履歴データ型を用いた版管理データモデルの提案

北川博之[†] 田中肇[††]
大保信夫[†] 鈴木功[†]

CADシステム，ソフトウェア開発等を支援するデータベースにおいては，各種設計データの版（バージョン）管理が必要である．版管理においては，版の導出関係管理と各版の生成時刻・削除時刻等の時間情報管理の両者が重要である．本論文では，抽象データ型である“履歴型”を導入することにより入れ子型リレーショナルモデルの枠組みの中で，版の導出関係と時間情報の管理を可能とするデータモデルTVDM (Temporal Version Data Model) を提案する．履歴型は版の導出関係と時間情報を一体として集約化した対象であり，履歴型に付随したオペレータを用いることにより，版の導出関係と時間情報に関する検索条件を明示的に表現可能である．また，版管理上は必須であるが通常のリレーショナルモデルではサポートされない推移的閉包も取り扱うことが可能である．本論文では，TVDMの基本概念について述べた後，その代数操作系TV代数を示す．また，データベース言語TV-QUELを提案し，TV-QUELインタプリタのプロトタイプ実装について述べる．

A Version Data Model Based on the History Data Type

Hiroyuki Kitagawa,[†] Hajime Tanaka,[††] Nobuo Ohbo[†] and Isao Suzuki[†]

In database management of engineering design applications, version management is indispensable. Management of version derivation relationship and temporal information management are two important issues in version management. In this paper, we propose a version data model named TVDM (Temporal Version Data Model), which provides basic facilities for version management within the framework of the nested relational data model. TVDM features an abstractdata type “History Data Type” dedicated to integrated management of version derivation relationship and temporal information such as version creation/deletion times. TVDM provides an extended relational algebra named TV-Algebra. A database language named TV-QUEL is also proposed for data definition and manipulation in TVDM, and design and implementation of a prototype TV-QUEL interpreter based on POSTGRES is outlined.

1. はじめに

近年，CADシステムやソフトウェア開発等における計算機利用の高度化に伴い，設計データベースの重要性が認識されつつある．設計環境では，通常，設計作業の進捗，製品の改良，製品系列の拡充等の要因によって，時間の経過と共に様々な版（バージョン）が作成される．したがって，版管理は設計環境を支援するデータベースに必要不可欠な機能の一つである[11]．

版管理においては，ある版が他のどの版に基づいて生成されたかという版の導出関係の管理や，ある版がいつ生成・削除されたかという時間情報管理が重要である[14),20]．例えば，“あるシステムをリリースした時点におけるある関連部品の最新の版”や“ある一定期間におけるある部品仕様の変遷”を調べるようなデータ検索がしばしば要求される．このような要求に対応するには，指定された版の親（導出元）や子供（導出先）の版の間の導出関係と，指定された時刻における最新の版といった時間情報の両者が管理されていなければならない．

本論文では，抽象データ型[16),28]である“履歴型”を導入することにより入れ子型リレーショナルモデル (nested relational model)[8),13),22]の枠組みの中で，版の導出関係と時間情報の管理を可能とするデータモデルTVDM (Temporal Version Data Model) を提案する．本アプローチの特徴は以下のとおりである．

(a) 拡張リレーショナルモデル

従来，事務用データベースの主力として用いられてきたリレーショナルモデルに代わって，オブジェクト指向型モデルや関数型モデルに基づくデータベースシ

† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†† (株)電通
Dentsu Inc.

ステムが設計データ管理においては注目されている[6]．また，これらのデータモデルの枠組みの中で版管理を行うためのモデルが提案されている[1),3)~5),10)]．しかし一方で，入れ子型リレーショナルモデルや抽象データ型の導入等のリレーショナルモデルの拡張によるアプローチも，これまで蓄積されたリレーショナルデータベース技術に立脚した安定したデータ管理や行える点や従来データベースとの統合利用が図りやすい点等から，各種設計データ管理要求に対応する有力な方法の一つと考えられている[17),24),30)]．本アプローチでは，上に述べたようにリレーショナルモデルの基本的な枠組みの中で版管理機構の構築を図っている．リレーショナルモデルに基づく版管理に言及したこれまでの代表的な研究としては Batory ら[2)]や Dittrich ら[7)]によるものがあるが，版の導出関係や時間情報管理の方式やデータベース操作系等に関する明確な提案はなされていない．また，版管理に関連した領域の一つである時制データベース (temporal database) の分野においては，データベースの時間的変化のモデル化やデータ操作系の研究がリレーショナルモデルをベースに盛んに行われてきた[19),21),26),27),29)]．しかし，時制データベースにおける研究はいずれも時間変化の結果として生成される版の導出関係が線形である場合のみを対象としており，一般の設計データに見られる枝分れのある版の導出関係を対象としていない．

(b) 抽象データ型“履歴型”の導入

リレーショナルモデルの枠組みの中で版管理を行おうとした際まず考えられるのは，各版の生成時刻，削除時刻，親の版，子の版等を表す基本的な属性を付加する方式である．この場合，版管理に関する問い合わせ記述では，SQL や QUEL 等の通常のリレーショナルデータベース言語記述の中でこれらの属性に対する検索条件を指定する．しかるにこれらの問い合わせでは，導出関係や時間的生成順序関係に関して版の間をたどるような検索条件や，いつの時点を検索対象とするかといった時制データベース的な条件指定が必要となることが多い．このような場合，上記の基本的属性を付加するアプローチでは，これらの検索条件を付加された属性に関する基本的条件の組み合わせとして逐一書き下す必要があるため，問い合わせ記述が複雑化しがちである．また，ある版から導出されたすべての子孫の版を求めるような推移的閉包 (transitive closure) の検索も版管理では必須であるが，通常のリレーショナルデータベース言語では記述することができない．一般的な推移的閉包を扱うための言語拡張も提案されているが[15),32)]．問い合わせ記述をさらに複雑化させる要因にもなり得る．本アプローチでは，上記のような版管理用の基本的属性を付加する代わりに，版の導出関係と時間情報を一体として集約化した対象である履歴型を導入する．履歴型に付随する各種のオペレータを問い合わせ中で用いることにより，上に述べたような導出関係と時間情報に関する検索条件をより明示的に表現することを可能とする．(本論文の第4章で，この点に関する具体的な比較を示す.) また，このような履歴型の持つオペレータを用いて，版管理上必要な推移的閉包を求めることも可能である．

本論文の第2章では，TVDM におけるデータ記述とその代数操作系である TV 代数 (TV-Algebra) を示す．第3章では，TVDM に対するデータベース言語である TV-QUEL を提案しその概要を述べる．第4章では，リレーショナルモデルをベースに履歴型を用いずに版の導出関係と時間情報を表現する二つの方式を取り上げ，その場合の問い合わせ記述を TVDM (TV-QUEL) における場合と比較検討する．第5章では，拡張可能リレーショナル DBMS の一つである POSTGRES[23),24)]上での TV-QUEL インタプリタのプロトタイプ実装について述べる．最後の第6章では，結論と今後の課題を述べる．

2. TVDM (Temporal Version Data Model)

一般に，設計データベース中のデータは，それが記述する設計対象の変化に応じて各種の版を持つ．以下では，ある設計対象に対応する版の集合を版集合 (version set) と呼ぶ．**図1**は，ある実体型「システム」の Sys-A に関する版導出の過程を時間軸に沿って表したものである．図中のノードが版を表し，点線の矢印が導出関係を表す．例えば，Sys-A 第2版は時刻 t2 に Sys-A 第1版から導出されている．Sys-A に関するこれらの版の集合が一つの版集合である．各版集合は，版集合識別子 (version set identifier) により一意に識別される．また，ある版集合中の各版は版番号 (version number) により識別される．したがって，版集合識別子と版番号により，各版を識別することができる．

TVDM では，通常のリレーションにおける単純値属性に加えて単純値からなる集合属性値を許す入れ子型リレーションとそのタプルを用いて，データをモデ

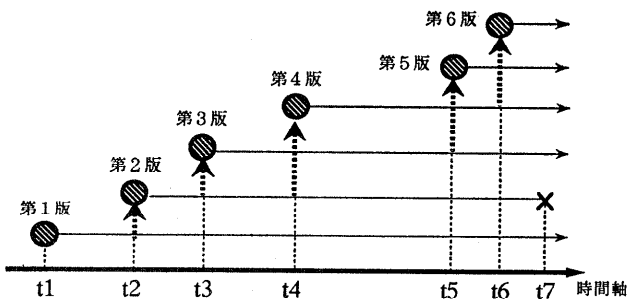


図 1 システム Sys-A の版導出過程
Fig. 1 Version derivation of system Sys-A.

ル化する．リレーションには，個々の版を表現するための版リレーション（version relation）と，各版集合の履歴を管理するための履歴リレーション（history relation）が存在する．前者は，1タプルで一つの版を表し，後者は1タプルで一つの版集合を表す．履歴に関する情報は履歴型の値を用いて管理する．履歴型は，一つの版集合に属する版の導出関係と各版の生成時刻・削除時刻を表現したデータ構造とその操作のためのオペレータを一体化した抽象データ型である．

以下の議論では，版の導出関係は木構造をなし，また各版の持つ属性値はその版を生成（導出）したトランザクション内で決定されるものとする．また，版が削除された時には，履歴にその版の削除時刻を記録し，その版から新たな版を導出することはできない．

2.1 データ型

TVDM には，単純値を表すための単純データ型（simple data type）と，集合値を表すための集合データ型（set data type）が存在する．

（1） 単純データ型

単純データ型としては，整数型，文字列型等の基本データ型に加え，版集合識別子型，版番号型，時刻型，履歴型を導入する．

（a）版集合識別子型

版集合識別子を表すデータ型で，適用可能なオペレータは値の等価性を判定する二項オペレータ＝とする．以下では，版集合識別子として整数文字列を用いる．

（b） 版番号型

版番号を表すデータ型で，適用可能なオペレータは値の等価性を判定する二項オペレータ＝とする．以下では，版番号として整数文字列を用いる．

（c） 時刻型

時刻を表すデータ型で，その取り得る値の集合は離散的な無限個の値からなる全順序集合とする[21),29)]．時刻型の値に適用可能なオペレータは，値の等価性や大小関係を判定する二項オペレータ＝，＜，＞とする．

（d） 履歴型

各版集合の履歴を表す内部データ構造とその操作のためのオペレータを一体とした抽象データ型．履歴型の内部データ構造は各版の版番号，親の版番号，子供の版番号，生成時刻，削除時刻を記録したもので，論理的には各版に関するこれらの情報を表す構造体を要素とする配列として表現可能である．**図2**に図1の履歴を表す履歴型の値の内部データ構造を示す．（実装例に用いたより具体的なデータ構造は第5章に示す．）ただし，この内部データ構造は利用者からは隠蔽されており，以下のオペレータを用いてのみ履歴型の値を操作可能である．（以下の説明では，引数 history，vno，t は，それぞれ履歴型，版番号型，時刻型の値を表す．）

〈参照用オペレータ〉

1) CREATED_TIME(history, vno)
版番号 vno の版の生成時刻を返す．

2) DELETED_TIME(history, vno)
版番号 vno の版の削除時刻を返す．

3) PARENT(history, vno)
版番号 vno の版の親の版の版番号を返す．

4) CHILD(history, vno, t)
時刻 t において版番号 vno の版から導出されていた子供の版の版番号を返す．返り値は版番号型の値の集合．

5) PREDECESSOR(history, vno)
版番号 vno の版の時間的直前に生成された版の版番号を返す．

6) SUCCESSOR(history, vno, t)
時刻 t において版番号 vno の版の時間的直後に

vno	parent	child	ctime	dtime
1	-	{2}	t1	-
2	1	{3,4}	t2	t7
3	2	{5}	t3	-
4	2	{6}	t4	-
5	3	{}	t5	-
6	4	{}	t6	-

図 2 履歴型の値の内部データ構造
Fig. 2 Internal data structure of a history type value.

生成されていた版の番号を返す.

7) CURRENT(history, t)

時刻tにおける最新の（最も新しく生成された）版の版番号を返す*.

下記8)〜11)のオペレータは，上記3)〜6)の各オペレータの推移的閉包（transitive closure）をとるオペレータであり，返り値はすべて版番号型の値の集合である.

8) PARENT*(history, vno)

9) CHILD*(history, vno, t)

10) PREDECESSOR*(history, vno)

11) SUCCESSOR*(history, vno, t)

〈更新用オペレータ〉

1) CREATE_ROOT(history, vno, t)

初版を登録する．上に述べた内部データ構造表現では，初版を表す構造体を登録し，その版番号 vno, 生成時刻tを記入する.

2) DERIVE(history, vno1, von2, t)

版番号 vno1 の版から版番号 vno2 の新しい版を導出し登録する．上記内部データ構造表現では，導出された新しい版を表す構造体を登録し，その版番号 vno2, 親の版番号 vno1, 生成時刻 t を記入すると共に，親の版の構造体に新しい版を子供として追加する.

3) DELETE(history, vno, t)

版番号 vno の版の削除を行う．上記内部データ構造表現では，版番号 vno の構造体に削除時刻tを記入する.

（2） 集合データ型

前述の各単純データ型の値集合のべき集合を，値集合とするデータ型である．単純データ型Dに対応する集合データ型を $P(D)$ で表す．集合データ型の値に適用可能なオペレータは，集合としての等価性を判定する $=$，集合同士の包含関係を判定する $\subset$, $\supset$，単純値を要素として含むかどうか判定する $\in$ とする.

2.2 リレーション

（1） リレーショナルスキーマ

属性 $A_1, \cdots, A_n$ からなるリレーショナルスキーマRを，$R=(A_1, \cdots, A_n)$ で表す．各属性 A_i には，一つのデータ型がドメイン（$Dom(A_i)$）として与えられる．TVDMでは，ドメインとして上に述べた単純データ型のみでなく集合データ型が許されることに注意されたい．R の属性の集合は Attr(R) で表す.

（2） リレーション

リレーション r は，組 $\langle R, I\rangle$ である．ここで，Rはリレーショナルスキーマであり，Iはリレーショナルインスタンスである．また，Iの要素であるタプルtに対し，$t[A_i]$ によってtの A_i 属性値を表す．また，属性値の部分列 $X=(A_{x1}, \cdots, A_{xm})$ に対して，$t[X]=(t[A_{x1}], \cdots, t[A_{xm}])$ とする.

2.3 リレーションの分類

TVDMには，既に述べたように版リレーションと履歴リレーションの2種類のリレーションが存在する．版管理をする対象の実体型に対して，版リレーションと履歴リレーションは必ず対でデータベース中に存在する．それぞれの定義は，以下のとおりである*.

（1） 版リレーション

ある実体型の各版を1タプルとして表現したリレーションで，以下のリレーショナルスキーマにしたがう.

R=(_vsetid, _vno, user-def-attr, ⋯)

- _vsetid：各版の属する版集合識別子を表す.
- _vno：各版の版番号を表す.
- user-def-attr：ユーザによって定義される属性.

（2） 履歴リレーション

ある実体型の各版集合を1タプルとして表現したリレーション．各版集合の履歴の情報を管理する．前述の版リレーションが定義されるとシステムによって，それに対応する履歴リレーションが自動的に生成される．リレーション名は，対応する版リレーション名の先頭に記号“_”を付けたものとする．以下のリレーショナルスキーマにしたがう.

_R=(_vsetid, _history)

- _vsetid：各版集合の版集合識別子を表す.
- _history：各版集合の履歴を表す．ドメインは履歴型.

〈例〉

図3に，版リレーション SYSTEMS と履歴リレーション _SYSTEMS の例を示す．版リレーション SYSTEMS は，ユーザ定義の属性として sname, spec, designer を持つ．版集合識別子1の版集合の履歴を表す history_1 は，図1の版導出過程を記録した図2の内部データ構造を持つ履歴型の値である.

上記の他，TVDMでは版管理を行わないデータをリレーションとしてデータベース中に保持することも

---

* 以下では，「カレント」な版をこの意味とする.

* システムが自動的に付加するリレーション名および属性名は，他のユーザ定義のものと区別するために，記号“_”で始まる名前を用いる.

SYSTEMS

_vsetid	_vno	sname	spec	designer
1	1	Sys-A	Spec-1	Tom
1	2	Sys-A	Spec-2	Tom
1	3	Sys-A	Spec-3	Tom
1	4	Sys-A	Spec-4	Marry
1	5	Sys-A	Spec-5	Jone
1	6	Sys-A	Spec-6	Marry
2	1	Sys-B	Spec-7	Michael
2	2	Sys-B	Spec-8	Michael
⋮	⋮	⋮	⋮	⋮

_SYSTEMS

_vsetid	_history
1	history_1
2	history_2
3	history_3
⋮	⋮

図 3 版リレーションと履歴リレーション
Fig. 3 Version relation and history relation.

可能である．これらのリレーションはすべてユーザ定義の属性からなり，版リレーションや履歴リレーションに関する上記の制約は存在しない．これらのリレーションを，版リレーションおよび履歴リレーションと区別してプレインリレーション（plain relation）と呼ぶ．

2.4 更新オペレーション

TVDM における更新オペレーションは，版リレーション（および履歴リレーション）に対するものと，プレインリレーションに対するものに大別できる．プレインリレーションに関する更新操作は，通常のとおりタプルの挿入，削除，属性値の変更である．以下に，版リレーションに対する更新用オペレーションを示す．relation は版リレーションを表すものとする．

1) Firstversion(relation, new_val_spec)

新しい版集合を生成しその初版を表すタプルを relation に挿入する．このタプルの _vsetid 属性値および _vno 属性値はシステムにより自動的に設定される．new_val_spec は初版の持つユーザ定義の属性の属性値を与えるリストで，対応する属性値がその値に設定される．また，履歴リレーション _relation には新たな版集合を表すタプルが挿入され，その _vsetid 属性値は上記 relation タプルの _vsetid 属性値と同じ値に設定される．その _history 属性値には 2.1 節で述べた CREATE_ROOT オペレータを内部的に用いて初版の版番号と生成時刻が登録される．上記 _vsetid 属性値と _vno 属性値の対が，本オペレーションの返り値となる．

2) Deriveversion(relation, vsetid, vno)

版集合識別子 vsetid，版番号 vno のタプルから新しい版を表すタプルを導出し，relation に挿入する．挿入するタプルの_vsetid 属性値には vsetid が設定され，_vno 属性値はシステムにより自動的に設定される．このタプルの持つユーザ定義属性値は本オペレーションの実行直後には版番号 vno のタプルと同じ値となるが，同一トランザクション内では以下に述べる Replaceversion オペレーションを用いて変更可能である．履歴リレーション _relation 中の版集合識別子 vsetid のタプルの _history 属性値には，2.1 節で述べた DERIVE オペレータを内部的に用いて導出した新しい版の版番号，生成時刻，導出関係が登録される．上記 _vno 属性値が，本オペレーションの返り値となる．

3) Replaceversion(relation, vsetid, vno, new_val_spec)

relation 中の版集合識別子 vsetid，版番号 vno のタプルのユーザ定義属性値を new_val_spec にしたがい変更する．システム定義の属性 _vsetid と _vno の値の書き換えは許されない．同一トランザクション内で上記の Firstversion または Deriveversion オペレーションを用いて生成したタプルの属性値変更のみが可能である．

4) Deleteversion(relation, vsetid, vno)

relation 中の版集合識別子 vsetid，版番号 vno のタプルを削除する．履歴リレーション _relation 中の版集合識別子 vsetid のタプルの _history 属性値には，2.1 節で述べた DELETE オペレータを内部的に用いて版番号 vno の版の削除時刻が登録される．

2.5 TV 代数

TVDM におけるデータ検索を行うための代数系を TV 代数（TV-Algebra）と呼ぶ．TV 代数は，標準的な入れ子型リレーショナルモデル[8),13),22)]におけるリレーショナル代数に準じたものであるが，2.1 節で述べた履歴型に付随するオペレータによるデータ検索を可能とするための Apply 演算子を追加した点が特徴である．TV代数の基本演算子は以下のものである．

1) Projection(π)

本演算子はリレーショナル代数の射影演算子に対応する．$r=\langle R, I\rangle$，$X=(A_{x1}, \cdots, A_{xm})$ を Attr(R) の部分列とすると，$\pi_X(r)=\langle R', I'\rangle$ は以下のとおり．

$R'=(A_{x1}, \cdots, A_{xm})$

$I'=\{t[X] \mid t \in I\}$

2) Selection(σ)

本演算子はリレーショナル代数の選択演算子に対応

するが，入れ子型リレーションでは集合値が許されるため選択条件式が拡張される．すなわち，$r=\langle R, I\rangle$ とし，X，Y をRの属性名または定数とする時，選択条件式pは，$X\theta Y$ またはこれに論理演算子 $\wedge$，$\vee$，$\neg$ を適宜適用した式とする．ただし，θ は以下のとおり．

(a) X，Yが共に単純データ型の時，θ はXとYのデータ型に定義された真偽値を返す二項オペレータ．

(b) Xが単純データ型でYが集合データ型の時，θ は $\in$．

(c) X，Yが共に集合データ型の時，$\theta\in\{=, \subset, \supset\}$．

pが $X\theta Y$ の時，$\sigma_p(r)=\langle R', I'\rangle$ は以下のとおりであり，pが $\wedge$，$\vee$，$\neg$ を含む場合も同様に $\sigma_p(r)$ は定義される．

$R'=R$

(a) X，Yが共に属性名である場合

$I'=\{t \mid t\in r\wedge t[X]\theta t[Y]\}$

(b) Xが属性名でYが定数の場合

$I'=\{t \mid t\in r\wedge t[X]\theta Y\}$

(c) Xが定数でYが属性名の場合

$I'=\{t \mid t\in r\wedge X\theta t[Y]\}$

3) Cartesian Product($\times$)

本演算子はリレーショナル代数の直積演算子に対応する．$r_1=\langle R_1, I_1\rangle$，$r_2=\langle R_2, I_2\rangle$ とし，R_1 の属性名と R_2 の属性名はすべて異なるものとする．この時，$r_1\times r_2=\langle R', I'\rangle$ は以下のとおり．

$R'=R_1\cdot R_2$

$I'=\{t_1\cdot t_2 t \mid {}_1\in I_1\wedge t_2\in I_2\}$

ただし，・は連接 (concatenation) を表す．

4) Set Union(U)

本演算子は集合としてのリレーションの和集合を求めるものである．$r_1=\langle R_1, I_1\rangle$，$r_2=\langle R_2, I_2\rangle$，R1=R2の時，$r_1 \cup r_2=\langle R', I'\rangle$ は以下のとおり．

$R'=R_1(=R_2)$

$I'=\{t \mid t\in I_1\vee t\in I_2\}$

5) Set Difference($-$)

本演算子は集合としてのリレーションの差集合を求めるものである．$r_1=\langle R_1, I_1\rangle$，$r_2=\langle R_2, I_2\rangle$，R1=R2の時，$r_1-r_2=\langle R', I'\rangle$ は以下のとおり．

$R'=R_1(=R_2)$

$I'=\{t \mid t\in I_1\wedge t\notin I_2\}$

6) Flat(μ)

本演算子は入れ子型リレーションの集合属性値を複数タプルに展開して単純属性値とするもので，Unnest演算子とも呼ばれる．$r=\langle R, I\rangle$，$R=(A_1, \cdots, A_{i-1}, A_i, A_{i+1}, \cdots, A_n)$ で，$Dom(A_i)$ が集合データ型の時，$\mu_{Ai}(r)=\langle R', I'\rangle$ は以下のとおり．

$R'=R$

$I'=\{t' \mid \exists t\in I\wedge t'[CA_i]=t[CA_i] \wedge t'[A_i]\in t[A_i]\}$

ただし，$CA_i=(A_1, \cdots, A_{i-1}, A_{i+1}, \cdots, A_n)$ であり，Rにおいて $Dom(A_i)=P(D)$ の時，R' において $Dom(A_i)=D$ となる．

7) Nest(ν)

本演算子は Flat とは逆に複数のタプルを集約することにより，単純属性値を集合属性値に変換する．$r=\langle R, I\rangle$，$R=(A_1, \cdots, A_{i-1}, A_i, A_{i+1}, \cdots, A_n)$ で，$Dom(A_i)$ が単純データ型の時，$\nu_{Ai}(r)=\langle R', I'\rangle$ は以下のとおり．

$R'=R$

$I'=\{t' \mid \exists t_1(t_1\in I\wedge t'[CA_i]=t_1[CA_i] \wedge \forall t_2(t_2\in I\wedge t_1[CA_i]=t_2[CA_i] \Rightarrow t_2[A_i]\in t'[A_i]))\}$

ただし，$CA_i=(A_1, \cdots, A_{i-1}, A_{i+1}, \cdots, A_n)$ であり，Rにおいて $Dom(A_i)=D$ の時，R' において $Dom(A_i)=P(D)$ となる．

8) Apply(α)

本演算子は履歴型をドメインとする属性を持つリレーションに適用する．2.1 節で述べた履歴型に付随する参照用オペレータと新たな属性名を指定して本演算子を適用すると，各タプルに対しそのオペレータの実行結果を新たな属性値として付加したリレーションが結果として得られる．_SYSTEM に対して各版集合の初版の生成時刻を求めるための Apply 演算 $\alpha_{CREATED\_TIME(\_history,1),ct1}$ を適用した結果得られるリレーションを**図4**に示す．形式的には，$r=\langle R, I\rangle$，$R=(A_1, \cdots, A_n)$ で，Fを履歴型に付随する参照用オペレータfとその引数の並び $f(arg_1, \cdots, arg_m)$ とする．ただし，$f: D_1\times\cdots\times D_m\rightarrow D$ とした時，$arg_i(1\leqq i\leqq m)$ は $Dom(A_j)=D_i$ なる属性 A_j またはドメイン D_i の

$\alpha_{CREATED\_TIME(\_history,1),ct1}(\_SYSTEMS)$

_vsetid	_history	ct1
1	history_1	t1
2	history_2	t8
3	history_3	t9
⋮	⋮	⋮

図 4 Apply 演算子の適用例
Fig. 4 Example of apply operation.

定数とする．この時，$\alpha_{F,A_{n+1}}(r)=\langle R', I'\rangle$ は以下のとおり．

$R'=(A_1, \cdots, A_n, A_{n+1})$

$I'=\{t \cdot y \mid t \in I \wedge y=f(arg'_1, \cdots, arg'_m)\}$

ただし，arg'_i は arg_i が定数の場合はその値そのものを，属性の場合は $t[arg_i]$ を表す．

上記の基本演算子の組み合わせとして，自然接合(⋈)，Set Intersection(∩) 等の演算子が定義される．Apply 演算子は履歴型の持つ各種オペレータの適用のために導入したものであるが，オブジェクト指向モデルに対する代数系の中でもメソッドの適用のための演算子が導入されている例がある[18),25)]．

〈TV 代数の適用例〉

以下に，TV 代数の適用例を示す．これらは，図3の SYSTEMS と _SYSTEMS に対する問い合わせである．

Q1. 版集合識別子1，版番号3の版の生成時刻と親の版の版番号を求める．

$$\pi_{ct3,pr3}(\alpha_{\text{PARENT}(\_history,3),pr3}(\alpha_{\text{CREATED\_TIME}(\_history,3),ct3}(\sigma_{\_vsetid=1}(\_SYSTEMS))))$$

Q2. 版集合識別子1，版番号3の版の時刻 t における子孫の版の版番号と仕様を求める．

$$\pi_{\_vno,spec}(\sigma_{\_vno \in ch3}(\alpha_{\text{CHILD*}(\_history,3,t),ch3}(\sigma_{\_vsetid=1}(\_SYSTEMS)) \bowtie \text{SYSTEMS}))$$

Q3. Sys-A 第6版の生成時刻における Sys-A 第2版の子供の版の版番号と設計者を求める☆．

$$\pi_{\_vno,designer}(\sigma_{\_vno \in ch2}(\alpha_{\text{CHILD}(\_history,2,ct6),ch2}(\alpha_{\text{CREATED\_TIME}(\_history,6),ct6}(\_SYSTEMS \bowtie \sigma_{sname=\text{"Sys-A"}}(\text{SYSTEMS})))))$$

3. データベース言語 TV-QUEL

本章では，TVDM に対するデータ定義，データ更新，データ検索を記述するための言語 TV-QUEL について述べる．TV-QUEL は，POSTQUEL[23),24)] の基本コマンドをベースとして設計されている．POSTQUEL が提供するリレーション操作用のコマンドには検索用の retrieve コマンドの他，create（リレーション生成），destroy（リレーション削除），insert（タプル挿入），replace（タプル属性値変更），delete（タプル削除）がある．TV-QUEL においても，プレインリレーションはこれらを用いることで従来どおり操作可能である．しかし，版リレーションは対応する履歴リレーションを伴っており，その更新に関しては2.4節で述べた更新オペレーションの体系に基づいた操作コマンドを提供することが望ましい．そこで，TV-QUEL では版リレーション（および履歴リレーション）操作のために下記のコマンドを追加して提供することとした．

1) vcreate

版リレーション（および履歴リレーション）の作成を行う．

2) vdestroy

版リレーション（および履歴リレーション）の削除を行う．

3) vfirstversion

初版の登録を行う．版集合識別子と初版の版番号が返り値となる．

4) vderive

新しい版の導出を行う．新たに導出された版の版番号が返り値となる．

5) vreplace

版リレーション中のタプルのユーザ定義の属性値の書き換えを行う．

6) vdelete

版の削除を行う．

検索用の retrieve コマンドのターゲットリストや where 節では，2.1 節の履歴型に付随する各種オペレータが記述可能である．Nest, Flat 以外の TV 代数の基本演算子によるデータ検索は，すべて retrieve コマンドで記述可能である．Nest, Flat 演算子に対応した操作に対応するため，TV-QUEL には nest, flat コマンドを用意している．

以下に，TV-QUEL による更新操作と問い合わせの例を示す．問い合わせは TV 代数の適用例と同じものを用いる．

〈更新操作例〉

M1. 版リレーション SYSTEMS（および履歴リレーション _SYSTEMS) を作成する．ただし，ユーザ定義の属性は，sname, spec, designer で，ドメインはすべて文字列 (char16) 型とする．

```
vcreate SYSTEMS(sname=char16, spec
        =char16, designer=char16)
```

この操作の結果，以下のような版リレーション SYSTEMS とそれに対応する履歴リレーション

☆ 属性 sname の値は版集合ごとに一意的と仮定する．

_SYSTEMS が生成される.

SYSTEMS(_vsetid, _vno, sname, spec, designer)

_SYSTEMS(_vsetid, _history)

M2. 版リレーション SYSTEMS に，Sys-A の初版の登録を行う.

```
vfirstversion SYSTEMS(sname="Sys-A",
                spec="Spec-1", designer
                ="Tom")
```

M3. Sys-A 初版から新たな版を導出する.（M2 の結果，版集合識別子および初版の版番号として共に1が返されたと仮定する.）

```
vderive S from S in SYSTEMS
where S._vsetid=1 and S._vno=1
```

M4. M3 で導出した版のユーザ定義の属性 "spec" の値を書き換える.（M3 の結果，新しい版の版番号として2が返されたと仮定する.）

```
vreplace S(spec="Spec-2")
from S in SYSTEMS
where S._vsetid=1 and S._vno=2
```

〈問い合わせ例〉

Q1. 版集合識別子1，版番号3の版の生成時刻と親の版の版番号を求める.

```
retrieve (ct3=CREATED_TIME
        (H._history, 3),
        pr3=PARENT(H._history, 3))
from H in _SYSTEMS
where H._vsetid=1
```

Q2. 版集合識別子1，版番号3の版の時刻 t における子孫の版の版番号と仕様を求める.

```
retrieve (S._vno, S. spec)
from S in SYSTEMS, H in _SYSTEMS
where S._vsetid=1 and H._vsetid=1
  and S._vno∈CHILD*(H._history, 3, t)
```

Q3. Sys-A 第6版の生成時刻における Sys-A 第2版の子供の版の版番号と設計者を求める.

```
retrieve (S._vno, S. designer)
from S in SYSTEMS, H in _SYSTEMS
where S.sname="Sys-A"
  and H._vsetid=S._vsetid
  and S._vno∈CHILD(H._history, 2,
    CREATED_TIME(H._history, 6))
```

4. 問い合わせ記述の比較

本章では，リレーショナルモデルをベースに履歴型を用いずに版の導出関係と時間情報を表現する二つの方式を取り上げ，その場合の問い合わせ記述を TVDM（TV-QUEL）における場合と比較検討する．ここで取り上げる表現方式の一つは，導出関係および時間情報を属性値としてリレーションに付加し，問い合わせ記述に SQL を用いた場合である．もう一つは，代表的な時制データモデルの一つである TRM (Temporal Relational Model) およびそのデータ操作言語 TSQL[21]を用いた場合である．以下では，図1に示した Sys-A の時間的変化を共通の例として用いる.

4.1 データ表現

オリジナルのリレーショナルモデルおよび TRM では各属性値としては単純値のみを許す第一正規型が仮定されるが，以下では TVDM におけるデータ表現との整合性のため属性値としては単純値の集合が許されるものとする.

(a) リレーショナルモデル

リレーショナルモデルに基づくデータ表現を**図5**に

SYSTEMS

vsetid	vno	parent	child	predecessor	successor	Ts	Tc	Te	sname	spec	designer
1	1	-	{2}	-	2	t1	t2	∞	Sys-A	Spec-1	Tom
1	2	1	{3,4}	1	3	t2	t3	t7	Sys-A	Spec-2	Tom
1	3	2	{5}	2	4	t3	t4	∞	Sys-A	Spec-3	Tom
1	4	2	{6}	3	5	t4	t5	∞	Sys-A	Spec-4	Marry
1	5	3	{}	4	6	t5	t6	∞	Sys-A	Spec-5	Jone
1	6	4	{}	5	-	t6	∞	∞	Sys-A	Spec-6	Marry
2	1	-	{2}	-	2	t8	t9	∞	Sys-B	Spec-7	Michael
2	2	1	{3}	1	3	t9	t10	∞	Sys-B	Spec-8	Michael
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

図5 リレーションによる表現
Fig. 5 Relational representation.

SYSTEMS

vsetid	vno	parent	child	predecessor	successor	current	delete	Ts	Te	sname	spec	designer
1	1	-	{}	-	-	true	false	t1	t2	Sys-A	Spec-1	Tom
1	1	-	{2}	-	2	false	false	t2	∞	Sys-A	Spec-1	Tom
1	2	1	{}	1	-	true	false	t2	t3	Sys-A	Spec-2	Tom
1	2	1	{3}	1	3	false	false	t3	t4	Sys-A	Spec-2	Tom
1	2	1	{3,4}	1	3	false	false	t4	t7	Sys-A	Spec-2	Tom
1	2	1	{3,4}	1	3	false	true	t7	∞	Sys-A	Spec-2	Tom
1	3	2	{}	2	-	true	false	t3	t4	Sys-A	Spec-3	Tom
1	3	2	{}	2	4	false	false	t4	t5	Sys-A	Spec-3	Tom
1	3	2	{5}	2	4	false	false	t5	∞	Sys-A	Spec-3	Tom
1	4	2	{}	3	-	true	false	t4	t5	Sys-A	Spec-4	Marry
1	4	2	{}	3	5	false	false	t5	t6	Sys-A	Spec-4	Marry
1	4	2	{6}	3	5	false	false	t6	∞	Sys-A	Spec-4	Marry
1	5	3	{}	4	-	true	false	t5	t6	Sys-A	Spec-5	Jone
1	5	3	{}	4	6	false	false	t6	∞	Sys-A	Spec-5	Jone
1	6	4	{}	5	-	true	false	t6	∞	Sys-A	Spec-6	Marry
2	1	-	{}	-	-	true	false	t8	t9	Sys-B	Spec-7	Michael
2	1	-	{2}	-	2	false	false	t9	∞	Sys-B	Spec-7	Michael
2	2	1	{}	1	-	true	false	t9	t10	Sys-B	Spec-8	Michael
2	2	1	{3}	1	3	false	false	t10	∞	Sys-B	Spec-8	Michael
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

図 6 TRM における表現
Fig. 6 Representation in TRM.

示す．各版を1タプルで表し，版管理のために属性 vsetid, vno, parent, child, predecessor, successor, Ts, Tc, Te を付加する．vsetid, vno, parent, child は，版集合識別子，版番号，親の版番号，子の版番号をそれぞれ示す．predecessor, successor は，時間的順序でその版の直前および直後に生成された版の版番号を示す．また，Ts, Tc, Te は，それぞれその版の生成時刻，カレントでなくなった時刻，削除時刻を表す．これらの属性の中には，他から導き出すことができるという意味で冗長なものが存在するが，問い合わせの便宜を重視してここではこのような設定とする．

(b) TRM

TRM に基づくデータ表現を**図 6** に示す．時制データモデルである TRM における表現では，版の一つの状態が1タプルで表現され，その状態の開始時刻と終了時刻を記録するための属性 Ts と Te が自動的に付加される．新たな子供の版の追加などの版の状態変化が起きた場合には，それまでの状態を表すタプルの Te にその変更時刻が代入されると共に，新たな状態を表すタプルがリレーションに追加される．ここでは版管理の目的のため，Ts と Te に加えてさらに，vsetid, vno, parent, child, predecessor, successor, current, delete を属性として付加する．current および delete 以外の属性の意味は(a)の場合と同様であるが，TRM では上に述べたようにこれらの属性値が変化した場合には新たな状態を表すタプルが付加される．属性 current はそのタプルが版がカレントであった状態を表す場合 true を，それ以外の場合 false をとるものとする．属性 delete はそのタプルが版が削除された状態を表す場合 true を，それ以外の場合 false をとるものとする．

4.2 SQL, TSQL, TV-QUEL の比較

ここでは，リレーショナルモデル，TRM, TVDM で表現されたデータに対する同一の問い合わせ例をそれぞれ SQL, TSQL, TV-QUEL で表現し，その問い合わせ記述を比較する．問い合わせ例は，版の導出関係と時間情報の両者に関する検索条件を伴う以下の二つとする．

Q4. 版集合識別子1の版集合において，時刻 t においてカレントであった版から時刻 t′ までに導出された子供の版の版番号を求める．

〈SQL による表現〉

```
SELECT S2.vno
FROM SYSTEMS S1, SYSTEMS S2
WHERE S1.vsetid=1 AND S1.Ts<=t
  AND t<S1.Tc AND S2.vsetid=1
  AND S1.vno=S2.parent
  AND S2.Ts<=t′
```

〈TSQL による表現〉

```
SELECT S2.child
FROM SYSTEMS S1, SYSTEMS S2
WHERE S1.vsetid=1
  AND S1.current=true
  AND S2.vsetid=1
  AND S2.vno=S1.vno
WHEN S1.INTERVAL OVERLAP t
  AND S2.INTERVAL OVERLAP t'*
```

〈TV-QUEL による表現〉

```
retrieve (vno=CHILD(H._history,
            CURRENT(H._history, t), t'))
FROM H in _SYSTEMS
WHERE H._vsetid=1
```

Q5. 版集合識別子1の版集合において，版番号3の版の時間的直後に導出された版の時刻 t における兄弟の版の版番号と設計者を求める．

〈SQL による表現〉

```
SELECT S3.vno. S3.designer
FROM SYSTEMS S1, SYSTEMS S2,
        SYSTEMS S3
WHERE S1.vsetid=1
```

```
typedef struct {  /* Element_Of_History型 */
  int vno;  /* 版番号 */
  int parent;  /* 親の版の版番号 */
  int num_child;  /* 子供の版の個数 */
  int child[MAX_CHILD];  /* 子供の版の版番号 */
  unsigned long ctime;  /* 生成時刻 */
  unsigned long dtime;  /* 削除時刻 */
} Element_Of_History;

typedef struct {  /* History型 */
  int num_element;  /* 版集合中の版の個数 */
  Element_Of_History element[MAX_H_ELEMENT];  /* 構造体Element_Of_Historyの配列 */
} History;

typedef struct {  /* Set_Of_Int型 */
  int num_element;  /* 集合中の要素数 */
  int element[MAX_I_ELEMENT];  /* 整数の配列 */
} Set_Of_Int

    :    :    :

Set_Of_Int *CHILD(history, vno, timepoint)
History *history;
int vno;
unsigned long timepoint;
{
  int i;
  Element_Of_History *ent, *cent;
  Set_Of_Int *result;

  result = (Set_Of_Int *) palloc(sizeof(Set_Of_Int));  /* POSTGRES中でのメモリ割当 */

    :    :    :

  result->num_element = 0;
  ent = find(history, vno);
  /* 版番号vnoのエントリを探索し存在すればそのポインタを，存在しなければNULLを返す．*/
  if (ent != NULL) {
    for (i =0; i < ent->num_child; i++) {  /* 全ての子供の版について繰り返し */
      cent = find(history, ent->child[i]);  /* 子供の版を探す */
      if (cent != NULL && cent->ctime <= timepoint)
          /* 子供の版がtimepointまでに導出されていたかチェック */
        result->element[result->num_element++] = ent->child[i];
    }
  }
  return(result);
}

    :    :    :
```

図 7 履歴型定義のためのCプログラム
Fig. 7 Program in C for history type definition.

* TSQL の WHEN 節は時間に関する条件記述に用いる．また，"S1.INTERVAL OVERLAP t" は意味的には "S1.Ts<=t AND t<S1.Te" と等価である．

```
    AND S1.predecessor=3
    AND S2.vsetid=1
    AND S2.vno=S1.parent
    AND S3.vsetid=1
    AND S3.parent=S2.vno
    AND S3.Ts<=t AND t<S3.Te
```

〈TSQL による表現〉

```
SELECT S3.vno, S3.designer
FROM SYSTEMS S1. SYSTEMS S2,
     SYSTEMS S3
WHERE S1.vsetid=1
  AND S1.predecessor=3
  AND S2.vsetid=1
  AND S2.vno=S1.parent
  AND S3.vsetid=1
  AND S3.vno∈S2.child
WHEN S2.INTERVAL OVERLAP t
  AND S3.INTERVAL OVERLAP t
```

〈TV-QUEL による表現〉

```
retrieve (S._vno, S. designer)
from S in SYSTEMS, H in _SYSTEMS
where S._vsetid=1 and H._vsetid=1
  and S._vno∈CHILD(H._history,
    PARENT (H._history,
    SUCCESSOR(H._history, 3, t)), t)
```

上記の問い合わせ記述例から以下のようなそれぞれの特徴が読み取れる．

SQL では，各版を表すタプルの属性値として版管理に関する情報が記述されているため，これらの属性に関するプリミティブな選択条件や接合条件の組み合わせとして検索条件を記述する必要がある．また，条件記述に必要なタプル変数も検索のナビゲーションのパスの長さにほぼ比例して増加する．

TSQL では WHEN 節と時間に関する条件記述用述語が与えられているため，SQL に比較すると時間的条件に関しては若干見通しが良い記述になっている．しかし，版の導出関係や時間的生成順序に関する検索条件を各属性に関するプリミティブな条件の組み合わせとして記述しなければならない点は，SQL の場合と変わらない．

上記の2方式に比較した場合，TV-QUEL では履歴型に付随する関数を用いることにより，導出関係や時間情報に基づく検索条件をより明示的に記述可能である．また，関数を入れ子にして用いることにより，検索条件記述に必要なタプル変数の数を低減している．上記2方式では，問い合わせ記述の便宜を重視して属性間に冗長性の多いデータ表現を用いたが，冗長性の少ない表現を用いた時には，SQL や TSQL での記述はより複雑になると考えられる．また，既に述べたように版の検索においては推移的閉包をとる操作が

```
TVQ> open demo_db
TVQ> retrieve(ct3 = CREATED_TIME(H._history, 3), \
              pr3 = PARENT(H._history, 3)) \
     from H in _SYSTEMS \
     where H._vsetid = 1

-------------------------------------------
| ct3                          | pr3      |
-------------------------------------------
| Thurs Dec 17 07:36:08 1992 JST| 2        |
-------------------------------------------

TVQ> retrieve (S._vno, S.designer) \
     from S in SYSTEMS, H in _SYSTEMS \
     where S.sname = "Sys-A" and H._vsetid = S._vsetid \
     and S._vno <- CHILD(H._history, 2, CREATED_TIME(H._history, 6))

-----------------------------
| _vno       | designer     |
-----------------------------
| 3          | Tom          |
-----------------------------
| 4          | Marry        |
-----------------------------

TVQ> vderive S from S in SYSTEMS where S._vsetid = 1 and S._vno = 6
_VNO: 7

   :       :       :
```

図 8 TV-QUEL インタプリタの実行例
Fig. 8 Session example of TV-QUEL interpreter.

しばしば必要となるが，TV-QUEL 以外の SQL や TSQL では通常このような推移的閉包をとることはできない．TV-QUEL では Q2 にその記述例を示したように，履歴型の関数を用いることでそれが可能である．

5. プロトタイプ実装

TV-QUEL インタプリタのプロトタイプを，拡張可能リレーショナル DBMS POSTGRES[23),24)]上で実装した．POSTGRES では，システムにあらかじめ定義されているデータ型に加え，ユーザが新たな抽象データ型を定義し属性のドメインとして用いることが可能である[28)]．

2.1 節で述べた TVDM に固有のデータ型の実現は次のとおりである．単純データ型のうち，時刻型は POSTGRES の組み込みデータ型である abstime 型を，版集合識別子型と版番号型は int4 型（4 バイト整数型）を用いた．履歴型はユーザ定義の抽象データ型として定義し，履歴型に付随する各種オペレータは C 言語でコーディングした手続きを，C 関数として POSTGRES 中に登録した．**図 7** に POSTGRES に履歴型を登録するための C 言語によるデータ定義とオペレータ CHILD のコードの一部を示す．第 2 章で述べたように，履歴型の内部データ構造は構造体 Element_Of_History の配列として表現している．CHILD はこの配列中を検索して与えられた時刻までに導出された子供の版番号を取り出すようコーディングしてある．これらの C 関数は，POSTGRES システム中では動的にリンクされて実行される．集合データ型も履歴型と同様に抽象データ型として定義し，∈，⊂，＝等のオペレータを C 関数として登録した．

プロトタイプでは，POSTGRES 上に TVDM のデータ操作をサポートするライブラリ LIB-TVQ を作成し，その上に TV-QUEL インタプリタを実装した．**図 8** に，TV-QUEL インタプリタの実行例を示す．

6. おわりに

本論文では，入れ子型リレーションモデルに版の導出関係と時間情報の管理を目的とした履歴型を導入したデータモデル TVDM を提案した．また，データベース言語 TV-QUEL を示し，リレーショナルモデルに基づく他の版情報管理法との問い合わせ記述の比較により，その有効性を示した．

今後の検討課題としては，以下のような項目があげられる．

（a） 合成オペレータの導入

本論文では，履歴型に付随するオペレータはプリミティブなものだけとし，その組み合わせで記述可能なオペレータについては定義していない．しかし，n 世代までの子孫を求めるといったようなある程度パターン化されたマクロな操作手順を合成オペレータとして記述可能とすることにより，問い合わせ記述の簡潔性の向上がより一層図れると考えられる．

（b） 履歴型の内部データ構造の検討

本論文におけるプロトタイプは，履歴型の内部データ構造として最も基本的な構造体の配列を用いている．しかし，抽象データ型の内部的データ構造は外部からは隠蔽されたものであり，履歴型のデータ構造としては他の方式を用いることも可能である．特に，データ構造を工夫することにより，グラフ構造中のナビゲーション，推移的閉包演算，時間情報検索等をより効率的に行える可能性もあり今後検討が必要である[9)]．

（c） 版リレーションと履歴リレーションの接合

TVDM では，版リレーションと履歴リレーションを導入しているが，第 3 章の問い合わせ例 Q3 に見られるようにこれらのリレーション間での接合演算がしばしば必要となる．この解決策としては，版リレーションと対応する履歴リレーションを接合したビューの導入や接合演算を支援する物理構造の検討が必要である．

（d） 構成管理などの検討

設計データ管理においては，版管理と密接に関連したトピックとして，種々の部品からなる対象の構成管理や長トランザクション管理が重要であることが知られている[2),10),12),31)]．今後，TVDM をベースとしたこれら情報の管理方式の検討が必要である．

謝辞 常日頃研究を進めるにあたり，ご助言，ご討議をいただいている筑波大学電子・情報工学系 藤原譲教授ならびにデータベース研究室の諸氏に感謝いたします．また，本論文の原稿を詳細に検討し的確なコメントをいただいた査読者の方々にお礼申し上げます．

参考文献

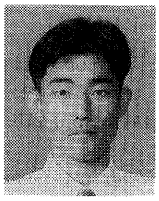
1) Ahmed, R. and Navathe, S. B.: Version Management of Composite Objects in CAD Databases, *Proc. ACM SIGMOD*, Denver, pp. 218-227 (1991).
2) Batory, D. S. and Kim, W.: Modeling Concepts for VLSI CAD Objects, *ACM Trans. Database Syst.*, Vol. 10, No. 3, pp. 322-346 (1985).
3) Beech, D. and Mahbod, B.: Generalized Version Control in an Object-Oriented Database, *Proc. 4th Int. Conf. on Data Eng.*, Los Angeles, pp. 14-22 (1988).
4) Chou, H. T. and Kim, W.: A Unifying Framework for Version Control in a CAD Environment, *Proc. 12th Int. Conf. on VLDB*, Kyoto, pp. 336-344 (1986).
5) Chou, H. T. and Kim, W.: Versions and Change Notification in an Object-Oriented Database System, *Proc. 25th ACM/IEEE Design Automation Conf.*, pp. 275-281 (1988).
6) Dittrich, K. R., Dayal, U. and Buchmann, A. P. (eds.): *On Object-Oriented Database Systems*, Springer-Verlag (1991).
7) Dittrich, K. R. and Lorie, R. A.: Version Support for Engineering Database Systems, *IEEE Trans. Softw. Eng.*, Vol. 14, No. 4, pp. 429-437 (1988).
8) Fischer, P. C. and Thomas, S. J.: Operations for Non-First-Normal-Form Relations, *Proc. IEEE COMPSAC 83*, Chicago, pp. 464-475 (1983).
9) Jagadish, H. V.: A Compression Technique to Materialize Transitive Closure, *ACM Trans. Database Syst.*, Vol. 15, No. 4, pp. 558-598 (1990).
10) Katz, R. H., Chang, E. and Bhateja, R.: Version Modeling Concepts for Computer-Aided Design Databases, *Proc. ACM SIGMOD*, Washington, D. C., pp. 379-386 (1986).
11) Katz, R. H.: Toward a Unified Framework for Version Modeling in Engineering Databases, *ACM Comput. Surv.*, Vol. 22, No. 4, pp. 375-408 (1990).
12) Kitagawa, H. and Ohbo, N.: Design Data Modeling with Versioned Conceptual Configuration, *Proc. IEEE COMSAC 89*, Orlando, pp. 225-233 (1989).
13) Kitagawa, H. and Kunii, T. L.: *The Unnormalized Relational Data Model—For Office Form Processor Design—*, Springer-Verlag (1989).
14) Klahold, P., Schlageter, G. and Wilkes, W.: A General Model for Version Management in Databases, *Proc. 12th Int. Conf. on VLDB*, Kyoto, pp. 319-327 (1986).
15) Kung, R. M. et al.: Heuristic Search in Data Base Systems, *Expert Database Systems*, Kerschberg, L. (ed.), Benjamin Cummings, pp. 537-548 (1986).
16) 姜 世杰ほか: CAD 支援を指向した複合対象抽象データ型の提案—ソリッド・データベースへの応用—, 情報処理学会論文誌, Vol. 31, No. 8, pp. 1194-1204 (1990).
17) Lohman, G. M. et al.: Extensions to Starburst: Objects, Types, Functions, and Rules, *Comm. ACM*, Vol. 34, No. 10, pp. 94-109 (1991).
18) Manola, F. and Dayal, U.: PDM: An Object-Oriented Data Model, *Proc. Int. Workshop on Object-Oriented Database Systems*, Asilmar (1986).
19) McKenzie, L. E. and Snodgrass, R.: Evaluation of Relational Algebra Incorporating the Time Dimension in Databases, *ACM Comput. Surv.*, Vol. 23, No. 4, pp. 501-543 (1991).
20) Navathe, S. B. and Ahmed, R.: Temporal Aspects of Version Management, *IEEE Data Eng.*, Vol. 11, No. 4, pp. 34-37 (1988).
21) Navathe, S. B. and Ahmed, R.: A Temporal Relational Model and a Query Language, *Inf. Sci.*, No. 49, pp. 147-175 (1989).
22) Ozsoyoglu, G., Ozsoyoglu, Z. M. and Matos, V.: Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions, *ACM Trans. Database Syst.*, Vol. 12, No. 4, pp. 566-592 (1987).
23) *POSTGRES Reference Manual*, Univ. of California, Berkeley (1991).
24) Rowe, L. and Stonebraker, M.: The POSTGRES Data Model, *Proc. 13th Int. Conf. on VLDB*, Brighton, pp. 83-96 (1987).
25) Shaw, G. M. and Zdonik, S. B.: A Query Algebra for Object-Oriented Databases, *Proc. 6th Int. Conf. on Data Eng.*, Los Angeles, pp. 154-162 (1990).
26) Snodgrass, R. and Ahn, I.: Temporal Databases, *IEEE Computer*, Vol. 19, No. 9, pp. 35-42 (1986).
27) Snodgrass, R.: The Temporal Query Language TQuel, *ACM Trans. Database Syst.*, Vol. 12, No. 2, pp. 247-298 (1987).
28) Stonebraker, M., Rubenstein, B. and Guttman, A.: Application of Abstract Data Types and Abstract Indices to CAD Data Bases, *Pcoc. Annual Meeting Database Week*, San Jose, pp. 107-115 (1983).
29) Tansel, A. U.: Adding Time Dimcnsion to Relational Model and Extending Relational

Algebra, *Inf. Syst.*, Vol. 11, No. 4, pp. 343-355 (1986).
30) The Committee for Advanced DBMS Function: The Third-Generation Database System Manifesto, *ACM SIGMOD Record*, Vol. 19, No. 3, pp. 31-44 (1990).
31) Xu, H. and Kambayashi, Y.: Element-Relationship-Mapping Model and Its Application to Software Development, *Proc. IEEE COMPSAC 87*, Tokyo, pp. 390-396 (1987).
32) Zloof, M. M.: Query-by-Example: Operations on the Transitive Closure, *IBM Res. Rep.*, RC 5526 (1976).

(平成4年7月20日受付)
(平成5年3月11日採録)

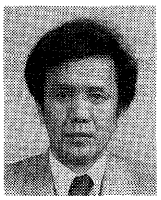
北川 博之（正会員）

1955年生．1978年東京大学理学部物理学科卒業．1981年同大学院理学系研究科博士課程退学．日本電気(株)勤務を経て，1988年筑波大学電子・情報工学系講師，現在，同学系助教授．理学博士（東京大学）．データベースシステム構成法，時間データ管理，エンジニアリングデータ管理，ソフトウェア開発支援システムなどに興味を持つ．著書「The Unnormalized Relational Data Model」(共著, Springer-Verlag). 電子情報通信学会，ACM，IEEE-CS 各会員．



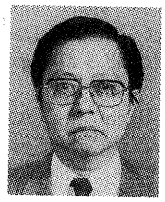
田中 肇（正会員）

昭和42年生．平成2年筑波大学第三学群情報学類卒業．平成4年同大学理工学研究科修士課程修了．現在，(株)電通に勤務．平成4年度本学会研究賞受賞．



大保 信夫（正会員）

昭和20年6月21日生．東京大学理学部卒業．理学博士．筑波大学電子・情報工学系勤務．研究テーマ：データベースシステム．



鈴木 功（正会員）

昭和8年生．昭和32年東京大学理学部化学科卒業．昭和37年同大学院化学系研究科博士課程修了．理学博士．同年ミネソタ大学博士研究員．昭和39年東京大学理学部助手．以後，同大学理学部講師，教育用計算機センター講師，助教授を経て，昭和53年から筑波大学電子・情報工学系教授．データベースシステム，情報検索等の研究に従事．人工知能学会，日本教育工学会，日本化学会，日本科学教育学会各会員．