョン管理と呼んでいる（図１)[6]。オブジェクト指向の用語によれば，回路の仕様がクラスに，その仕様を実現している回路図がインスタンスに対応する。さらに，インスタンスごとには，変更履歴が存在しているということになる。

一つの階層には複数の回路図バージョンが存在するので，回路図の階層構造も複数存在し得る（図１）。回路図の階層構造を管理する機能を，コンフィギュレーション管理機能と呼んでいる。

２．２．２　トランザクション

（A）事務用データベースとトランザクション

トランザクションは，一つの完結したデータ操作を行うオペレーションの集まりと定義される［4,10,11,12］。“完結したデータ操作”とは，データベースに課せられたデータの一貫性制約を保存するという意味である。トランザクションは，ユーザが定義した一連のデータ操作演算から構成されている。言い換えれば，トランザクションは，データベースの一貫性を保証しつつ，データベースの状態を変える一連のデータ操作演算ということができる。トランザクションには次のような特徴がある。

(1) トランザクションが完結したとき，データの更新はすべてデータベースに反映される。
(2) トランザクションが中断（abort)されたとき，その時点までに行われたデータの更新は，すべて帳消しにする。

つまり，更新の対象となる一連のデータは，すべて正しく更新されるか，全然更新されないかのいずれかで，中間の状態は許されない。この性質を，トランザクションの原子性（atomiticy)と呼んでいる。

トランザクションの原子性を実現するための一つの手法に，次のようなものがある。

(1) トランザクションの終り（更新の原子性を保証するタイミング）をユーザに明示してもらう（ＳＱＬでは COMMIT 文，または ROLLBACK 文を使う）[4]。
(2) データの更新命令があったとき，データベースの更新とともに，更新命令と更新の前後の値をログ・ファイルに書き出しておく。
(3) トランザクションが正常に終了したとき，ログ・ファイルを削除する。
(4) トランザクションが中断されてとき，ログ・ファイルを使って，データベースの内容をトランザクション開始時の状態に復帰する。

事務分野のアプリケーションは，データベースの小さな部分に対し，短い時間に，単純なデータ操作を行い，更新の原子性を厳しく守る必要があるという特徴がある。ログ・ファイルによるトランザクション処理の実現手法は，このような特徴に合致したものであった。

（B）設計データベースとトランザクション

設計支援システムにおけるトランザクション（以後，設計トランザクションと呼ぶ）は，次のような特徴がある［6］。

(1) 設計トランザクションは長い時間にわたることがある。

設計支援システムにおけるトランザクションの例として，回路図の編集，使用部品の選定，シミュレーション，部品の配置処理，部品間の配線処理といったものが考えられる。これらの処理には，数時間から数日といった時間がかかることも珍しくない。

(2) 設計トランザクションは多種類のデータ型のデータ更新を含む。

設計におけるオブジェクトは，一般に，文字列，数値，図形，画像情報の組み合わせによって表現される。これらのデータは，一貫した状態で操作されなければならないものである。

(3) 一つのオブジェクトが複数のユーザによって同時に更新されることはない。

一つの回路図を，複数のユーザが，同時に更新することに意味がないことからも明らかである。つまり，ユーザによって占有されるデータの最小単位は，複数のデータ型のデータによって表現されたオブジェクトである。ただし，同じ仕様を満たすオブジェクトを，複数のユーザが，同時に更新することは考えられる。これは，バーション管理として扱われる。

(4) 設計トランザクションにおける原子性は複数レベルある。

(2)で述べたように，設計トランザクションは多種類のデータ型のデータ更新を含むものである。原子性という側面から考えるとき，それぞれのデータ型のデータが原子的に更新されるレベルがあり，つぎに，データ型間の更新が原子的に行われるレベルがある。さらに，オブジェクトの階層構造の更新も原子的に行われる必要がある。

以上のような設計トランザクションの性質から，チェック・アウト／インの概念に基づくトランザクション管理手法が開発されている（図２）。この手法は，

データベースを“図書館”に見立て，チェック・アウトの時に

(1) 貸し出すオブジェクトの識別名

(2) 貸し出した時刻

(3) 借りていったユーザの名前

などの情報をデータベースに記憶しておく。

チェック・インした時には，

(1) オブジェクトの階層構造の更新

(2) 更新によって影響を受けるオブジェクトへの変更通知(Change Notification)，

などを行う。

設計トランザクションは，ネスト構造を成す。例えば，オブジェクト単位のトランザクション（オブジェクトの階層構造の更新を含む）は，オブジェクトの内部構造の変更に関するトランザクション（オブジェクトの引用・削除を含む）から構成され，さらに，内部構造の変更に関するトランザクションは，オブジェクトを表現しているそれぞれのデータ型に関するトランザクションから構成されている。

データベースの回復処理は，トランザクションのレベルごとに行われる。従って，最下位のトランザクション以外は，原子性を満足するとは限らない。

2．2．3　過去の設計事例の再利用

オブジェクトを検索する機能に関しても，従来データベースと設計データベースとでは違いが認められる。従来のデータ検索では，特定の属性値または値の範囲をもつレコードを取り出すだけであったが，ＣＡＤアプリケーションにおけるデータ検索では，２．１節で述べたように，過去の設計事例（オブジェクト）を取り出し，再利用を支援することが要求される。

オブジェクトの再利用は，おおよそ次のプロセスから成る。

(1) 検索したいオブジェクトの仕様を指定する。

(2) 候補となるオブジェクトをデータベースから取り出す。

(3) 最適なものを選ぶ。

(4) 書き替える。

(1)(2)のプロセスは，従来のデータベースにおける検索条件の指定と検索処理に相当する。これまで，検索条件の指定は、属性の値または値の範囲を指定することによって行われていた。一方、ＣＡＤアプリケーションを考えると，“属性の値”を指定することが検索条件として必ずしも適当であるとは限らない。さらに，検索条件を満たすオブジェクトを取り出すだけでは不十分で，検索条件として指定された仕様と検索したオブジェクトとの類似度を計算し，この類似度に基づいた順序に従ってオブジェクトを提示することが好ましい。つまり、オブジェクトの再利用を考えるとき、検索条件を満たすオブジェクトを検索することに加え，再利用の可能性を考慮したオブジェクトの知的な検索がきわめて重要な役割を演ずることになる。

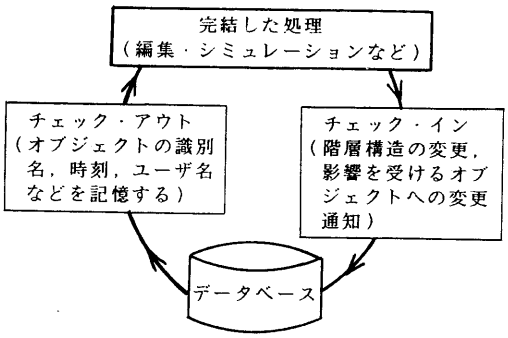

図２．チェック・アウト／インとトランザクション

3．　回路図の構造と図形の記述

この章では，回路図の構造と図形情報をリレーショナル・データベースに基づいて表現する方法について述べる。データと手続とのパッケージ化による，いわゆる，オブジェクト指向のアプローチが有効であることを例示する。

3．1　回路図の構造の記述

回路図は，より基本的な回路図を組み合わせて作られている。回路図の構造を表現するためには，異なるデータ型のレコードの集まりが必要となる。回路図のモデル化の一例として，回路図を次のような構造をもつオブジェクトとして捕えることができる。

(1) 回路図は，端点を持っている。

(2) 端点は，入力端点と出力端点がある。

(3) 結線は，二つの端点を結ぶ。

“回路図”と“端点”を“識別できるもの”と考え，“結線”を“端点”どうしを“結ぶもの”と解釈することができる。このように，モデル化の対象となる現実世界を，識別できる“実体”と，実体どうしを結ぶ“関連”によって表現する手法が“実体－関連モデル”と呼ばれるものである。実体を長方形で，関連を菱形で表現することが一般的であり，ＥＲ図と呼ばれてい

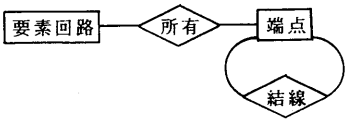

図３．実体・関連図の例

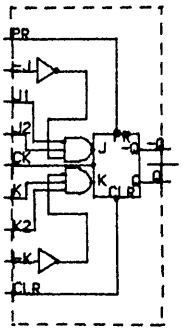

図４．回路図の例

```
COMP( DIV_ID/ID, DIV/ABS ).
     1  D001, JNCPS(X-3.0,Y)
     2  D002, ANT3S(-9.0,+2.0)
     3  D003, ANT3S(-9.0,-2.0)
     4  D004, INTS(-14.0,+12.0)
     5  D005, INTS(-14.0,-12.0)
     6  D0, M70(X,Y)


TERM( T_ID/ID, DIV_ID/ID, CHR/ID, XC/AD, YC/AD ).
     1  Q0, D001, TP_N, -2.5, +2.0
     2  Q1, D001, TP_N, -2.5, -2.0
     3  I0, D001, TP_N, -9.5, +2.0
     4  I1, D001, TP_N, -9.5,
     5  I2, D001, TP_N, -9.5, -2.0
                 . . .
    26  I006, D0, TP_N, -20.0, -8.0
    27  I007, D0, TP_N, -20.0, -12.0
    28  I008, D0, TP_N, -20.0, -16.0
    29  Q000, D0, TP_N, , +2.0
    30  Q001, D0, TP_N, , -2.0


CONN( L_ID/ID, S_ID/ID, DIV_S/ID, D_ID/ID, DIV_D/ID ).
     1  L001, I000, D0, I4, D001
     2  L002, Q0, D001, Q000, D0
     3  L003, Q1, D001, Q001, D0
     4  L004, I008, D0, I3, D001
     5  L005, I004, D0, I1, D001
                 . . .
    11  L011, Q0, D004, I2, D002
    12  L012, I007, D0, I0, D005
    13  L013, Q0, D005, I0, D003
```

図５．回路図を表現するリレーションの例

る [2,11] 。図３は，上記の構造をもつ回路図を図示したものである。

一般に，実体－関連モデルで表現されたオブジェクトは，リレーショナル・データベースで実現することができる。ここで考えている回路図の場合，実体“回路図”と“端点”，それに関連“結線”に対してリレーションを割り当てれば良い。図４に示した回路図を，リレーショナル・データベースを使って表現した例を図５に示した。このデータベースには COMP , TERM, CONN と名付けられた３個のリレーションが使われている。つまり，回路図というオブジェクトは，“回路図”，“端点”，“結線”に対応するリレーションにまたがるレコードの集まりによって表現されるわけである。これらのレコードは，回路図オブジェクトをアクセスするたびに，ひとつの単位として扱われるものである [5,6]。これに対し，事務処理の応用では，オブジェクトは一つのリレーションに含まれることが多く，このリレーションがアクセスの単位となる。

図５のリレーション COMP は回路図が使っている部品回路図に関する情報を記憶しており，属性として DIV-IDと DIVをもっている。属性 DIV-ID は，部品となっている回路図のキーである。属性 DIVは，部品となっている回路図の識別名，表示座標を保持している。

リレーション TERM は回路図の端点に関する情報を記憶しており，端点のキー（T-ID），端点が属している回路図の識別名（DIV-ID），端点の特性（CHR ）それに表示座標（XC,YC ）を保持している。

リレーション CONN は，結線の状態を記憶しているリレーションで，結線の識別名（L-ID）と，結線の始点の識別名（S-ID,DIV-S）と終点の識別名（D-ID,DIV-D）を保持している。なお，結線の図形表現は線画像ファイルに記憶されており，結線の識別名をキーにして操作される。

部品回路図が更新される度に，リレーション COMP と TERM の内容が更新される。また，結線が行われる度にリレーション CONN と線画像ファイルの内容が更新される。

3．2 回路図の図形表現の記述

回路図の図形表現は，回路図の構造を常に反映したものでなければならない。この要求に答えるために，回路図の構造情報を図形に変換する手続きを使う。

例えば，回路図の入力端点と出力端点を表示するためには図６に示した手続きを実行する。つまり，リレーション TERM から部品となっている回路図の端点を除き（ DIV-ID ＝'D0' ），端点の属性に応じて円形（ CIRCLE ）または，十字印（ CROSS)を表示する。

```
DISPLAY   CIRCLE ( RADIUS = '0.5',
  SELECT  XC, YC
  FROM    TERM
  WHERE   DIV_ID = 'DO'
    AND   CHR = '***I*' ) ;
DISPLAY   CROSS  ( RADIUS = '0.5',
  SELECT  XC, YC
  FROM    TERM
  WHERE   DIV_ID = 'DO'
    AND   CHR = '***N*' ) ;
```

図6．回路図の図形表現の手続

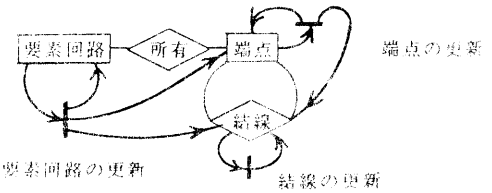

図7．ERT図の例

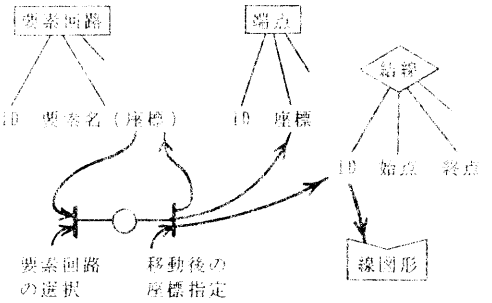

図8．更新処理の詳細

4．回路図編集とトランザクション

この章では，回路図に対する操作とトランザクションとの関係を記述する表記法を提案する．また，回路図の編集操作の実現例を示す．

4．1 実体・関連図とトランザクション

オブジェクト同士の依存関係を表現する手法の一つに実体・関連図（以後，ER図）がある．ER図は，実世界の構造を把握し，データベース設計を支援するのに有効なものであるが，ER図にはトランザクションの概念が含まれていないために，更新処理を含めた実世界を記述することは難しい[11]．そこで，実体と関連，それにトランザクションの相互関係を表現できるようERR図を拡張したERT図（Entity-Relationship-Transaction Diagram）を用いる[13]．トランザクションをモデル化するためには，複数の実体と関連との同期を扱わなければならないことが多い．ERT図では，ペトリネットを使ってトランザクションにおける同期を表現している．

図7は，回路図を構成するオブジェクトとオブジェクトの更新処理の関係をERT図で表現したものである．例えば，“要素回路”に対する更新処理は“要素回路”に関するデータの更新のみならず，“結線”と“端点”に関するでーたの更新も引き起こすことを示している．

図8は，“要素回路”に対する更新処理を詳細に表したものである．この図では，更新処理が

(1) “要素回路”の“座標”を書き替えていること
(2) “端点”の“座標”を書き替えていること
(3) “結線ID”を手掛かりに線図形の更新を行うこと
(4) 処理は“要素回路の選択”と“移動後の座標指定”という事象によって同期が取られていること

を示している．

4．2 回路図編集機能の実現

図9は，要素回路を移動する機能を実現するための手順をHCP図で表現したものである．ユーザは，移動したいモジュールを選び，移動先を指定する．これに応じて，要素回路の座標，端点の座標，要素回路に接続している結線の図形情報を更新する．

図10は，要素回路の出力端点に接続している結線の図形を更新する処理を，埋め込みSQLに準じて表現したものである．このプログラムで使われている変数およびサブルーチンの意味を次に示す．

＊変数X，Yは，要素回路の移動前の座標を記憶している．この変数の値は“要素回路の選択”のときに指定され，これに基づいて，ユーザが指定した要素回路の識別名を検索する．
＊変数XX，YYは要素回路の移動量を記憶している．この変数の値は，“移動後の座標指定”の時に指定され，これに基づいて，結線の図形を更新する．
＊WIDは検索された要素回路に接続されている結線の識別名を記憶する変数である．この変数の値は，リレーション CONN の属性 L-ID から選択されたものである．

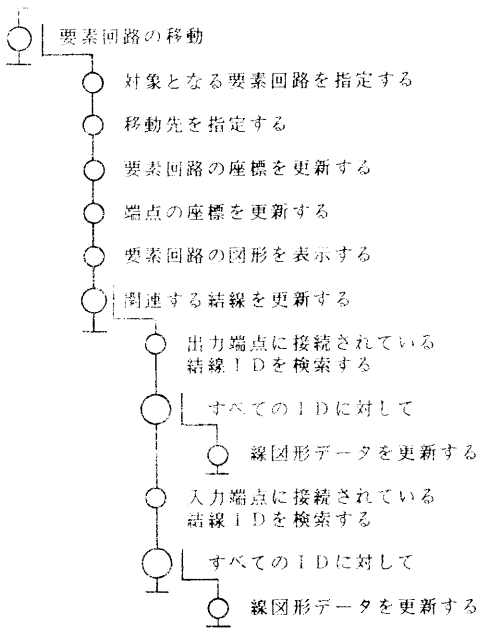

図９．要素回路を移動する手続

```
C/**  Get identifiers of wires connected
C/    to output ports of the specified
C/    component circuit.
C
      $LET  WIREID  BE
     *      SELECT  L_ID
     *      INTO    $WID
     *      FROM    CONN
     *      WHERE   DIV_S =
     *              (SELECT  DIV_ID
     *               FROM    COMP
     *               WHERE   DIV(1) = $X
     *               AND     DIV(2) = $Y
      $OPEN  WIREID
C
C/**  Update geometric data of the
C/    selected wires.
C
      DO  100  K = 1, 1000
      $FETCH  WIREID
      IF( SYS_CODE .NE. 0 )          RETURN
      CALL    UPDWIRE( WID,XX,YY,0.0,0.0 )
  100 CONTINUE
C
      OUTPUT    "//TOO MANY WIRES//"
      RETURN
      END
```

図10．プログラム例

＊ SYS-CODE はデータベースの検索が正常に実行されたときに値がゼロとなる変数である。

＊サブルーチン UPDWIRE は，指定された結線の図形を始点および終点の移動量に応じて変更する。

図11は，要素回路の移動をしたときの一例である。回路図の編集作業のトランザクションの構造は，図12に示したように３レベルある。つまり，

・一枚の回路図を，回路図の階層構造から取り出し，再度戻すといった複合オブジェクトに関するトランザクション

・要素回路の移動，端点の移動，結線の更新といった回路図を構成するオブジェクトに関するトランザクション

・要素回路，端点，結線といったオブジェクトを表現しているデータに関するトランザクション

のネスト構造を成している。

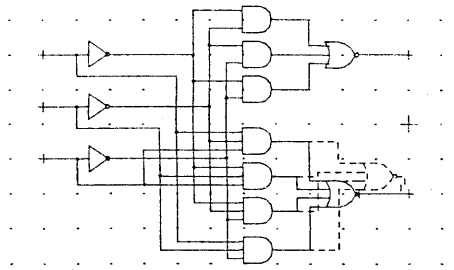

図11．要素回路を移動した例

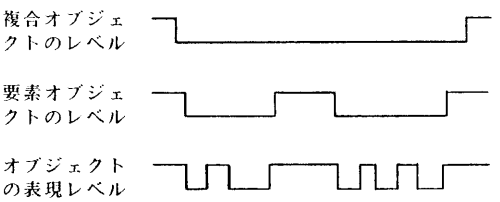

図12．要素回路の移動とトランザクションの構造

5．オブジェクトによるオブジェクトの検索

この章では，オブジェクトとバリューとの違いを述べると共に，データベースでは，オブジェクトによるオブジェクトの検索が本質的であることを論ずる。ま

EMP	Name	Gender	Age
	AA	M	21
	BB	M	35
	CC	F	17
	DD	F	23

図13. リレーションの例

た，設計で扱われるオブジェクトは，多様な側面をもつために，類似に基づく検索が適当であることを述べる。回路図による回路図の検索事例も示した。

5.1 オブジェクトとバリュー

データベースには，現実世界を忠実に反映したデータが維持されなければならない。つまり，データベースが扱っているデータは，現実との対応が明確なオブジェクトであり，時間と無関係に存在するバリューとしてのデータを扱っているのではない。

オブジェクトは時間とともに変化するものであり，年齢，給与や氏名などがその例である。オブジェクトを操作するにはオブジェクトを使うのが自然であるが，従来のデータベースで扱っていたオブジェクトの多くが数値や文字列で表現されていたために，オブジェクトとバリューとが，ややもすると混乱して使われることがある。

オブジェクトをバリューによって検索する方法は，多くの場合に有効なものであるが，オブジェクトが時間とともに変化するために，めんどうな問題を引き起こす場合があることが知られている。その簡単な例として，図13に示したリレーションEMPから“従業員DDよりも若い従業員”を検索したいとする。従業員DDの年齢は23であるから，この検索をSQL[4]で表現すると

```
SELECT      *
FROM        EMP
WHERE       Age  <  24
```

となる。検索結果は

EMP	Name	Gender	Age
	AA	M	21
	CC	F	17
	DD	F	23

である。一方，従業員の年齢は年とともに変化するから，リレーションEMPの属性Ageの値が正しく更新されていると仮定すると，2年後に上記の検索をすると結果は，

EMP	Name	Gender	Age
	AA	M	23
	CC	F	19

となる。時間に対して不変なバリューによって，時間とともに変化するオブジェクトを検索しているのであるから当然の結果である。この結果は検索式を書いたユーザの真意と一致しないから，このユーザは2年前に書いた検索式を

```
SELECT      *
FROM        EMP
WHERE       Age  <  26
```

と書き換えなければならない。このような書き換えが必要になったのは，データベースはオブジェクトを管理しているのに，検索はバリューによって行っていることに起因する。

上記の質問は

```
SELECT    *
FROM      EMP
WHERE     Age  <
      (  SELECT    Age
         FROM      EMP
         WHERE     Name = "DD"
```

と表現することもできる。この表現は，年齢の値を直接指定するものよりも変更の可能性が少ない表現であるが，変更の可能性が無いわけではない。なぜなら，この従業員の名前が変わることが起り得るからである。Codd[3]らは，ユーザによって指定されたキーによって現実世界のオブジェクトを特定するのには限界があることを論じるとともに，データベース・システムが生成する識別名（サロゲート）を導入することを提唱している。

5.2 オブジェクトと類似検索

設計分野で扱われるオブジェクトは，複数のデータ型のデータによって表現されることが多い。また，オ

ブジェクトは，オブジェクトの階層構造の中で位置付けられるものである。つまり，オブジェクトは要素となるオブジェクトを引用したり，上位のオブジェクトから引用されることもある。このよに，オブジェクトは複雑な側面を有するものである。

一方，バリューを基本とするデータベースの検索手法は，オブジェクトの限られた側面（属性値）に注目した検索機能を提供している。この手法が複雑な側面を持つ設計分野のオブジェクトの検索に適しているかということになると疑問が残る。むしろ，オブジェクトそのものを検索条件とし，データベース内のオブジェクトとの一致の度合い（類似度と相違度）に基づいた検索が好ましい。図14は，従来の検索手法と一致の度合いに基づいた検索手法の概念を図示したものである。

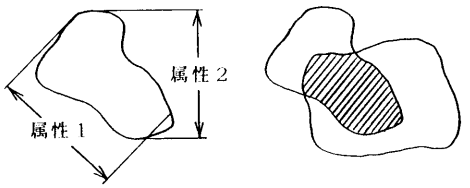

図14．属性値による検索と部分一致による検索

5．3　回路図による回路図の検索

ここでは，回路図を検索条件として回路図を検索するアプローチについて説明する［14］。この方法によれば，ユーザは欲しいと思っている回路図の概略図（以後ルアー回路図と呼ぶ）を通常の回路図の編集と同様な方法で編集し，その後，このルアー回路図に類似する回路図の検索をシステムに依頼するだけで所望の回路図を検索できるものである。

まず，ルアー回路図と候補回路図との類似度，相違度を次の計算式に基づいて計算する。

類似度　　Cardinal（　X∩Y　）

相違度　　Cardinal（　X－Y　）

＋　Cardinal（　Y－X　）

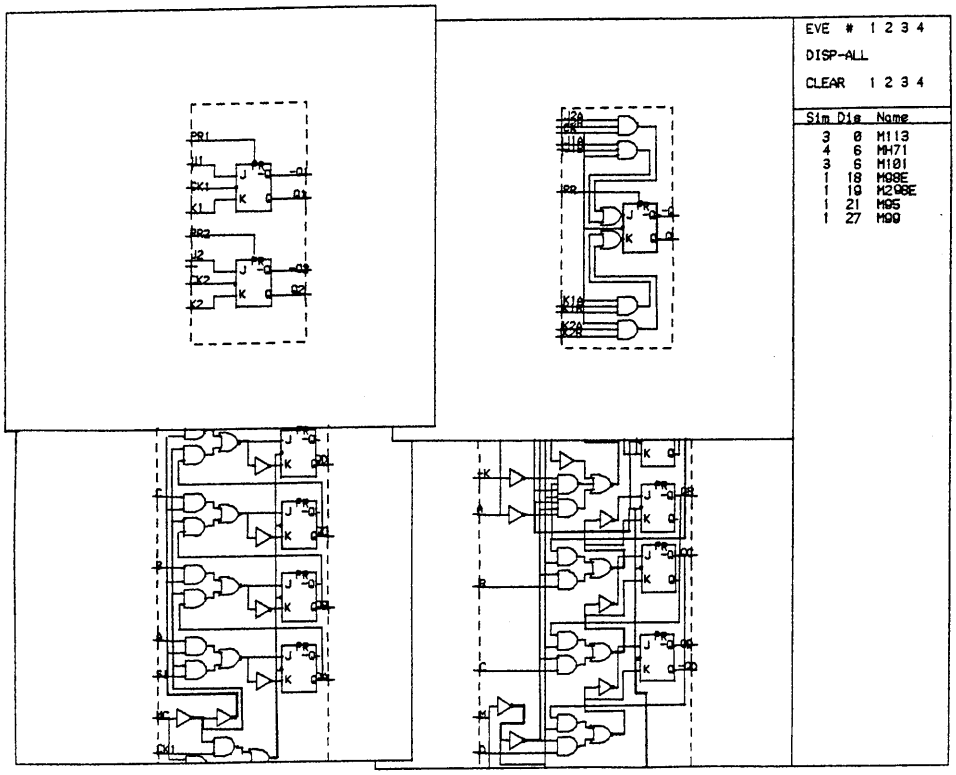

図15．類似回路の検索例

ただし，X，Yは，それぞれ，ルアー回路図の部分回路の集合，候補回路図の部分回路の集合である。従来の類似検索が，オブジェクトの類似度にのみ着目していたのに対し，この手法は類似度に加え相違度も扱っていることが特色である。

続いて，候補回路図をユーザにとって役立ちそうな順序に並べ変えなければならない。この順序は，ユーザの嗜好にも依存するものであるから，一概には決まらないものであるが，実現が簡単で，しかも効果的であるルールとして

(1) 相違度の少ないものが良く

(2) 同じ相違度ならば類似度が大きいものが良い

ということが経験的にわかった。今回の実現では，まず相違度でソートし，次に類似度でソートして候補回路図の順序を並べ換えている。

5.4 実現結果

図15は，1個のプリセット付きフリップ・フロップをルアー回路図として類似検索をした結果の表示画面である。利用者は，類似度，相違度を参考にしながら類似回路候補を選択し，画面に表示し，図面の内容を確認することができる。図15上側の2枚の図面は，データベースが“より適切である”と判断した回路図で，下側は，“より適切でない”と判断した回路図である。

6. おわりに

事務処理をはじめとする従来のデータベース応用では，一つのレコード（タプル）に意味があることが多かった。他方，ＣＡＤなどでは，一連のレコードの集まりが意味のある情報単位（オブジェクト）となることが多い。ここで，複雑な構造を持つオブジェクトをどのようにモデル化し，更新の一貫性を維持し，検索し，再利用するか，ということが問題となる。

本文では，回路図を例にしながら，上記の問題に対する解決案を論じた。要約すると次のようになる。

(1) 構造を持つデータのモデル化手段として，データと手続をパッケージ化する，オブジェクト指向のアプローチを述べた。

(2) 更新の一貫性に関しては，トランザクションのネストによる対応法を述べた。

(3) 検索・再利用については，類似度と相違度に基づく検索法を示した。

参考文献

[1] Batory,D.S. and Kim.W : Modeling concepts for VLSI CAD objects, ACM TODS Vol.10,No.3, 1985,pp.322-346.

[2] Chen,P.P. : The entity-relationship model towards a unified view of data, ACM Transactions on Darabase Systems, Vol.1, No.1,1976.

[3] Codd,E.F.: Extending the database relational model..., ACM TODS, Vol.4,No.4, 1979, pp397-434.

[4] Date,C.J. : An introduction to database systems , Volume 1 , 4th ed. Addison-Wesley , 1987.

[5] Haskin,R. and Lorie,R. : On extending the functions of a RDB system,Proc. ACM SIGMOD 1982,pp207-212.

[6] Katz,R.H. : Information management for engineering design,(Surveys in Computer Science), Springer-Verlag,1985.

[7] McLeod,J. : Now, a way to do on-the-spot checks of IC design changes , Electronics, November 12, 1987, pp.105-106.

[8] Weber,S, : CAE industry gropes for a tool-integration strategy, Electronics,June 1988, pp.118-119.

[9] Wells,D.L.(Chair) : PANEL : Object-oriented database --- evolution or revolution, Proc. 5th Int. Conf. on DATA ENGINEERING, IEEE, 1989, pp.417-423.

[10]植村：データベース・システムの基礎，オーム社，1979.

[11]酒井：情報資源管理の技法，オーム社，1987.

[12]穂鷹：データベース要論，共立出版，1978.

[13]宇田川，溝口：回路図の編集と拡張関係データベースADAMのデータ更新処理，情報処理学会第34回全国大会予稿集，pp.515-516，1987.

[14]宇田川，辻，市川：オブジェクト指向リレーショナル・データベースADAMにおける類似検索機能，情報処理学会“アドバンストデータベースシステム”シンポジウム予稿集，1988，pp.33-42.