データベースシステム 109－10
（1996. 7. 24）

ビューを活用した
複合オブジェクトの一貫性検査

掛下 哲郎　　加島 久美子
佐賀大学 理工学部 情報科学科

〒840 佐賀市本庄町1番地、Tel. (0952) 28-8565、E-mail: kake@is.saga-u.ac.jp

共有DBにおいて、DB一貫性の保持は極めて重要である。本論文では、OODBにおける一貫性検査のための手法を提案する。提案手法は著者らが提案しているビュー機能を活用しており、以下の特徴を持つ。(1) DBが必ず満たさなければならない「固い制約」として、参照構造、クラス、メソッドなどに関する制約を柔軟に検査できる。(2) DBが可能な限り満たすことが要望されている「柔らかい制約」についての検査を行なえる。(3) 一貫性検査によって発見されたOODBの矛盾を、ビュー機能を用いてGUIベースで修正できる。本論文では、各種の固い制約について、制約を満足しないインスタンスを検索するビューを定義する。また時間割DBを例に用いて、各種の柔らかい制約を検査するための二次元ビュー表を示す。

キーワード：オブジェクト指向データベース、一貫性検査、柔らかい制約、ビュー

Integrity Checking of Complex Object using Multiple Views

Tetsuro Kakeshita　　Kumiko Kashima
Department of Information Science, Saga University

Saga 840, Japan、Tel. +81-952-28-8565、E-mail: kake@is.saga-u.ac.jp

Database integrity must be preserved in a shared database. This paper proposes an integrity checking method for OODB. The method is based on the view function which the authors have proposed in the preceding works and has the following advantages. (1) It can conveniently check various required constraints, concerning referential structure, class, method, etc. (2) It can also check optional constraints, which a DB is required to satisfy as much as possible. This is performed by defining a table of views. (3) The view function allows GUI based updates to the contradictory instances.

Keywords : Object oriented database, Integrity checking, Optional constraints, View

1 まえがき

種々の利用者や各種の応用プログラムによるデータの共有は、データベースの本質的な特徴である。データベース研究は、このために重要と考えられる種々の技術を提案しているが、その中で、データ一貫性の保持は古くから研究されている重要なテーマの一つである。DB技術に対する要求の高まりを受けて、オブジェクト指向モデルや意味的データモデルを始めとする各種の高度なデータモデルが提案されているため、一貫性保持の問題はますます複雑化している。本論文では、著者らが提案しているビュー機能を活用して、オブジェクト指向データベース (OODB) における各種の一貫性検査を行なうための手法を提案する。

ビュー機能は、ビューとビュー間操作からなる [1, 2]。ビューは、指定された選択条件を満足する複合オブジェクトを検索して、指定された属性を表示できる。またビュー間操作は、一つのビュー上に表示されている複合オブジェクトを、その複合オブジェクトが満足しない選択条件を持つ他のビュー上に移動[1]することで、新しいビューの選択条件を満足するように、移動された複合オブジェクトを宣言的に編集できる。ビュー機能には (1) 複雑なデータベースを宣言的に検索/操作できることの他に、(2) OODBだけでなくRDBやE-RモデルDBなども操作できる、(3) GUIに適する、(4) OQL[3] 等とは異なり、インスタンス指向の操作と集合指向の操作の双方をサポートするといった利点がある。

一貫性制約は、全てのインスタンスが必ず満たさなければならない「固い制約」と、可能な限り満たすことが要望されている「柔らかい制約」の二種類に大別できる。

関係DBの世界では、従属性制約を始めとする各種の一貫性制約を検査するために、トリガが提案されてきた。また、OODBにもトリガの概念が導入され [4]、それを一般化したアクティブデータベース [5] の概念も一貫性管理に使用できる。しかし、これらの技術によって検査できる一貫性制約は固い制約のみであり、現実のデータベースシステムに適用する場合には柔軟性に欠ける。

一方、柔らかい制約は必ずしも満足させることが要求されておらず、互いに矛盾する場合もある。また、固い制約としては記述困難な制約を柔らかい制約を用いて近似的に表現する場合もある。柔らかい制約に関する研究は人工知能、オペレーションズリサーチ、知識ベース等の分野で行なわれている。佐藤は、柔らかい制約を複数の制約条件の間の優先度として定式化した [6]。また、新谷らの研究では、制約条件間の優先度を矛盾しないように決定する方法を論じている [7]。また、スケジューリング問題を解く際の制約条件として柔らかい制約を導入した研究 [8] 等もある。これらの研究は全ての柔らかい制約が論理式等を用いて定式化可能であることを前提としている。しかし実際には、定式化が容易でない制約や、厳密な定式化を行なうと記述量が増大するため望ましくない制約 (例：時間割は、各教官の都合にあっていることが望ましい) なども存在する。

本論文では、一貫性制約を満足しない複合オブジェクトをビュー上で検索することで固い制約の検査を行なう。これによって、制約を満足しないインスタンス変更や追加は拒否される。ビューの定義を若干拡張することでアクティブデータベースにおけるECAルール等も記述できるため、制約記述能力は高い。また、柔らかい制約を検査する主体をあくまでも利用者 (DB管理者を含む) とし、検査を容易に行なうために必要な機能をビューによって提供するアプローチを提案する。複数のビューを用いてインスタンスを二次元の表形式に分類表示することにより、各種の検査を容易に行なえ、実現も容易になる。さらに、ビュー間操作を用いることで、問題のあるインスタンスを直ちに編集できる。

本論文は、以下のように構成されている。2節ではOODBを表現するための基本的なデータモデルを導入して、その上でビューを定義する。3節では、OODBに関して知られている各種の固い制約を分類し、それらを自動的にチェックするためのビューを定義する。柔らかい制約に対しては、4節で時間割DBを用いて制約の例を挙げる。列挙された制約について一貫性検査を行なうためのビュー表を定義する。

2 ビューの定義

2.1 基本データモデル

基本データモデルのスキーマSはクラスNの有限集合である。クラスの構成要素は属性、リンクおよびメソッドである。属性Aは基本データ型 (整数、文字列等) であり表示可能である。これに対してリンクLは自クラスまたは他クラスを参照するための型であり表示できない。また、メソッドMは引数リスト $(p_1, p_2, \cdots)$ を持つ。メソッドの引数および戻り値は、属性またはリンクとする。Sに含まれる属性の集合を$\boldsymbol{A}$、リンクの集合を$\boldsymbol{L}$、メソッドの集合を$\boldsymbol{M}$と定義する。クラスNは組クラスN_T、集合クラスN_I、選択クラスN_Sのいずれかである。各クラスはその種類に応じて以下に定義する構造を持つ。

[1] この他に追加、コピー、生成、削除操作がある。

$$N_T = \langle \boldsymbol{A}', \boldsymbol{L}', \boldsymbol{M}' \rangle \quad \boldsymbol{A}' \subseteq \boldsymbol{A}, \boldsymbol{L}' \subseteq \boldsymbol{L}, \boldsymbol{M}' \subseteq \boldsymbol{M}$$
$$N_I = \langle L \rangle \quad L \in \boldsymbol{L}$$
$$N_S = \langle \boldsymbol{L}' \rangle \quad \boldsymbol{L}' \subseteq \boldsymbol{L}$$

各クラスは、複数のインスタンスを持つことができる。各インスタンスは固有の識別子を持つ。属性 A の値は A の型に属する値 $a_i (i = 0, 1, \cdots)$、リンク L の値は L が指すクラスのインスタンスの識別子の値 $l_j (j = 0, 1, \cdots)$ である。各クラス Nのインスタンス I_Nはクラスの種類に応じて次のように定義される。

I_{N_T} = $\langle \boldsymbol{a}', \boldsymbol{l}', \boldsymbol{M}' \rangle$
$\boldsymbol{a}'$は $\boldsymbol{A}'$に属する各属性の値からなる組
$\boldsymbol{l}'$は $\boldsymbol{L}'$に属する各リンクの値からなる組

I_{N_I} = $\langle \{\mathbf{NIL}\} \rangle$ または $\langle \{l^1, l^2, \cdots\} \rangle$
l^pは NIL でないリンク L の値

I_{N_S} = $\langle l \rangle$
lは $\boldsymbol{L}'$に属するいずれかのリンクの値

クラス Nのインスタンス I_Nの属性 (またはリンク)Xの値を $I.X$で表す。また、引数 $q_1, q_2, \cdots$を与えた場合のメソッド m の戻り値を $I.m(q_1, q_2, \cdots)$ で表す。クラス Nは (一般に複数の) サブクラス N' を持つことができるが、N'のインスタンスは Nのインスタンスでもある。スキーマ Sのインスタンスは Sに属する全てのクラスのインスタンス全体からなる集合である。また、Sに属するクラスのインスタンスからなる集合を複合オブジェクトと定義する。

図 1に基本データモデルにおける時間割 DB のスキーマ例を示す。

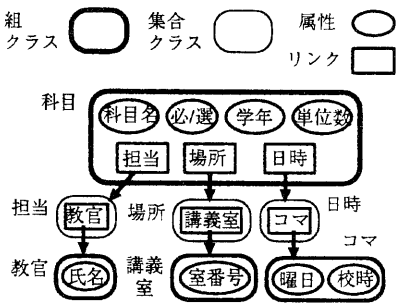

図 1: 基本データモデルのスキーマ例

2.2 ビュー

本節ではビューを定義する。これに先立ち領域変数を定義する。クラス Nに対し、領域変数 $D^1_N, D^2_N, \cdots$ を割り当てることができる。D^i_N は Nの任意のインスタンス I^j_N[2]にマッチングできる。D^i_Nが I^j_Nにマッチングした時、N の任意の属性/リンク Xおよびメソッド m について

$$\begin{aligned} D^i_N &= I^j_N \\ D^i_N.X &= I^j_N.X \\ D^i_N.m(q_1, q_2, \cdots) &= I^i_N.m(q_1, q_2, \cdots) \end{aligned}$$

が成立する。また、メソッド m について以下の式が成立する。

$$D^i_N.m = \begin{cases} True & m \text{ が呼び出された時} \\ False & m \text{ が呼び出されていない時} \end{cases}$$

ビュー Vは射影属性と選択条件の組によって定義される。

射影属性 $PA(V)$ は、以下に列挙する要素を含む集合であり、ビュー上に表示する値を指定する。

1. 属性 $D^i_N.A$
2. メソッドの戻り値 $D^i_N.m(q_1, q_2, \cdots)$
3. D^i_Nにマッチングするインスタンスの数を返す関数 $\mathbf{Count}(D^i_N)$
4. D^i_N, D^j_Mにマッチングするインスタンスから構成される組の数 $\mathbf{Count}(D^i_N, D^j_M)$
5. $D^i_N.A$ の合計値 $\mathbf{Sum}(D^i_N.A)$

選択条件 $SC(V)$ は以下の 1〜3 で定義する論理式であり、複合オブジェクトを検索するために用いる。また、$SC(V)$ に含まれる領域変数の集合を $DV(V)$ と定義する[3]。

1. 項$_1$ θ 項$_2$ は論理式である。但し、項$_i (i = 1, 2)$ は領域変数 D^i_N、属性 $D^i_N.A$ 、リンク $D^i_N.L$、メソッドの戻り値 $D^i_N.m(q_1, q_2, \cdots)$、定数 c、'undefined' のいずれかである。また、θは '=', '≠', '<', '>', '≤', '≥', '⊂', '⊃', '⊆', '⊇', '∋', '∌' のいずれかである。項$_1$と項$_2$の間には、比較演算子θが定義されていなければならない。
2. 'true(D^i_N)' は選択条件である。
3. 式 F_1と F_2が選択条件ならば、$(F_1 \wedge F_2)$、$(F_1 \vee F_2)$、$\neg F_1$は選択条件である。

選択条件 $D_N.A\ \theta\ c$ (θは '=', '≠', '<', '>', '≤', '≥' のいずれか)、および $D_N.X\ \theta\ D'_{N'}.X'$ (θは '⊂', '⊃', '⊆', '⊇' のいずれか) は、$D_N, D'_{N'}$がマッチングしたインスタンス $I_N, I_{N'}$の属性等の値や、リンク値の集合について θ で指定された比較が成功した場合に真となる。$D_N.L\ \theta\ D_{N'}$ (θは '=', '≠', '∋', '∌' のいずれか) は、D_Nがマッチングしたインスタンス I_Nが、$D_{N'}$がマッチングしたインスタンス $I_{N'}$を参照し

[2] Nのサブクラスのインスタンスを含む。
[3] $PA(V)$ に属する領域変数は、常に $SC(V)$ 中に出現する。

ている(参照していない)場合に真となる。$D_N\ \theta\ D'_N$ (θ は '=', '$\neq$' のいずれか) は、D_N, D'_NがマッチングしたインスタンスI_N, I'_Nが同一である(互いに異なる)場合に真となる。$D_N.X$='undefined' は D_N がマッチングしたインスタンスI_Nについて、Xが属性ならば$I_N.X = \mathbf{NULL}$の場合に真となり、Xがリンクならば$I_N.X = \mathbf{NIL}$ (Nが集合クラスの場合には$I_N.X = \{\mathbf{NIL}\}$) の場合に真となる。'true(D^i_N)' は、D^i_Nにマッチングするインスタンスに対して常に真となる。

ビュー Vの選択条件 $SC(V)$ を真にするように $DV(V)$ に割り当てられたインスタンスの集合を、Vに属する複合オブジェクトとよぶ。Vに属する各々の複合オブジェクトに対して、$PA(V)$ で指定される属性値(メソッドの戻り値を含む)を射影することによって構成される組をビューVに属する組と定義する。ビューVは、Vに属するすべての組を表示する。この際に同一の値を持つ組は重複して表示しない。また、同一のインスタンスが複数の組に属する場合がある。ビューは DBMS によって識別子を割り当てられ、必要に応じて固有のビュー名を持つ。

二つのビュー V_1と V_2について、表1に示す集合操作を定義する。

$V_1 \cap V_2$	V_1と V_2の双方に属する組を表示するビュー
$V_1 \cup V_2$	V_1と V_2のいずれかに属する組を表示するビュー
$V_1 - V_2$	V_1に属し、V_2に属さない組を表示するビュー

表 1: 複数ビューの集合演算

検索対象の複合オブジェクトを構成する各インスタンスにマッチングする領域変数の集合を定義し、各々の領域変数に対して必要な選択条件(属性値等の条件やインスタンス間の参照関係など)を記述することによって、任意の複合オブジェクトを検索できる。

3 固い制約の検査

文献 [9, 10, 11] に基づいて OODB における固い制約を調査した結果、下記のように分類された。(1) 属性値に関する制約, (2) 参照構造に関する制約, (3) 集合内の要素に関する制約, (4) 継承に関する制約, (5) メソッドに関する制約。本節では、この分類に従って一貫性制約を満足しないインスタンス等を検索するためのビューを示す。

これらのビューは比較的複雑な定義になっているが、DB 管理者 (DB 設計者を含む) が前もって定義しておくため、一般の DB 利用者が自らビュー定義を行なう必要はない。また、ビューに属する (一貫性制約を満足しない) インスタンスが発生した際にだけ表示してビュー間操作による編集を可能にすれば十分なため、不必要に画面を専有しない。

3.1 個別インスタンスに関する制約

個別インスタンスに関する制約の代表例は、関係 DB における従属性制約や組内制約条件、及び空値パターンを制限する対象集合制約などである。このうち OODB においては、従属性制約に基づくキー制約と空値に関する制約が主なものである。

キー 指定された属性について、同一値を持つインスタンスが複数存在しない。

空値 指定された属性に対する空値の入力は許されない。

クラス C の属性 A についてキー制約を検査するためには、以下のビュー V_1を定義して、異なるインスタンスと等しい属性 A の値を持つインスタンスを検索すれば良い。

$$DV(V_1) = \{D^1_C, D^2_C\colon \mathbf{C}\}$$
$$SC(V_1) = (D^1_C.\mathbf{A} = D^2_C.\mathbf{A})$$
$$PA(V_1) = D^1_C.\mathbf{A}$$

また、空値の検査を行なう場合には指定された属性の値が Null であるようなインスタンスを検索すれば良い。なお対象集合制約等のより複雑な制約の場合には、選択条件に許容されない空値パターンを記述する必要がある。

3.2 参照構造に関する制約

OODB の一つの特徴は、リンクを活用した複雑なデータ構造の表現にある。このようなリンク構造に関する一貫性制約としては以下のものが挙げられる。

参照一貫性 クラス N のインスタンスがクラス M のインスタンスから参照されているならば、必ず逆方向にも参照が存在する。

一対一/一対多/一対 n 対応 クラス N のインスタンスとクラス M のインスタンスが一対一 (一対多、一対 n) 対応している。なお、n は定数である。

上への対応 クラス N の全てのインスタンスについて、クラス M のインスタンスが対応する。

木/鎖構造 クラス N 内のインスタンス間の参照構造が木構造 (または鎖構造) である。

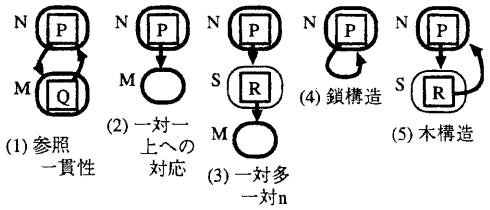

図 2: 参照構造の検査対象スキーマ

これらの制約を検査するためのビューを以下に示す[4]。なお、スキーマは図 2に示す。クラス Nに対応する領域変数は D_N^iのように示す。

[参照一貫性] N → M 方向のリンクが存在して、M → N 方向のリンクが存在しないインスタンスを検索することで検査できる。

$SC(V_2) = ((D_N.\mathbf{P} = D_M) \wedge (D_M.\mathbf{Q} \neq D_N))$
$PA(V_2) = \{\ D_N, D_M\ \}$

[上への対応] クラス M の全てのインスタンスから、N のインスタンスによって参照されているものを除外することで検査できる。この際にビュー間の差演算を用いる。

$SC(V_3) = \mathbf{true}(D_M) \qquad PA(V_3) = \{\ D_M\ \}$
$SC(V_4) = (D_N.\mathbf{P} = D_M) \quad PA(V_4) = \{\ D_M\ \}$
$SC(V_5) = V_4 - V_3$

[木構造] 複数の親節点を持つ子節点の検索 (V_6) 及び、節点数 (V_7) と枝数 (V_8) の比較による連結性の検査により検査できる。連結性の検査を行なうためには、ビュー間の算術比較演算が必要である。

$SC(V_6) = ((D_N^1.\mathbf{P} = D_S^1) \wedge (D_S^1.\mathbf{R} = D_N) \wedge (D_N^2.\mathbf{P} = D_S^2) \wedge (D_S^2.\mathbf{R} = D_N))$
$PA(V_6) = \{\ D_N^1, D_N^2\ \}$
$SC(V_7) = \mathbf{true}(D_N) \quad PA(V_7) = \mathbf{Count}(D_N)$
$SC(V_8) = ((D_N^1.\mathbf{P} = D_S^1) \wedge (D_S^1.\mathbf{R} = D_N))$
$PA(V_8) = \mathbf{Count}(D_N^1, D_N)$

これらの制約の他に、経路存在従属性 [12] などの制約も提案されているが、参照一貫性を検査するためのビューを拡張して定義できる。**is-part-of** 関連や **is-member-of** 関連なども知られているが、空値制約 (3.1 節) と集合に関する制約 (3.3 節) によって記述できる。

しかし、ビュー定義及びビュー間での集合演算や算術比較演算だけでは判定できない一貫性制約も存在する。その代表例としては、一般のグラフ構造における各種制約 (連結性、非巡回性、彩色可能性など) のように定型化されていないアルゴリズムによる検査が必要なものが挙げられる。このような制約を検査するためには、検査アルゴリズムをメソッド化して、それを実行する必要がある。

[4] スペースの都合で、一対 x 対応及び鎖構造を検査するビューは省略する。詳細は、文献 [14] を参照のこと。

3.3 集合に関する制約

ODMG を始めとする各種の OODB では、数学的な集合の他に **bag**(重複つき集合), リストなどをサポートしている。これらの集合類に関する制約としては以下のものがある。

順序制約 集合内の要素は、指定された順序に従って格納されている。

要素数制約 集合内の要素数は、指定された上限と下限を守らなければならない。

このうち、順序制約は、隣接する 2 つの要素 e, e' をリンクによって関連付けておくことで、組 (e, e') を検索できる。さらに、順序を属性値によって定義する場合には、定義を満たさない組を容易に検索できる。なお、より複雑な順序定義は、メソッド呼出しを用いて検査できる。

また、要素数制約は **Count** 関数を用いることで容易に検査できる。

3.4 継承に関する制約

継承に関する制約としては以下のものが挙げられる。

結び制約 クラス A のインスタンスは、A のいずれかのサブクラスのインスタンスである。

相互排他制約 クラス A の複数のサブクラスのインスタンスになるものは存在しない。

分割制約 結び制約と相互排他制約が同時に成立する。

クラスに対応する領域変数には、サブクラスのインスタンスもマッチングする。この性質を用いると、それぞれのクラスとサブクラスに属するインスタンスを個別のビュー上で検索しておき、ビュー間の集合演算を用いることで上記の制約は容易に検査できる。

3.5 メソッドに関する制約

OODB の重要な特徴はメソッドであるが、副作用を持つメソッドは、実行の際に一貫性制約を満足しなくなる可能性がある。これを防ぐために、以下の一貫性制約が通常用いられる[5]。

[5] オブジェクト指向設計においては、各クラスの正当性を検査するために状態遷移図を用いることが多い [13]。この状態遷移図も当該クラスの可能な状態について、クラス内のメソッドを実行した後の状態を検査するものであり、前条件と後条件の検査に帰着できる。

前条件と後条件 指定されたメソッドを実行する際には、前条件を満たさなければならない。また、実行後には後条件を満たさなければならない。

値変更制約 指定された属性値を変更する際、変更前後の値が特定の関係を満足しなければならない[6]。

インスタンスの状態(属性及びリンク)がメソッド実行によって変更されるため、変更前後のインスタンスにマッチングする領域変数 D^B, D^A を別に定義して、前条件と後条件をそれぞれ記述する。また、二つの領域変数を関連付けるために関数 old を定義する。従って、以下の式が成立する。

$$D^B = \mathrm{old}(D^A)$$

これによって、メソッドに関する制約を検査できる。

また、メソッドを用いた一貫性制約の検査を行なうために、RDB におけるトリガやアクティブ DB における ECA ルール [5] の概念が提案されている。このうち OODB にも適する ECA ルールはビューによって以下のように記述できる。

イベント及び条件節はビューの選択条件として記述する。このうちイベントは、メソッドの起動と考えてよい。クラス N に対応する領域変数を D_N とすると、N のメソッド m が起動された時には、$D_N.m$ が真となる。これによって、イベントが発生した時だけ真になる選択条件を記述できる。また、トリガアクションはビューの射影属性によって記述する。アクションをメソッドに記述しておき、射影属性でこのメソッドを呼出すことによってアクションを起動できる。

上記の方法でビューを定義すれば、メソッド実行を必要とする一貫性制約も検査できる。

4 柔らかい制約の検査

固い制約は DB 中で必ず成立することを義務づけられているのに対して、柔らかい制約は可能な限り満たすことを要望されている制約である。従って、複数の柔らかい制約は互いに矛盾する場合もある。また、固い制約としては記述困難な制約を柔らかい制約を用いて近似的に表現する場合もある。現実の世界においては、固い制約よりも柔らかい制約の方がはるかに種類が多い。

柔らかい制約は極めて多様なため、パターンを定型化することは困難である。そこで、柔らかい制約の検査は基本的に利用者(または DB 管理者)が行なうこととし、ビュー機能は利用者のチェック作業を支援するために、DB 内のデータを目的に応じて検索し、複数のビューを用いて分類表示する機能を提供する。利用者が検査項目に都合の良い一覧表示形式を使用することで、一貫しない可能性のあるデータを容易に探すことができる。また、ビュー間操作を用いることで、矛盾すると判断されたデータを直ちに修正することもできる。

本節では、時間割 DB の例を用いて柔らかい制約を例示し、それらを検査するためのビューを定義する。

4.1 時間割 DB における柔らかい制約

時間割 DB における柔らかい制約の例としては、「各教室では、同一校時に複数科目を開講できない」といった制約が挙げられる。この制約は原則的には守られるべきであるが、いくつかの例外が許されている。例えば、(1) 隔週に開講される講義、(2) 講義担当者間の話し合いで講義室の使用スケジュールを重複しないように決める場合、(3) 複数科目の合併授業などである。これと同様に、時間割 DB において考えられる制約を以下に挙げる。これらの制約の間には優先度の違いも存在することに注意されたい。

1. 同学年の必修科目は、他の科目 (必修/選択) と同一校時に開講できない。
2. 各教官は、同一校時に複数科目を開講できない。
3. 全ての科目は時間割中にある方が良い。
4. 卒業単位を満たすために最低限必要な科目は開講しなければならない。
5. 異なる学年の必修科目でも、同一校時に開講しない方が良い。
6. 同一学年の講義は連続したコマで開講した方が良い。
7. 各学年について、特定曜日の講義負担が過重にならない方が良い。
8. 指定された校時には、授業を開講できない。
9. 科目の開講校時は、担当教官の都合に合っている方が良い。

柔らかい制約に関する判断を行なうための例外情報は極めて多岐に渡るため、DB に入力されないことも多い。また、柔らかい制約の間には優先度があるため、DB に入力されていた場合でも、場合によって異なる取り扱いが必要になり、全ての制約及びそれらの間の優先度を記述する作業は膨大なものになる。このような作業は、定型的かつ膨大なデータ量を持ち、頻繁に使用される DB アプリケーションにおいてのみ意義を持つ。しかし、最近重要性を増している高度 DB 上での作業は非定型的なものが多いため、上記の仮定は成立しない場合が多い。

[6] この制約は、個別インスタンスに関する制約だが、検査方法がメソッドに関する制約と同一なので、本節に分類した。

	1	2	3	4	5
月	人工知能	システム理論	数値解析		
火			実験I, 実験II	実験I, 実験II	
水		計算の理論	データベース		
木			実験I, 実験II	実験I, 実験II	
金		OS			

表 2: 二次元ビュー表を用いた時間割 DB の制約検査例

4.2 二次元ビュー表を用いた柔らかい制約の検査

上記で列挙した各種の柔らかい制約を検査するために、複数のビューを二次元的に配列したビュー表の使用を提案する。ビュー表上では、DB の各種インスタンスが分類表示されるため、利用者がこれを調べることで容易に柔らかい制約を満たしていないインスタンスを発見できる。二次元ビュー表の行と列は利用者によって任意に設定できるため、柔軟な DB 検査を行なえる。複数の二次元ビュー表を表示することで、複数の柔らかい制約を同時に検査することもできる。利用者が発見したインスタンスについては確認や他の制約との間での価値判断等を行ない、修正すべきならばビュー間操作を用いて GUI ベースで編集を行なえる。一つのビュー表上での編集結果は他のビュー表へも伝搬されるため、編集の結果を直ちに確認できる利点もある。

例　特定の学年に関する柔らかい制約として挙げられている「同一学年の講義は連続したコマで開講した方が良い」、「各学年について、特定曜日の講義負担が過重にならない方が良い」を検査するためのビュー表を表 2に示す (3 年生の例)。

ビュー表は、以下に定義するビュー V_{ij}の配列である。Day は月から金、$Hour$ は 1 から 5 のいずれかである。

$$
\begin{aligned}
DV(V_{ij}) &= \{D_C\text{: 科目}, D_D\text{: 日時}, D_H\text{: コマ}\} \\
SC(V_{ij}) &= (D_C.\text{日時} = D_D) \wedge (D_D.\text{コマ} = D_H) \\
&\quad \wedge (D_H.\text{曜日} = Day) \\
&\quad \wedge (D_H.\text{校時} = Hour) \wedge (D_C.\text{学年} = 3) \\
PA(V_{ij}) &= \{D_C.\text{科目名}\}
\end{aligned}
$$

この表により、同一曜日に割り当てられている負荷の重い科目、連続しない校時に割り当てられている科目、1 科目だけが開講されている曜日などを容易にチェックできる。ビュー表に属するビューの定義はほとんど共通なので、一つだけを完全に定義すれば、残りのビューはビュー定義のコピーと若干の編集によって簡単に定義できる。また、科目をビュー表内でドラッグすればビュー間での移動操作を実行できるため、曜日と校時を容易に変更できる。 □

上記の例では、特定の学年に関する柔らかい制約の検査例を挙げたが、教官に関する制約や、講義室に関する制約、必修科目に関する制約も表 3に示す選択条件を各ビューに設定することで検査のためのビュー表を定義できる。なお、選択条件中の領域変数は、それぞれ、以下のクラスに対応している。

D_C 科目	D_D 日時	D_H コマ
D_I 担当	D_T 教官	D_P 場所
D_R 講義室		

教官に関する制約においては、教官毎に $Name$ の値を設定してビュー表を定義する。また、講義室に関する制約では、室番号毎に $RoomNo$ の値を設定してビュー表を定義する。

4.1 節で挙げた制約としては、この他に単位数に関する制約が挙げられるが、これは、曜日と校時を割り当てられている科目を検索して、単位数の合計を求めることによって検査できる。単位数の合計は、Sum 関数によって求められる。

本論文ではスペースの都合で記述を省略するが、病院での看護婦の勤務スケジュールに関する DB を構築した場合の各種の柔らかい一貫性検査もビュー表によって実現できる [14]。

5 むすび

本論文では、種々の一貫性制約を「固い制約」と「柔らかい制約」に大別して、それぞれをチェックできるビューを定式化した。ビュー間での集合演算、算術比較演算を定義し、Count 関数, Sum 関数, old 関数を導入することで柔軟な制約検査が行なえる。これらの演算や関数の実現は容易であるため、本方式は実装の際にも優れている。

データベースの一貫性保持機能に対する要求はやむことがない。しかし全ての一貫性制約を固い制約として保持する方式だけでは、実現はもとより記述も困難な場合に直面することが予想される。それに

	選択条件
教官に関する制約	$(D_C.\text{日時} = D_D) \wedge (D_D.\text{コマ} = D_H) \wedge (D_H.\text{曜日} = Day) \wedge (D_H.\text{校時} = Hour)$ $\wedge (D_C.\text{担当} = D_I) \wedge (D_I.\text{教官} = D_T) \wedge (D_T.\text{名前} = Name)$
講義室に関する制約	$(D_C.\text{日時} = D_D) \wedge (D_D.\text{コマ} = D_H) \wedge (D_H.\text{曜日} = Day) \wedge (D_H.\text{校時} = Hour)$ $\wedge (D_C.\text{場所} = D_P) \wedge (D_P.\text{講義室} = D_R) \wedge (D_T.\text{室番号} = RoomNo)$
必修科目に関する制約	$(D_C.\text{日時} = D_D) \wedge (D_D.\text{コマ} = D_H) \wedge (D_H.\text{曜日} = Day) \wedge (D_H.\text{校時} = Hour)$ $\wedge (D_C.\text{必/選} = \text{必修})$

表 3: 各種制約を検査するためのビュー表

対して「柔らかい制約」の検査法は、利用者自身が一貫性検査を行なうことを前提として、利用者の作業を軽減するための方策に力点を置いており、様々な要求に柔軟に応えられる。

ソフトウエアにおけるテストやデバッグ作業は定型的な作業ではないため、ソフトウエアツールはバグ発見の自動化を目指すのではなく、利用者の支援に徹する。本論文のアプローチは、**DB** をソフトウエアと同一視すると上記の考え方とも共通している。従来のデータベース研究において、このような考え方が取り上げられることは少なかったが、データベース自身の複雑さもソフトウエアと同様に急速な増大傾向にあることを考慮すると、今後の一貫性管理にとって、本アプローチは重要な考え方になると著者は考えている。

本研究は、ビュー機能を用いた宣言的なデータベース操作に関する研究の一環として行なったものである。今後の課題としては、ビュー機能の利用者インタフェース [15] に基づいたプロトタイプの実装、選択条件を満足する複合オブジェクトの高速検索などが挙げられる。

参考文献

[1] 掛下、村田、“ビュー機能を用いた E-R モデルデータベースの操作”, 情報処理学会 DBS 研究会 101-5, 1995.

[2] 村田、“宣言的なデータベース操作のためのビュー機能”、佐賀大学情報科学科 修士論文, 1996.

[3] R. G. G. Cattell 編集、河込 他訳,“オブジェクト・データベース標準: ODMG-93”, 共立出版

[4] 石川、酒井、“オブジェクト指向データベースにおける制約管理について”, 情報処理学会 DBS 研究会 89-10, 1992.

[5] 石川、“アクティブデータベース”, 情報処理, Vol. 35, No.2, pp.120–129, 1994.

[6] 佐藤、“解釈の順序による柔らかい制約の定式化”, 情報処理学会論文誌, Vol. 31, No. 6, pp. 772–782, 1990.

[7] 新谷、“デフォルト制約に基づく主観的評価の一貫性保持について”, コンピュータソフトウエア, Vol. 8, No. 4, pp. 45–55, 1991.

[8] 北野, “遺伝的アルゴリズム・2”, 産業図書, pp.89–125, 1995.

[9] エンブレイ、“オブジェクト指向システム分析”, 東京電機大学出版局、1993.

[10] Khoshafian, S., “オブジェクト指向データベース”, Bit 別冊, 共立出版, 1995.

[11] 上林、“データベース”, 第 3 章, 昭晃堂, 1986.

[12] 神谷 他, “オブジェクト指向データベースにおける経路存在従属性と属性キー制約”, Proc. ADBS'93, pp. 73–82, 1993.

[13] Rumbaugh 他著、羽生田監訳、“オブジェクト指向方法論 OMT”、凸版印刷, 1991.

[14] 加島、“ビュー機能を用いたデータベース一貫性の検査”, 佐賀大学情報科学科 卒業論文, 1996.

[15] 村田、掛下、“宣言的なデータベース操作のためのビュー機能のインターフェース”、, 情報処理学会 DBS 研究会 109-8, 1996.