オブジェクト指向データベースにおけるバージョン管理モデルの設計と実装

吉沢直美† 小野美由紀† 石川博†

設計分野で必要とされる試行錯誤を支援するバージョン管理モデル（V model）の設計と実装を行った．このモデルはOODB上においてオブジェクトを対象とするバージョン管理機能を，拡張性を考慮して後天的かつ適正な性能で提供するものであり，管理情報の適用対象からの独立性，機能の適用対象オブジェクトと非適用対象の混在，複数のオブジェクト間における異なるバージョンの組合せを利用すること（what-if型の試行），クラス，インスタンスの双方をバージョン管理機能の適用対象とすること，等を特長とする．なお性能に関しても，機能提供の後天性や機能仕様の拡張性を特長とするDBシステムの仕様に依存しないモデルである点を考慮すると，妥当なものが得られたと考える．今後は，ここで提案するモデルを用いた場合のオブジェクト整合性の改良および，より下層において機能を実現することによる時間性能を重視した仕様への変換を予定している．

Design and Implementation of a Version Control Model for Object-Oriented Databases

YOSHIZAWA NAOMI,† ONO MIYUKI† and ISHIKAWA HIROSHI†

We have designed and implemented a version control model for supporting user experimentation in design using Object-Oriented Databases. This model manage version information as independently on data objects, and allows the user to add versioning functions to existing databases and offers the version control functions in consideration of the extendibility with proper performance. This also enables the user to specify three versioning states such as versioned, suspended, and unversioned ; to do what-if experimentation; and to apply version management to class and instance object. Time performance of the system implementing the model is good when we consider the point that model is implemented independently of specific DB systems. In future, we plan to extend our model for maintaining more complete object consistency and to improve the performance.

1. はじめに

マルチメディア情報管理をはじめとして，CAD情報管理，文書管理等，様々な分野でOODB（オブジェクト指向データベース）が使用されている．OODBの特長を生かし，より複雑な応用分野に利用するためにはバージョン管理の概念が必要である．

しかし，バージョン機能の導入目的はアプリケーションごとに様々である．対象分野を設計支援に限定した場合でも，同一仕様の設計における異なる実現手段の比較評価，動作対象に対する最新版反映の遅延，版情報と更新履歴情報との統合管理等，各アプリケーションはそれぞれ異なった目的に重点を置き，この結果バージョン機能に対して異なった適用対象に対する異なった仕様を要求する．

そこで我々はバージョン機能の仕様そのものを比較的簡単に変更可能なバージョン管理モデルの構築を目指す．モデルは機能の後天的提供を実現し，提供機能自体をユーザが選択・拡張可能であるよう構成される．この際，バージョン機能の有無による影響が通常操作に及ばないよう考慮する．

この目標の実現のために，我々が必要とするバージョン管理モデルは既存のものとは異なる仕様になる．

以下に，既存モデルのいくつかを示しておく．

(1) 履歴データタイプの導入による版管理モデル[6]
履歴タイプと名付けられたテーブルを用意し，ここで版管理情報と履歴情報の双方を一括管理することにより，管理面，検索面での連携を可能とする．拡張RDB上におけるタプルのみを対象とするのに加えて，情報保存の目的が再使用に備えたものではなく，履歴管理であるとい

† 株式会社富士通研究所 マルチメディアシステム研究所メディア統合研究部
Media Integration Laboratory FUJITSU LABS. LTD.

う点で我々の目的と異なっている．

(2) Odin[5]
旧版を最新版と同じ性能で参照可能であること，および，版管理機能適用時にも版モデルのない場合と同程度の性能であること，を目標とする．加えて版管理機能導入による既存機能への空間的・時間的悪影響の排除，実行性能向上，に対する注意が払われている．ただしここでの適用対象はOODB上におけるインスタンスのみであり，我々はクラスに対する適用とインスタンスとの整合性に対する考慮を行う．

(3) Version server data model[4]
それまでに提案された各種のバージョン管理機能を統合・改良したものであり，将来のレビューのために設計決定過程を記録するための履歴機構を用意することを目的とする．これに対して我々の提示するモデルでは，旧バージョンの頻繁な参照や再利用を前提とした旧状態の保存を目標とするため，このシステム以上にバージョン検索効率，クラスとインスタンスとの整合性確保に対する配慮を行う．

以下本論文ではこれまで我々が検討し，V modelと命名したモデルの概要と実現を示す．

本文の構成は以下のとおりである．第2章では，設計に先立ち考慮したユーザからの要求，および，モデルの設計方針を示す．第3章ではモデル構成について説明する．ここで提示した特長を持つバージョン管理機能を既存DB上に実装した．第4章は上記モデルの構成上特徴となる機能とその実現法の説明であり，第5章，第6章はそれぞれ基本機能の使用法，機能拡張を行う場合の一例である．さらに，試作した機能群の性能に関する評価を第7章にまとめておく．最後に第8章において，今後の研究方向について述べる．

2. 設計思想と方針

いくつかのアプリケーションを対象としてバージョン管理機能に対する要求を調査し，これを基に設計方針を決定した．

2.1 ユーザ側からの要求

まず設計情報管理を必要とするユーザを対象とする要求調査の結果を以下に示す．

2.1.1 適用対象

設計情報の保存・管理システムとして注目されているものにOODBがあげられる．そこでこのOODB上でのバージョン管理機能の提供を想定する．この際，適用対象を一種類に限定しない．すなわちインスタンスへの機能適用に加えてクラスへの適用を必要とする．

この要求を満たすために，OODB上におけるクラス，インスタンスそれぞれのオブジェクトを独立した適用対象とする．たとえば，同一クラスに属するインスタンス群の一部のみを適用対象とすることを可能とし，またインスタンスとクラスの双方に対してそれぞれバージョン機能を適用することを可能とする．

2.1.2 性能要求

バージョン機能の導入により不可避となる必要記憶容量の増加の抑制，および，バージョン機能を必要としない場合の実行性能の劣化防止，が要求された．

前者の要求を満たすために旧情報の保存形態に注意を払う．理想的には，必要とする記憶容量が少なく，情報操作に要する時間が短いものが望ましいが実現は困難である．そこで，時間性能，空間性能それぞれを優先するものをともに用意し，選択をユーザに任せることを考える．また後者の要求を満たすために，バージョン機能が適用対象外のオブジェクトや，システム機能に代表される他の既存機能には影響しないようにモデル構成を決定する．

2.1.3 機能要求

改版機能や旧状態の参照機能の重要性は共通の認識であるが，その他の機能に対する要求はアプリケーションごとに様々である．これらアプリケーションより提示される様々な要求のすべてを満たすことは現実問題として困難である．

そこで，ユーザによる拡張，希望する仕様の構築を可能とする要素機能群を提供する方向で検討する．さらに性能に対する要求を考え合わせると，提供される機能群は，互いに独立し，使用されない機能に起因する性能劣化が生じないように不要な機能を排除した新構成を構築可能である必要がある．

なお今回はバージョン機能を必要とする二大目的のうち，履歴管理ではなく旧情報の再利用に重点を置く．特にバージョン機能を使用する主目的として，設計システムにおける試行錯誤の支援を想定する．このため旧バージョンの再使用を前提とした保存，what-if型の試行（任意のオブジェクトの任意のバージョンの組合せによる実行，すなわち，複数のオブジェクトそれぞれに対して使用するバージョンを指定可能とする機能），等の機能が要求される．

さらに機能実現の容易さを実現するために，オブジェクトに対するバージョン機能の適用が生成時以後任意の時刻に可能であるよう，その適用法を考える．

2.2 設計方針

以上第2.1節に示した要求は一言でいえば，ユーザ

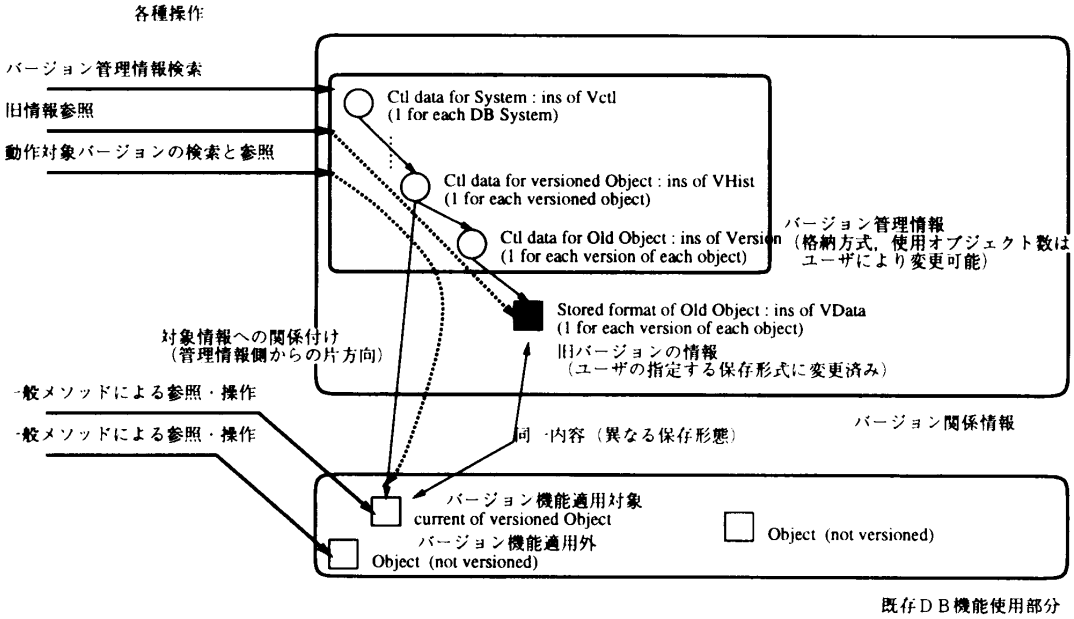

図1 管理情報と適用対象オブジェクトの関係
Fig. 1 Basic object composition of V model.

による仕様決定と性能決定，ということである．これを満たすために，V model は以下を設計方針とする．

(1) 機能の後天性

バージョン機能自体の変更が容易であるよう，システムに対する機能提供はユーザが後天的に行う．この際，バージョン機能適用後のオブジェクトを対象とする既存機能操作が可能であり，既存機能はバージョン機能により使用不可能となるような深刻な影響を受けることはないものとする．

なお対象となるシステムは一定条件を満たす不特定多数の OODB システムとする．

(2) 機能の独立性・モジュール化

ユーザによるカスタマイズ，機能拡張を簡易に実現可能とする．このため，バージョン機能自体もモジュール化し，機能仕様変更時の変更範囲を限定する．すなわち，ユーザはバージョン機能の適用の際，他の機能の修正を必要としない．

(3) 同一機能の複数手法による提供

表面的に同一な機能の実現手段として性能の異なる複数の手法が存在する場合，それぞれを提供する．選択は実行時にユーザにより行われる．

3. モデル概要

以上で示した設計方針に対応するため，V model は以下の構成をとる．

3.1 適用対象

試行錯誤の支援を目的として，関係するオブジェクトの任意のバージョンを組合せて使用することが要求されるため，バージョン機能はクラスとインスタンスそれぞれを独立に適用対象とする．このため，オブジェクトの，特にクラスとインスタンスの，バージョンの違いにより DB 内の整合性が破壊される可能性があるが，これを避けるために，それぞれのオブジェクトのバージョン管理情報として他のバージョン適用されたオブジェクトとの整合条件を与える．

なお，コンフィグレーション☆を対象とするバージョン管理は行わないが，後述するスナップショット機能を提供することにより，これによる不都合の幾分かを回避可能である．

3.2 管理情報と適用対象オブジェクト

管理情報と適用対象オブジェクトの関係を**図1** に示す．

機能実現のために使用される各種情報は以下のように分類される．

(1) 管理情報

バージョン管理の実現のためには各種の制御情報の保存が必要である．この制御情報をその影響範囲や意味付けにより複数のオブジェクトに分割保存し，これら一連のオブジェクト群を管理情報保存オブジェクトと呼ぶ．

基本となる管理情報保存オブジェクトは，システム

☆ 複数のオブジェクトを構成したもの．一般に DB 中でオブジェクトとして認識されていないため，オブジェクトを対象とするバージョン管理機能の対象外となる．

全体のバージョン情報をまとめるもの（Vctl），それぞれの適用対象オブジェクトのために用意されるもの（VHist），それぞれのバージョンごとに用意されるもの（Version）の3種である．これらのオブジェクトは必要に応じて用意されバージョン機能適用対象外のオブジェクトのための領域が用意されることはない．

それぞれのオブジェクト間には各種の関係付けがなされている．ここで保存される関係としては，導出関係，作成時順関係（いずれもVersionに保存）等の検索効率を考慮したものや，呼び出し関係にある他のオブジェクトとの関係付け（VHistに保存）等が存在する．

なお，変更履歴情報等をこの管理情報として保存することにより，履歴情報管理機能を充実されることが可能である．

ここに保存された情報の参照操作はバージョン機能として提供される．

(2) バージョンのオブジェクト情報

バージョン機能は，本来更新により消滅するはずのオブジェクト情報を各種の属性を付加したうえで保存し，保存された情報を後に再利用する機能である．この保存のための領域を旧情報保存オブジェクト（VData）として用意する．

なお，バージョン機能の提供により問題となる必要記憶容量の増大は一般にこの領域によるものであるため，この領域の増加の抑制を考える．具体的には，旧情報をほとんどそのままの形で保存するための方式以外に，差分抽出により保存対象情報の一部を保存することでこれに代える方式，を用意する．ただし後者は，保存時における差分抽出時間，および参照/再利用時の復元時間を要するため，時間的な性能劣化の原因となる．したがって使用する方式の指定は，時間性能と空間性能のどちらを優先させるべきかを判断可能である者（ユーザ）に一任する．

ここに保存される情報はバージョン機能が適用されたオブジェクトの旧状態であり，その参照はバージョン機能より行われる．

(3) 適用対象オブジェクト（動作対象）

バージョン機能の適用対象はオブジェクト，すなわちクラスとインスタンスである．

既存機能による適用対象オブジェクトの操作がそのまま可能であるよう，特定の1つのバージョンはバージョン機能適用後も既存の形式で存在させる．このバージョンを動作対象バージョンと呼ぶ．

すなわち動作対象バージョンは既存DBの通常のオブジェクト形式のまま存在するバージョンであり，バージョン機能が適用された以後の既存機能の操作対象となる．これを対象とする各種操作はクラス・インスタンス間変更伝播を含めバージョン機能の適用されない通常オブジェクトと同様に実行される．また複数のバージョン中より1つを特別視することにより整合性保持の際の基準としても使用される．この動作対象バージョンはユーザにより直接変更される他，改版により最新版に自動的に変更される．

3.3 変更対象と実行対象

試行錯誤の支援を目的としているため，最新版以外を対象とする変更と実行を可能とする．

この変更により改版によるオブジェクトの導出関係は同一バージョンからの複数の導出（alternative）を認めるもの，すなわち木構造となる☆．

実行に関しては，副作用（自他オブジェクトの変更）をともなう可能性を持つため，実行対象を一時に1バージョンに限定し，同一オブジェクトであれば変更対象と操作機能のバージョンを一致させ整合性を保つよう考慮する．たとえば自己変化を目的としたメソッドが同一オブジェクトの他のバージョンを操作し自身に変化を及ぼさない，等ということがないよう制限する．この操作対象としては動作対象バージョンを選択する．

動作対象バージョンはメソッド実行可能なただ1つのバージョンであり，変更可能なバージョンである．また既存機能がバージョン機能を介さず参照・操作可能なただ1つのオブジェクトであるため，その時点で参照頻度の非常に高いオブジェクトを動作対象として選択することにより，参照性能の向上が可能となる

3.4 機能一覧

モデル上で実現される基本機能をここで一覧する．

(1) 適用対象オブジェクトの指定と解除

Ret=START(); .....(A)

Ret=SetObjectState(Object Obj, State); .....(B)

Obj: 適用対象オブジェクト．クラス/インスタンス

State: 適用開始（START）/適用終了（END）/他

Ret: 成功/失敗 □

システム上でバージョン機能の使用開始を指示する機能（A），およびDB上に存在するオブジェクトを対象としてバージョン機能の適用を開始/終了する機能（B）である．

バージョン機能の後天的提供と適用/非適用オブジェクトの混在を実現するために提供され，ユーザはバー

☆ この導出関係は管理情報として保持される

ジョン機能の適用を希望するオブジェクトをこの機能を用いて陽に指定する．

(2) 管理情報検索

VHist Hist=
GetHistorical(Object Obj); .....(C)
Version VObj=
GetVersionObject(VHist Hist, VerNo); .....(D)
Hist: 対象オブジェクトの管理情報保存インスタンス
VerNo: 参照を希望するバージョンのID
VObj: 指定バージョンに固有の管理情報を保存するインスタンス □

適用対象オブジェクトに対応する管理情報保存オブジェクトを抽出する機能（C），および特定バージョンに対応する管理情報保存オブジェクトを抽出する機能（D）である．

V model では適用対象となるオブジェクトそれ自体の形式は表面上いっさい変化しない．したがって管理情報との対応関係も管理情報側にのみ存在し，適用対象側ではその情報を持たない．このため各種バージョン機能は対象オブジェクトに対応する管理情報を介して実行される．この管理情報の検索操作を提供する．

この検索操作は以後の各種操作の前に必ず実行されるため，何らかの共通設定を必要とする場合にはこの仕様を変更すると良い．たとえば管理情報と対象オブジェクトとの整合性の確認/復元操作はここに組み入れられている．

(3) オブジェクトの改版

Version VObj=ReplaceLastVerion(VHist Hist, ModifyInfoList, VDataType, ChangeNotif);
ModifyInfoList: オブジェクト更新情報．更新メソッドと概メソッドの引き数の一覧．
VDataType: 旧情報の保存方式．全情報の保存（ALL）/親バージョンとの差分のみを保存（DIFF）
ChangeNotif: この操作の変更伝播種別．
改版（VUP）/通常更新（OW）/伝播なし（OFF）
VObj: 最新バージョンに固有の管理情報 □

対象オブジェクトの改版を行う機能である．

提供されている任意の更新メソッドに対応させるため，その更新メソッドと引き数をそのまま ModifyInfoList に指定する．

また最も性能的に問題となる旧情報の保存操作において優先すべき性能をユーザ制御とするため，保存方式の指定をここで行う．

なお，VObj はここで更新された最新のバージョンの属性・管理情報を保持する管理情報保存オブジェクトの一種であり，たとえば VObj.VerNo はここで更新されたバージョンの ID を保持する．

(4) 動作対象バージョンの変更

Version VObj=ReplaceActiveVersion
(VHist Hist, VerNo, ChangeNotif);
VerNo: 新たに動作対象となるバージョンの ID
VObj: 新たに登録されたバージョンに固有の管理情報を保存するインスタンス □

指定された番号のバージョンを動作対象，すなわち既存 DB 機能が対象オブジェクトとして参照/操作可能な形式に変更する機能である．この機能はバージョン機能外部からは，該当オブジェクトが変化した（更新された）ように観測される．したがってクラスの更新にともなうインスタンスの更新等の各種変更伝播を必要とする．ただし使用される伝播範囲や条件等あらかじめ管理情報としてユーザが登録可能である．この変更伝播操作終了後のクラス・インスタンスを含めた他のオブジェクトとの関係はあらかじめ指定された，ユーザが整合性を保っていると認識する状態となる．

(5) 旧バージョン参照

Object Obj=
GetOldVersion(VHist Hist, VerNo);
VerNo: 参照を希望するバージョンの ID
Obj: 該当バージョンのオブジェクト構成 □

旧バージョンの情報を参照する機能である．

一般的には以下の記述により，該当 ID の Slot1 の値が取得可能である．

GetOldVersion(GetHistorical(Obj), No).Slot1 □

(6) オブジェクトの削除

Ret=DeleteVersion(VHist Hist, VerNo); .....(E)
Ret=Obj.Delete(); .....(F)
VerNo: 削除を希望するバージョンの ID
Obj: 削除を希望するオブジェクト
Ret: 成功/失敗 □

指定バージョンの削除（E），および指定オブジェクトの削除（F）を行う機能である．すなわちバージョン単位の削除を可能とする．

前者はバージョン管理情報と旧情報をその保存領域ごと削除する．この際導出関係の修正がなされ，バージョンが子（該当バージョンに変更を与えることにより生成されたバージョン）を持っている場合には，この子バージョンは削除対象の親の直接の子として再登録される．

後者はその全バージョンを対象として削除操作を

行ったうえで，動作対象，すなわち，DBにより該当オブジェクトとして認識されるものの削除を行う．

(7) 管理情報参照/設定

管理情報はオブジェクト提供されているため，その参照/設定はDBにより提供されているオブジェクト情報参照機能をそのまま利用可能である．

4. 実現方式

以上で示したモデルの実装例を以下に示す．

4.1 適用対象DB

まず，提示したバージョン管理機能の後天的実現を可能とするDBの条件について説明する．

(1) 多重継承

複数のクラスを継承して新たなクラス定義を実現する機能である．

たとえば，クラスライブラリとして提供される複数のバージョン機能モジュールを取捨選択して，機能拡張や機能限定を行う際に使用される．

(2) デモン

指定メソッドの実行にともなってその前後に実行される操作，あるいは対象データへのアクセスがなされた場合等，特定の状況が発生した場合にその内容に応じて実行される操作である．

オブジェクト削除時に，該当オブジェクトのバージョン管理情報の削除をあわせて行う，通常のオブジェクト更新実行時にこれを改版と認識して更新に先立ち情報保存を行う，等の目的で使用される．

(3) オブジェクト/メソッドの動的呼び出し

操作対象となるオブジェクトのタイプや操作内容をプログラム実行時に決定する機構である．

特に任意のオブジェクトを対象としてバージョン機能を適用する際に使用される．

以上のように，モデルの実現のために適用対象DBに要求される機構はオブジェクト指向の特長と一致しており，適用対象システムとしてのOODBの選択は，設定目標，すなわち，拡張性や独立性の実現のために正しいものであったと考える．

なお実装にあたり，適用対象システムとして当社で研究・拡張中のOODB・JASMINE[2),3)]☆を採用した．

4.2 DB上における機能実現

既存DB上において，V modelを基とするバージョン機能は，提供機能と管理情報保存領域を定義したクラスを登録することにより実現される．

先に提示した基本機能はモデル実現のために用意されているが，上記クラスを再定義することにより，異なる仕様を実現するものに変更可能である．

4.3 管理情報とオブジェクトの構成

必要な管理情報を以下の条件に従って用意する．

- インスタンスによる保存領域の用意

  後天的適用を目指しているため，当然のことながら，適用対象は管理情報の保存領域あらかじめ持たない．したがって機能の適用開始時にこの領域を生成し適用対象と対応付ける必要がある．V modelは既存機能群に対する影響を排除するため，オブジェクトそのものを変更することなくこの対応付けを実現する必要がある．そこで，管理情報保存領域を適用対象オブジェクトとは別に独立した☆☆ インスタンスの形で用意する．

- 適用対象と管理情報の対応

  適用対象オブジェクトに対する変更を回避するため，管理情報との対応関係は管理情報側が一方的に保持する．このためオブジェクト側からの管理情報の検索機能が必要となるこの検索効率を高めるため，インデクスを用意し，あわせてこれを適用対象オブジェクトの一覧として使用する．

- 管理情報間の関係付け

  ユーザによる試行錯誤，すなわち，バージョン参照操作や動作対象バージョンの変更操作が頻繁であることを考え，バージョン検索の所要時間を短縮するために，関係する管理情報保存オブジェクト間をリンクする．導出関係，生成時順等を関係情報として保持し，特に重要なものは双方向に関係付ける．これを用いて高速なオブジェクト検索を可能とする．

- 管理情報保存領域の分割

  バージョン機能の利用者による機能拡張/変更が容易であるよう，保存領域は保存対象情報の役割や影響範囲に応じて分割して用意する．

  ユーザは管理情報定義クラスやその操作定義を保持するクラスを必要数継承して新たな管理情報保存クラスを再定義し，これを使用することが可能となる．

以上を考慮して保存領域定義として以下を用意する．

(1) 管理情報保存オブジェクト

(a) システム情報保存領域（ins of Vctl）

DBシステム全体に影響する管理機能を保存する領域であり，システム上にただ1つ存在する．バー

☆ このシステムの商業版としてはOODB・ODB Ⅱが存在する．

☆☆ 継承機構などを使用しないことを意味する．継承機構による管理情報と適用対象オブジェクトの対応関係の実現は，適用対象のクラス定義の変更を要するため望ましいことではない．

ジョン機能が適用されているオブジェクトの一覧等，システム全体に関係する情報が保持される．

(b) バージョン情報保存領域（VHist）

バージョン機能の適用対象オブジェクトごとに用意され，各適用対象オブジェクトに固有かつ全バージョンに共通する情報を保持する．各適用対象の最新バージョン，動作対象バージョン，オブジェクト更新の際にデフォルトで使用される旧情報保存方式，動作対象バージョン変更時の他のオブジェクトへの変更伝播の有無，等に関する情報がここでの保存対象となる．

(c) 特定バージョン情報保存領域（Version）

各バージョン機能の適用対象の各バージョンのみに関係する情報を保持し，それぞれのバージョンごとに用意される．各バージョンのID，該当バージョンの導出関係，生成時順，時刻情報，ユーザ情報，等がここで保存される．

また履歴情報の管理をバージョン機能が行う場合には，変更要因，更新状況，等の情報保存領域定義を作成し，ここに継承させる．

(2) 旧情報保存オブジェクト

各適用対象のそれぞれのバージョンの内容を保持するためのものであり，各適用対象オブジェクトの各バージョン単位で用意される．

まず，基本定義（VData）を用意し，このクラスの継承により各種の保存方式，たとえば，保存時の時間性能と空間性能それぞれを優先させたもの，で使用される保存領域定義を行う．

(a) 基本クラス（VData）

すべての情報保存機能の基本となる機能と保存領域定義がここで行われている．たとえば使用された保存方式を示す情報がこの保存領域の保存対象となる．したがってユーザはこれを継承した新クラスを用意することにより新たな情報保存機能と保存領域を定義することが可能である．

以下に示すクラスもこのクラスを継承することにより実現されている．

(b) 全情報保存形式（VDataAll）

スロット，メソッド名の一覧とその定義/値等，オブジェクト構成要素を保存するための領域定義と保存操作メソッド定義である．またこのクラスの形式により保存された情報を復元する際の復元方式，等もここで記述されている．

(c) 差分情報保存形式（VDataDiff）

比較対象となったバージョンと，そのバージョンと比較した場合の該当バージョンの差異，等を保存する領域とその操作（保存/復元）機能がここで定義される．

4.4 提供機能の実現

ここではモデルを構成する各機能の実現について提示する．

各機能の実現に先立ち，まず以下の方針を確認する．

- バージョン機能の適用後も，DB既存機能はそのまま使用可能とする．したがって既存機能に対する仕様変更は可能なかぎり回避し，やむをえず変更を要する場合にも該当機能のデモンを用意することにより実現し，該当機能本体に対する変更は行わない．
- ユーザによる仕様変更の際の変更範囲の限定を実現するために，提供機能を機能種別ごとにモジュール化し，さらに提供メソッド自体をモジュールを構成することにより実現する．

  さらにユーザが必要とするクラスのみを抽出して使用可能であるよう構成する．これによりバージョン機能登録時に必要部分のみを実装し，不要部分が存在することよる性能劣化の回避を可能とする．
- バージョン機能の導入に起因する主に時間面での性能劣化を既存DB機能には影響させない．

  たとえば，バージョン機能の適用対象に対する通常更新操作（動作対象のバージョン機能を用いない変更）は該当オブジェクトの変更に管理情報の変更がともなわず，管理情報と実オブジェクトとの整合性を破壊することになるが，これにより生じる混乱を消滅させ，整合性の復旧を行う機構はバージョン機能側で用意する．これは更新操作に先立ち該当オブジェクトがバージョン対応のものであるか否かを調査し管理情報の変更を行う時間を更新操作に要求することを回避したためである．この方針は，システム全体としては総合性能の劣化を招くが，ここでは機能のモジュール化を重視し，上記方式を採用する．

以上の方針に従って機能の実現を試みる．先にモデル概要で提示した各機能は機能提供クラスVerFuncのメソッドとして実現された．

(1) 機能適用対象の指定と解除

システムにおける機能の使用開始はVctlの，またオブジェクトを対象とする機能の適用開始はVHist, Versionの，それぞれのインスタンス生成とこれに対する情報設定操作により実現される．

適用終了機能は対象オブジェクトのすべてのバージョンとその管理情報を順次削除することにより実

現される．ユーザにより陽に実行される場合の他，該当オブジェクトの削除にともないそのデモンにより実行される場合が存在する．

(2) オブジェクトの改版

この機能は，指定された旧情報保存方式を確認し保存すべき情報の保存形態への変更と保存を行う機能，管理情報の保存を行うための領域を用意しここに新バージョンの属性情報を保存する機能，および，通常のオブジェクト更新操作と各種の情報設定操作より構成される．外部的にはオブジェクトの更新操作そのものである．ただしオブジェクト更新中で行われる変更伝播に関しては，デモンを用意しバージョン機能を呼び出すことで実現される．これは伝播操作として更新と旧バージョンの使用の2通りが考えられるためである．どちらを使用するかは管理情報を参照して決定する．なお，保存形態への変更と保存操作に関しては第(5)項参照のこと．

(3) 動作対象バージョンの変更

動作対象バージョンの変更操作は各種属性情報の設定と指定バージョンの復元（既存オブジェクト形態への変更）情報を使用することを除けば，オブジェクトの更新操作である．ただし関連オブジェクトへの変更伝播はバージョン機能外部の既存機能中での実行に先立ち，バージョン機能としてユーザの指定に合わせて実行される（第(4)項参照）．

(4) 動作対象バージョンの変更伝播

関係するオブジェクト群がバージョン機能の適用下にある場合，基本的に任意のバージョンの組合せが可能である．

したがって，逆にシステム内の情報整合性を保つために，たとえば同一クラスのインスタンスに関してはバージョン数を合わせる等，特定のものとの組合せのみを希望する場合には，その制約条件あらかじめ保存し，これを用いて制約どおりの組合せを実現する機能が必要となる．この機能を以下で実現する．

(a) VHist上での制約条件記述領域の用意

たとえばクラスの管理情報側に伝播範囲（たとえばクラスのインスタンス）と伝播条件（クラス定義のスロット名とタイプに一致するまでインスタンスのバージョンを順次下げる）等をメソッド形式で記述する．この情報の変更は通常の管理情報設定機能を用いて行う．

(b) 制約条件実行

制約条件はオブジェクトの動作対象バージョン変更操作中で実行される．

(5) 旧情報の保存

いくつかの性能の異なる方式を用意することにより記憶容量や保存・参照操作の時間性能のユーザによる制御を可能とする．旧情報保存領域VDataはここで使用される保存方式それぞれのために複数種類が用意される．

初期状態では，全情報を保持する形態と，情報を差分形式☆に変換して保存するもの，の2通りを用意する．各改版操作時にどちらを採用するかはユーザが実行関数の引き数の形で指定する．保存領域定義はVDataを継承する．保存領域定義と上記指定内容の対応関係はクラスVData中に表形式で存在し，表の更新機能はユーザに公開される．

領域生成と，領域への保存操作は先に指定されたフラグに基づいて相当するクラス（継承クラスとしてVDataが存在）のものを使用する．

また，新たな保存領域と保存方式（メソッド）を定義し，これを先の対応表に追加登録することにより，ユーザ定義の保存形態を追加することも可能である．

(6) 管理情報検索

バージョン機能適用開始操作や，管理情報自体の削除を実行するオブジェクト削除操作を除き，すべての機能はあらかじめこの管理情報検索操作を実行し，ここで得られた管理情報を指定することによりこれが管理しているオブジェクトを対象とする操作を実行する．

したがって管理情報と対象オブジェクト間の整合性の確認等，バージョン機能が必須とする前処理はここに組み入れられることが望ましい．

たとえば適用対象オブジェクトの通常更新操作，すなわち管理情報の変更をともなわない動作対象バージョンの変更に起因する管理情報と対象オブジェクト間の整合性の破壊が，ここで管理情報側に存在する動作対象バージョン変更時刻と，DB側に存在するオブジェクト更新時刻を比較することにより検索される．両者が異なっている場合，すなわち，整合性が破壊されている場合にはその旨のメッセージを発生し，VHist中に存在する動作対象バージョンに関する情報領域に動作対象がバージョン機能の管理

☆ 2つのオブジェクトが存在する場合に，一方を他方との差分をとることにより表現するものである．
通常方式が保存・参照時の情報変換に要する時間を必要としない時間効率の良いものであるのに対し，情報保存時における差分情報の抽出，情報参照時における差分情報から全体への復元，等の操作を必要とするため，保存時，参照時の双方で時間的性能は低下する．ただし保存されている情報は圧縮されているため必要記憶容量の削減が可能である．

外である旨の記述を行う．以上の操作を行うことにより，たとえば動作対象バージョン検索の際，更新操作直前に存在したオブジェクトのID等の間違った結果結果を返す，等の問題を回避可能となる．

5. 使　　用

以上に示したバージョン機能の一般的使用形態は以下のようになる．

(1) 機能提供クラスの登録

機能提供クラスVerFunc，管理情報保存定義VCtl，VHist，Version，VData等を必要に応じてカスタマイズした後に登録する．なお，適用対象となるオブジェクト（以下でTestObj）はこの時点ですでに存在していてもよい．

(2) 機能提供開始

```
VerFunc.START();                                    □
```

システム上でのバージョン機能の使用を宣言する☆．

(3) オブジェクトへの適用開始

```
/* TestObj.Slot1 == 0 */
VerFunc.SetObjectState(TestObj, START);             □
```

(4) 管理情報IDの検索と情報参照

```
Hist=VerFunc.GetHistorical(TestObj);
/* VerFunc.GetOldVersion(Hist, 1).Slot1 == 0 */
/* TestObj.Slot1 == 0 */                           □
```

各種の情報参照が可能である．ここで取得可能な情報としては最新バージョン（Hist.LastVersion），動作対象バージョン（Hist.ActiveVersion），コメント（Hist.Comment）等がある．

(5) 改版

```
VerFunc.ReplaceLastVerion(Hist, [SetSlot, TestObj,
Slot1, 10], ALL, OFF);
/* VerFunc.GetOldVersion(Hist, 1).Slot1 == 0 */
/* VerFunc.GetOldVersion(Hist, 2).Slot1 == 10 */
/* TestObj.Slot1 == 10 */                          □
```

オブジェクトTestObjのSlot1の値を10に変更する．これを改版と認識させる．

(6) 既存機能の使用

```
TestObj.Slot1=100;
/* Hist.ActiveVersion == 2 */
/* VerFunc.GetOldVersion(Hist, 2).Slot1 == 10 */
□
```

既存機能の適用後，上記のように管理情報側（TestObjのSlot1の値は10）と対象オブジェクト（値は100）情報の間に非整合が生じる可能性がある．しかしこの非整合は整合操作を行う管理情報検索機能の実行後には解消される．以下の式中のNILは現動作対象がバージョン機能の管理対象ではないことを示す．

```
Hist=VerFunc.GetHistorical(TestObj);
/* Hist.ActiveVersion == NILL */                   □
```

(7) 動作対象バージョンの変更

```
VerFunc.ReplaceActiveVersion(Hist, 1, OFF);
/* VerFunc.GetOldVersion(Hist, 1).Slot1 == 0 */
/* VerFunc.GetOldVersion(Hist, 2).Slot1 == 10 */
/* TestObj.Slot1 == 0 */                           □
```

動作対象をバージョンIDが1であるものに戻す．TestObj.Slot1の値は0として認識される．例題は単純なインスタンスの変更であるので，変更伝播は実行されないよう指定（OFF）されている．

(8) バージョン機能の適用終了

```
VerFunc.SetObjectState(TestObj, END);
/* VerFunc.GetOldVersion(Hist, 1).Slot1: error */
/* VerFunc.GetOldVersion(Hist, 2).Slot1: error */
/* TestObj.Slot1 == 0 */                           □
```

ここで適用対象オブジェクト自体は削除されないことに注意．なお存続するのは最新版ではなく動作対象バージョンである．

6. 拡　　張

以上で基本機能に関して説明したが，ここでは特長とする拡張の容易さのためにどのような考慮がなされているかを実際の拡張操作を例にあげて説明する．

6.1 管理情報の追加

たとえば履歴管理などを目的としてバージョン管理の保存対象となる情報を追加することを考える．

この場合にはVHist，Versionに新たな項目を追加することによりユーザ定義の情報をバージョン管理情報として保存することが可能であり，これらのクラスに対するメソッド追加によりそれらの情報を使用した操作を提供することが可能となる．

6.2 マルチビュー対応

システムがマルチユーザ対応の場合を考える．このとき，複数ユーザが異なるバージョンのオブジェクトの使用を希望する場合，同一のバージョンの使用を希望する場合，それぞれが考えられる．そこでオブジェクト空間を分割し，それぞれの空間に検索優先順位を

☆ なおこの後使用される表記法の意味は以下のとおりである．
A.M()：オブジェクト（クラス・インスタンス）Aのメソッド（関数）M
A.S：オブジェクトAのスロット（データ項目）S

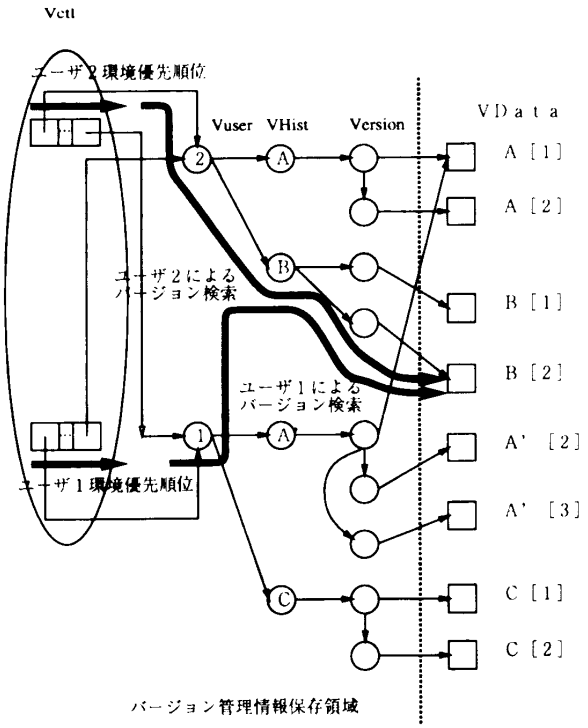

図2 オブジェクト検索優先順位
Fig. 2 User environment priority.

割り当て，バージョン検索機能を空間全体を一時に検索するものからこの順位に従って順次検索するものに変更する．**図2** にこの機能の概念を示す．ユーザ1，ユーザ2がそれぞれ自分のローカルエリアを最優先，相手先をその次に指定している場合，オブジェクトAのように双方に存在するものはそれぞれのものを該当オブジェクトとして使用し，Bのようにユーザ1が保持していないものに関しては相手先を検索し相手が認識するオブジェクトとその管理情報を使用することになる．

この機能は以下により実現される．

(1) オブジェクト空間の分割

Vctl内に存在するバージョン機能適用対象一覧を分割し，それぞれにIDを対応付ける．バージョン機能一覧は各オブジェクト空間単位にこのIDで管理される．

(2) 順位保存/設定機能の追加

上記の各空間単位に用意される適用対象一覧のIDとその空間の検索順位を対応付けて保存するための領域定義と機能定義を行うクラスを用意する．

この領域はユーザ個数，あるいはビューの個数生成される．

(3) オブジェクト検索機能の変更

バージョン機能は専用の管理情報検索機能を用いて対象を特定し，これに対する操作を行う．この検索操作に代わり，先に保存されたID順に空間検索を行う機能を用意する．

それ以外のメソッドは操作/参照対象オブジェクトの決定時にこの機能を利用することになり，変更は必要とされない．

6.3 旧情報保存方式の追加

基本機能として提供されている2つの方式に加えてユーザ定義の保存方式を使用することを考える．

旧情報保存方式の選択はオブジェクトの改版実行時にユーザにより陽に指定され，保存時に保存情報とともに使用された保存方式が記録される．この保存情報の参照は，動作対象変更/旧情報参照時に記録された保存方式に対応するオブジェクト復元機能を呼び出すことにより実現される．保存方式はオブジェクトの改版時でフラグの形で指定され，このフラグ値と保存操作の関係はテーブルにより管理されていたる．したがって方式の追加は，このテーブルに新たな対応関係の記述と，その際使用される情報保存領域/機能定義の登録を行うことにより可能である．

したがって新たな保存方式は以下により導入される．

(1) 保存方式（オブジェクトの保存形態への変換方式）と復元方式（保存された情報から通常オブジェクト形態への変換方式）定義．

希望する保存形態への変換と保存形式からオブジェクト情報を取得するメソッド，および変換後の情報を保存する領域を，VDataを継承する新たなクラスを用意しここで定義する．

(2) フラグの決定と定義登録．

保存領域定義とフラグの対応関係を保存するVData中に存在する対応表に追加登録を行う．

VData.StoreFormatList.add([flag := 3, VDataNew]);

StoreFormatList: 対応表のList表現

VDataNew: 新規保存方式．これにより改版時にflagとして“3”が指定された場合，変換メソッドとしてVDataNew.Storeが，復元時にVDataNew.Restoreが使用される． □

なお，VDataNewはVDataを継承し基本機能を保持していることが必要である．

6.4 スナップショットの保存/復元機能

V modelはクラスとインスタンスの双方をそれぞれ独立に適用対象とする点を特長としているため，逆にある時点の状態の復元，すなわち特定時刻に存在したオブジェクト群のすべてをその時点で動作対象となっていたバージョンに変更すること，を実行するためにはユーザが陽に全オブジェクトの動作対象を変更する必要がある．

これを補助するために，ある時刻に関係する一連のオブジェクト群をその動作対象バージョンとともにまとめて記憶する機能，および，記憶された情報を用いて動作対象バージョンの変更をまとめて行う機能を提供する．記憶された情報をスナップショット情報と呼び，上記の2機能をスナップショット機能と呼ぶ．

この機能は以下の機能と領域の定義を登録することにより実現される．

(1) スナップショット保存/復元機能の定義

VersionSnap VSnap=GetSnapshot(Schema); ....(G)
Ret=SetSnapshot(SnapshotID); .....(H)
Schema: 保存対象範囲
VSnap: スナップショット情報保存インスタンス
SnapshotID: 復元を希望するスナップショットのID
Ret: 成功/失敗 □

関係する一連のオブジェクトのその時点で使用されていたバージョンを記録する機能（G），および一連のオブジェクト群のバージョンをまとめて過去のある時点で使用されていたものに変更する機能（H）である．前者は，保存領域VSnapを生成し，その時点で使用されているオブジェクトをその際の動作対象のIDとともに保存する機能であり，後者は前者で保存された情報を用いて複数のオブジェクトの動作対象をまとめて変更する機能である．

なお，後者で使用されるスナップショットIDは前者で返されたインスタンス中に（VSnap.SnapIDとして）記録されている．

(2) スナップショット情報保存領域定義

時刻情報，および，オブジェクトとそのバージョン情報を記録するための領域である．オブジェクトとそのバージョンIDの一覧の他，各スナップのID，取得時刻等が保存される．

実際にはこのスナップIDとして情報保存インスタンスのIDが使用される．

7. 評価結果

以上で説明した各機能をワークステーション上で実現した場合の評価を行う．

詳細な取得値をあげることはせず，結果概要を以下に示す．部分的に改良の余地はあるが，モデルの根本に基づく問題は発生していないと考える．

- 時間的性能
  各メソッドともその必要操作量を考えると，相対的に問題となるほど長時間を要する機能は見られない．改版機能に要する時間は，ユーザに対する公開メソッドレベルで同一状況における既存更新機能に要する時間の数倍☆1 のオーダである．
  管理情報の保存モジュールの増加により多量のクラス定義を必要とする場合にはこのクラス数の増加に起因する検索時間性能の劣化が生じる可能性があるが，これに関しても常識的には許容範囲にとどまるものと予想される．また，この問題は管理情報保存領域定義を統合し，必要クラス数を減少させることにより解消可能である．
- 空間的性能
  必要記憶容量と保存を要するオブジェクト数は機能提供により急速に増大する．これに関しては，管理情報を最低限にとどめること，および旧情報保存方式を変更する☆2ことで抑制可能である．
- その他
  機能の後天性・拡張性の確保のため提供機能のモジュール化を行い，関係外の機能に対する影響を排除した☆3 ため，バージョン機能のみに，既存機能とバージョン機能との併用により発生する情報不整合の解消を行うための負荷がかかっている☆4．この負荷は，必要に応じて各操作をDB既存機能に組み入れる☆5ことにより解消可能である．この場合にはDB上の既存操作機能の性能劣化を生じることなるが，全体としての性能は向上すると考える．

8. おわりに

以上，既存DBへの後天的適用を可能とし，ユーザ

☆1 特定DB下における試作とテストデータに対する機能適用のため，本文中で具体的数値をあげることはあまり意味がないと思われるが，改版実行は更新実行所要時間に対して0.145 [s] の所要時間の増加であった．これは実用に耐えうる範囲であると考える．

☆2 たとえば先に述べたように旧情報保存方式として差分のみを保存する方式を採用することがこれにあたる．

☆3 厳密にはデモンとしてオブジェクト監視機能がが動作しているため，回避の完全なる達成ではない．

☆4 負荷が生じるのは具体的には以下の点である．
- 管理情報とオブジェクトの非整合性検索
  更新操作が生じたか否か，すなわち整合性が破壊されているか否かに関係なく，管理情報検索時に整合性の確認を行うために生じる．
- オブジェクト削除デモン
  適用対象オブジェクトの削除の可能性に関係なくデモンによる監視を行うために生じる．
- オブジェクト変更時の変更伝播
  動作対象バージョンの変更の際，二重の変更伝播が生じる，すなわちバージョン側による伝播の後，オブジェクトの変更にともなう変更伝播を行うことに起因する．

☆5 既存のオブジェクト更新機能に管理情報とオブジェクトとの整合性の復旧操作を組み入れる，オブジェクト削除機能に管理情報の削除機能を組み入れる，等を意味する．

による拡張を容易にするバージョン管理モデルを構成し，実DB上への実装を行った．この方式は，性能的にも，後天的かつ既存部分と完全に分割されたものであるにもかかわらず実用に耐えうるものとなっている．

さらに，管理情報構成や提供機能群をシステム提供機能と一体化し機能提供層をより下層に変更することにより，より一層の性能向上が可能である．これは，適用対象データ側からの管理情報の直接参照，デモンによる実現部分の既存機能内への組み込み，等が可能となるためである．

今後は，変更伝播仕様の再検討およびアクセス制御機能の導入による情報整合性の改良，特定OODB上でのモデル最適化と性能を考慮した実現を予定している．

参考文献

1) 石川ほか：エンジニアリング業務支援とオブジェクト指向データベース，情報処理，Vol.32, No.5, pp.593–601 (1991).
2) Ishikawa, et al.: The Model Language and Implementation of an Object-Oriented Multimedia Knowledge Base Management System, *ACM Trans, Database Systems*, Vol.18, No.1, pp.1–50 (1993).
3) Ishikawa, et al.: An Object-Oriented Database System Jasmine: Implementation, Application, and Extension, to appear in *IEEE Trans. Knowledge and Data Engineering* (1996)
4) Katz, R.H.: Toward a Unified Framework for Version Modeling in Engineering Databases, *ACM Computing Surveys*, Vol.22, No.4, pp.375–408 (1990).
5) 木村　裕ほか：オブジェクト指向DBMS-Odinの版管理方式，第45回情報処理学会全国大会論文集 (4), 2R-2 (1992).
6) 北川博之ほか：履歴データ型を用いた版管理データモデルの提案，情報処理学会論文誌，Vol.34, No.5, pp.1031–1044 (1993).
7) 鈴木卓治ほか：データベースに対する一括更新の正当性の検査方法，情報処理学会論文誌，Vol.35, No.8, pp.1567–1578 (1994).
8) 宇田川佳久ほか：設計データベースのバージョン管理とその実現，電子通信学会技術報告データベース・システム，63-2 (1988).
9) 吉沢，小野，石川：オブジェクト指向データベースにおけるバージョン管理モデルの設計，第48回情報処理学会全国大会論文集 (4)，2F-1 (1994).

(平成7年4月18日受付)
(平成8年2月 7 日採録)

吉沢　直美（正会員）

1965年生．1988年早稲田大学理工学部電子通信学科卒業．同年（株）富士通研究所入社．以来オブジェクト指向データベース，マルチメディアデータベースの研究開発に従事．

小野美由紀（正会員）

1962年生．1985年専修大学経営学部情報管理学科卒業．同年（株）富士通研究所入社．以来，オブジェクト指向型言語処理系，オブジェクト指向データベース，マルチメディアデータベースなどの研究開発に従事．

石川　博（正会員）

1956年生．1979年東京大学理学部情報科学科卒業．同年（株）富士通研究所入社．以来データベースの研究開発に従事．現在マルチメディアデータベースとデータウエアハウスに興味を持つ．1992年東京大学理学博士号取得．著書にObject-Oriented Database System (Springer-Verlag, 1993)．1994年情報処理学会坂井記念特別賞受賞．情報処理学会誌編集委員会SWG主査，情報処理学会データベースシステム研究会幹事．群馬大学非常勤講師．情報処理学会，電子情報通信学会，IEEE, ACM各会員．