データベース・レコードの仮想実体化フレームワーク

大東 誠[†,††] 田中 譲[†,††]

近年，3D の CG 技術を利用して DB に蓄積された大量データの解析・利用を支援するインタラクティブ視覚化の重要性が増している．従来の視覚化システムは，データの種類に特化した視覚化スキーム（データの表現法や配置法）と，この視覚化スキームに依存する数種のインタラクションを提案している．これらのシステムでは，あらかじめ定義された数種の視覚化スキームとインタラクションしか使用できないため，ユーザによる柔軟なデータの解析・利用操作が制限される．本稿では，ユーザによるカスタマイズが可能な任意の視覚化スキームを用いて，様々な DB レコードをインタラクティブな 3D オブジェクトとして仮想空間内に実体化するためのフレームワークを提案する．このフレームワークは 3D 部品アーキテクチャを基盤とする．ユーザは 3D 部品を組み立てて任意の視覚化スキームを定義し，様々な DB レコードを仮想実体化することができる．本研究では，RDB レコードに限らず，入れ子構造等，複雑な構造を持つ属性を含む DB レコードの視覚化スキームも，部品合成により定義可能な構成法を提案する．これらを用いて DB レコードを仮想実体化することにより，ユーザは仮想空間や仮想実体に対して，解析・利用目的に応じた任意のアプリケーション・ツールを導入し，それらと自由に連携することが可能となる．

A Framework for the Virtual Reification of Database Records

Makoto Ohigashi[†,††] and Yuzuru Tanaka[†,††]

In this paper, we propose a generic visualization framework that reifies database records as interactive 3D components. Conventional information visualizations are proposing data-specific visualization schemes (view designs, the ways of arrangement of a large set of data, and the interactions with them). We propose a generic visualization framework that allows users to define their visualization schemes and to reuse the visualized records and the result data of analysis. Our proposed framework provides many forms of user interaction and the reusability of visualized objects. A 3D component-ware architecture provides the basis of our virtual reification framework. Our framework provides a dynamic view-definition and view-modification mechanism so that users can easily change the view of records or distribution of records to possibly find a significant structure among them. The framework also presents both retrieved records and the results of their data analysis as virtually reified objects so that users can easily send them to others or reuse them in other environmens.

1. はじめに

近年，3D の CG 技術を利用して DB に蓄積された大量データの解析・利用を支援するインタラクティブ視覚化の重要性が増している．従来の視覚化システムでは特定のデータ・ドメインを対象とし，それに対する視覚化スキーム（データ表現とデータ配置法）を提案している．データの表現法としては，色や大きさの異なる立方形状[4),7),10)]や属性値に応じて長さの異なる枝付き形状で表す手法[11)]等が提案されている．またデータの配置法としては，1D[8),9)]，2D[1),12)]，3D[5),6),16)]に配置する手法やバネモデルを用いて 2D[7)]や 3D[10),11)]に配置する方法等が提案されている．

これらの視覚化スキームはシステムが視覚化対象とするデータ・ドメインに特化されているため，システムが扱う特定のデータに対してユーザの要求する視覚化スキームが提供されていない場合，ユーザはシステムの変更，もしくは開発を強いられる．これを解決するためには，視覚化スキームのみを提供するのではなく，カスタマイズ可能な視覚化スキームを提供することにより，それを用いてユーザが視覚化システムを構築可能な仕組みを提供する必要があると考えた．

また，従来の視覚化システムは DB データを 3D の CG 技術やアニメーション技術を用いて単に「表示」をしている．これらのシステムにおいて，ユーザには「表示」されたデータに対して，付加情報や関連情報

† 北海道大学大学院工学研究科電子情報工学専攻
Graduate School of Engineering, Hokkaido University
†† 北海道大学知識メディアラボラトリー
Meme Media Laboratory, Hokkaido University

の要求[7),10),13)]，他のビューへのナビゲーション[14)]等のため，あらかじめ定義されたインタラクションのみが許されている．インタラクション手法はシステムの提案する視覚化スキームに強く依存するため，特定の視覚化スキームのみを提案する従来の視覚化システムにおいては，ユーザは視覚化されたデータに対してあらかじめ設定された操作しか行うことができない．

ユーザによる柔軟な DB レコードの解析・利用操作を支援するためには，目的に応じたアプリケーション・ツールの導入，および，これらのツールと DB レコードとの自由な連携が必要であると考える．これを実現するためには，まず DB レコードを単に可視化するだけではなく，仮想的な「もの」として仮想空間内に実体化する必要がある．これにより，仮想実体化された DB レコードは単なるデータではなく，各種の event に反応したり様々な message を受け取り実行することが可能なインタラクティブ・オブジェクトとして扱うことが可能となる．さらに，仮想実体化された DB レコードと各種アプリケーション・ツールを連携させる標準的なプロトコルが必要となる．これにより，仮想実体化された DB レコードを任意のアプリケーションと組み合わせて再利用することが可能となる．たとえば，個人情報データをある属性値に応じて歩行速度の異なる仮想の人として視覚化したとする．この仮想の人を直接操作してコピーし，仮想都市にドロップして群衆シミュレーションを行う等，直接操作による再利用が可能となる．

本稿では，汎用 DBMS に蓄積された各種データを仮想的な「もの」として仮想空間内に実体化するためのフレームワークを提案する．提案するフレームワークは 3D 部品アーキテクチャを基盤とし，部品合成による視覚化スキームの定義とそれに基づく視覚化システムの構築を可能とする．本研究では RDB レコードに限らず，入れ子構造等，複雑な構造を持つ属性を含むレコードの視覚化スキームも部品合成により定義可能な構成法を提案する．ユーザによる DB レコードの柔軟な解析・利用操作を支援するため，仮想空間や DB レコードの仮想実体に対する任意のアプリケーション・ツールの導入，および，それらのツールと実体化された DB レコードとの連携のための枠組みを示す．

本稿では以下，2 章では提案する仮想実体化フレームワークの概要について述べ，3 章で，フレームワークの基盤として用いる 3D 部品アーキテクチャについて説明する．4 章では，仮想実体化フレームワークの実現アーキテクチャについて詳細を述べる．5 章で，仮想実体化のテンプレートとなる表現モデルについて述べ，6 章で仮想実体化例を紹介してその有用性を示す．

2. 仮想実体化フレームワークの概要

この章では，仮想実体化フレームワークについて概要を述べる．まず，2.1 節で，提案する仮想実体化フレームワークを構成する要素機能について述べ，2.2 節で仮想実体化フレームワークの実装システムに必要な条件について議論する．

2.1 仮想実体化フレームワークの要素機能

本稿で提案する仮想実体化フレームワークは，以下の 5 つの要素機能から構成される．

(i) DB レコードの取得
(ii) 仮想実体の生成
(iii) レコード表現モデル
(iv) 仮想実体集合の空間属性管理
(v) レコード集合に対する演算と結果の実体化

(i) は，DB レコードをシステム内に取り込み，利用可能にする機能である．DBMS との通信機能と検索条件の指定，検索結果の保持が必要となる．

(ii) は，検索結果レコードとそのレコード表現とを結合する機能である．レコード表現を基に検索結果レコードを仮想実体として提供する．

(iii) は，各レコードの実体化のテンプレートとなる合成インタラクティブ部品である．レコードの各属性とその表現部品とのマッピングが定義されている．

(iv) は，実体化されたレコード集合の配置，配色等，その空間属性の管理を行う機能集合である．各機能を組立て可能な要素部品集合に分解し，その組立て規則とともに提供する．

(v) は，検索結果レコード集合に対する演算機能集合である．必要に応じて導入可能な演算機能を持つ部品集合を提供する．各部品は，演算結果データ集合を仮想的な実体として利用可能にするため，これらを必要に応じて部品化する機能を提供する．

図 1 に示すように，本研究では，ユーザが対象デー

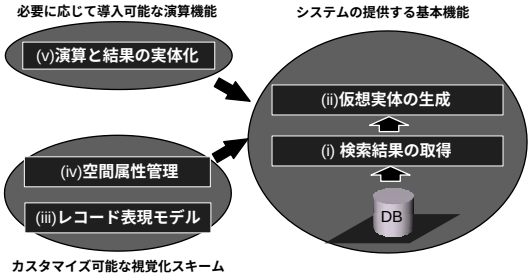


図 1 仮想実体化フレームワークの要素機能
Fig. 1 The basic functions of the virtual reification framework.

タに応じた適切な仮想実体化システムを構築可能とするため，(iii) の「レコード表現モデル」と (iv) の「仮想実体集合の空間属性管理」機能の定義をユーザに委ね，これらと (v) の「演算と結果の実体化」機能を動的にシステムに組込み可能な構造を提案する．

2.2 仮想実体化システムの必要条件

視覚化スキームをカスタマイズして各種 DB データを視覚化し，かつ，それらを仮想的な実体として様々な用途に再利用するためには，仮想実体化システムは以下の条件を満たす必要がある．

(i) 視覚化スキームのカスタマイズ

視覚化スキームは，特定データの多面的観察ならびに異なる DB データの視覚化のためにカスタマイズできる必要がある．また，カスタマイズを効率良く行うため，システムは視覚化スキームを容易にカスタマイズする GUI と，カスタマイズ結果をモードを切り換えずに確認できる仕組みを提供することが望まれる．

(ii) 多様なレコード表現定義

システムは，レコード集合の多面的観察の支援，ならびに DB スキーマやユーザの関心点に応じた適切な表現の定義を可能とするため，多様なデータ表現要素の提供，およびこれらの自由な組合せにより複雑なデータ表現を定義可能な機構を提供する必要がある．

(iii) データの仮想実体化と再利用

データを仮想的な実体として利用するため，システムはこれらを直接操作可能な可視化部品として提供する．この部品化された DB データを任意のアプリケーションと組み合わせて再利用可能とするため，この部品は (iii-1) データを内部に保持する部品構造と (iii-2) 標準化された部品間のデータアクセス・インタフェースを持つ必要がある．(iii-1) により，個々の DB データを独立した「もの」として再利用でき，(iii-2) により，他のアプリケーションとの機能連携が可能となる．

(i) と (ii) により，ユーザは任意の DB データを適切なスキームを用いて視覚化できる．また，(iii) により，視覚化されたデータを仮想的な実体と扱うことが可能となり，解析・利用目的に応じた種々のツールの導入，および，それらとの自由な連携が可能となる．

3. 3D 部品アーキテクチャ

2.2 節で示した条件を満たすため，著者らは IntelligentBox システム[15),17)] の 3D 部品アーキテクチャを基盤として仮想実体化フレームワークの実装を行った．

IntelligentBox は，マルチメディア・データやアプリケーション・ツール等，計算機上のあらゆるオブジェクトを直接操作可能な可視化部品（ボックスと呼ぶ）として提供する．ユーザはボックスを組み合わせて，より複雑なデータ構造や機能を持つボックスを定義できる．DB レコードを多様なボックスを用いて定義することにより，2.2 節の (ii)「多様なレコード表現定義」の条件を満たす．5.2 節で詳細を述べる．

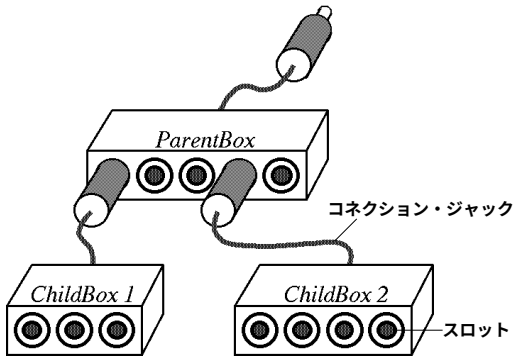


図 2 ボックスの機能連携イメージ
Fig. 2 A schematic explanation of functional linkages among boxes.

各ボックスは，状態データを保持する任意個のスロット☆と呼ぶ内部オブジェクトを持つ．スロットは他のボックスとのデータ通信のためのインタフェースとなる．IntelligentBox では，部品間の機能連携プロトコルが標準化されており，合成部品を含めた様々なボックス間で機能連携を定義できる．DB レコードをボックスとして提供することにより，スロットによる DB データの保持，ならびに，標準的なアクセス・インタフェースの提供を実現し，2.2 節の (iii)「データの仮想実体化と再利用」の条件を満たす．

ボックス間の機能連携は，親子関係を定義することにより可能となる．親子関係は，親と子となる 2 つのボックスを指定して定義する．また，子ボックスは親ボックスの任意のスロットの 1 つと結合し，結合されたスロットを通してデータ授受が行われ機能が連携される．結合するスロットは，メニューで決定される．**図 2** では，この結合をコネクション・ジャックで示す．機能連携されたボックス集合は「合成ボックス」と呼ばれ，他のボックスと同等に扱うことができる．

合成ボックスの定義に限らず，分解，部分修正等の編集作業は画面上で直接操作して行い，それらの結果動作はモードを切り換えることなく画面上で確認できる．視覚化スキームのカスタマイズを合成ボックスの編集により行うことで，2.2 節の (i)「視覚化スキームのカスタマイズ」の条件を満たす．カスタマイズ手法については，5.3 節で詳細を述べる．

☆ 本稿では，スロットを#slotname と表記する．

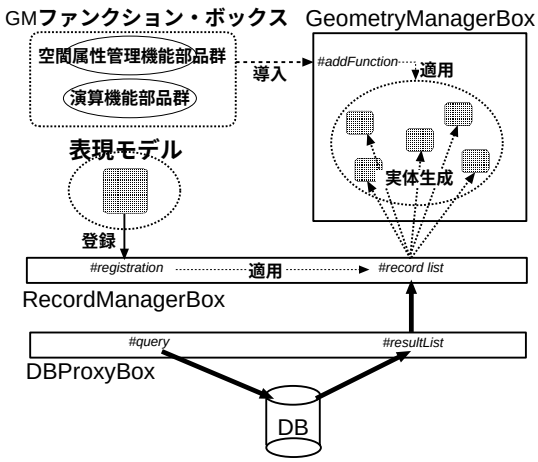


図 3 フレームワークの実現アーキテクチャ
Fig. 3 The proposed framework architecture.

表 1 フレームワーク要素機能の部品化
Table 1 Required functions and their corresponding components in our framework.

要素機能		部品
(i) レコード取得	→	DBProxyBox
(ii) 仮想実体生成	→	RecordManagerBox
(iii) 表現モデル	→	ユーザ定義合成ボックス
(iv) 空間属性管理	→	GeometryManagerBox +
(v) 演算と部品化		GM ファンクション・ボックス

4. フレームワークの実現アーキテクチャ

IntelligentBox を基盤システムとした仮想実体化フレームワークの実現アーキテクチャを**図 3** に示す．2.1 節で述べたフレームワーク要素機能を部品化し，それらを組み合わせて実現している．

この章では，部品化したフレームワーク要素機能について述べ，それらの機能合成によるレコード集合の仮想実体化プロセスを説明する．

4.1 フレームワーク要素機能の部品化

表 1 にフレームワーク要素機能の部品化一覧を示す．(i) の「DB レコードの取得」機能と (ii) の「仮想実体の生成」機能はそれぞれ後述の DBProxyBox と RecordManagerBox により部品化する．IntelligentBox における (iii) の「レコード表現モデル」は合成ボックスをテンプレートとして用いることに相当する．このため IntelligentBox で実装するフレームワークではこれを提供せず，ユーザによる定義表現を組込み可能な機構のみを提供する．表現モデルについての詳細は第 5 章で述べる．(iv) の「実体化レコード集合の空間属性管理」機能と (v) の「演算と結果の部品化」機能は，これらを組立て可能な要素機能に分解し，後述の GM ファンクション・ボックスにより部品化する．これらの機能を仮想実体集合に作用させる部品が後述の GeometryManagerBox である．

表 2 DBProxyBox のスロット一覧
Table 2 The slot list of a DBProxyBox.

#class	アクセスするクラス名
#query	SQL 文
#resultList	検索結果レコード配列
#currentRecord	1 レコード情報
#next, #previous	currentRecord 選択
#search	検索開始トリガ
#insert	currentRecord の挿入トリガ
#delete	currentRecord の削除トリガ

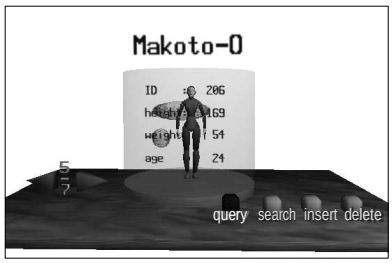
4.1.1 DBProxyBox

DBProxyBox は外部 DBMS とのインタフェース機能を持つ．DB に対して検索・挿入・削除要求を送信しその結果を受信する．SQL 文等のアクセス情報とアクセス結果は DBProxyBox 内部のスロットに保持される．**表 2** に DBProxyBox のスロット一覧を示す．

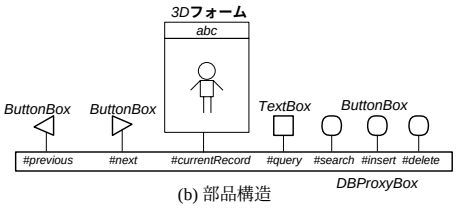
〈検索機能〉 DBProxyBox はスロット #search へのアクセスをトリガとしてスロット #query に保持する検索文を DBMS に送信し，受信した評価結果をレコード配列としてスロット #resultList に格納する．フレームワークはこのスロット値を利用する．スロット #currentRecord にはスロット #resultList のレコード配列中の 1 レコードが格納される．格納するレコードはスロット #next，スロット #previous で選択される．SQL 文はスロット #query にテキスト入力部品である TextBox 等を結合して入力する．

〈挿入・削除機能〉 DBProxyBox は検索機能のほか，レコードの挿入・削除機能を持つ．DBProxyBox はスロット #delete へのアクセスをトリガとしてスロット #currentRecord に選択されたレコードをスロット #class で指定されるクラスから削除する delete 文を DBMS に送信する．また，スロット #insert へのアクセスをトリガとしてスロット #currentRecord に入力されたレコードをスロット #class で指定されるクラスに挿入する insert 文を DBMS に送信する．

〈3D フォームインタフェース〉 提案するフレームワークではレコード配列を保持するスロット #resultList を利用する．まず最初にスロット #currentRecord を利用して 1 レコードの視覚化・挿入・削除を行う 3D フォームインタフェースを提案する．画面ハードコピーと部品構造を**図 4** に示す．3D フォームは 5 章で述べるレコード表現モデルと同じ構造を持つ．図 4 (b) 左にあるスロット #next，スロット #previous に結合された ButtonBox で表示レコードを選択し，ス

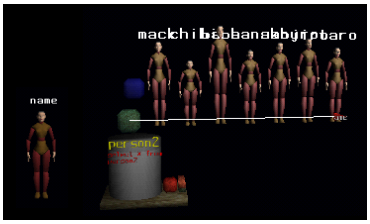


(a) 画面ハードコピー

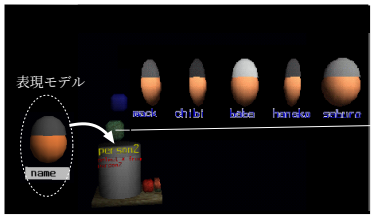


(b) 部品構造

図 4 3D フォームインタフェース
Fig. 4 A display hardcopy and the composition structure of a 3D form interface.



(a) 人型形状を持つレコード表現



(b) 顔型形状を持つレコード表現

図 5 表現モデル交換によるレコード表示の更新
Fig. 5 A dynamic replacement of a doll model with a face model for representing each record.

ロット#delete に結合された ButtonBox をクリックすると DB からそのレコードが削除される．また，3D フォームの各属性の値を変更し，スロット#insert に結合された ButtonBox をクリックすると DB に新たなレコードが挿入される．

4.1.2 RecordManagerBox

RecordManagerBox は，登録された表現モデルに基づき各レコードを表示する機能を持つ．RecordManagerBox は，レコード表現モデルを保持するスロット#registration とレコード配列を保持するスロット#recordList を持つ．スロット#recordList にレコード配列が格納されると各レコード情報を持つレコードスロットをレコード数だけ生成する．同時に，登録されている表現モデルの複製を同数生成し，レコードスロットに結合する．各複製オブジェクトは，結合されたレコードスロットからレコード情報を読み出し，その値に応じた表示を行う．スロット#registration に新規の表現モデルが登録されると，RecordManagerBox は表示しているレコード実体をすべて削除し，登録された表現モデルの複製を各レコードスロットに結合して仮想実体を再生成する．

図 5 (a) は，人の形状を持つ表現モデルを用いた個人情報データの表示結果を表している．height 属性を形状の高さで表現している．図 5 (b) は，同じデータに対し顔型の表現モデルをスロット#registration に登録した結果を表している．weight 属性を形状の幅，age 属性を頭部の色で表現している．表現モデルの交換は画面上での直接操作により行われ，モードを切り換えることなくその結果が表示される．

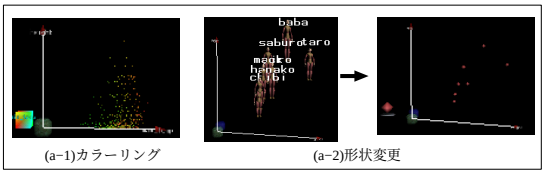
表 3 GM ファンクション・ボックスの種類とその部品化例
Table 3 The classification of geometrical functions povided as GM function boxes.

機能の種類	機能部品例
(a) intrinsic 型	Color, Scale, Transparency, SwapShape etc.
(b) extrinsic 型	CoordinateOrigin, CoordinateAxis, SpringModelArrangement etc.
(c) operation 型	RegionSelection, VisualProjection HilightSelection etc.

4.1.3 GM ファンクション・ボックス

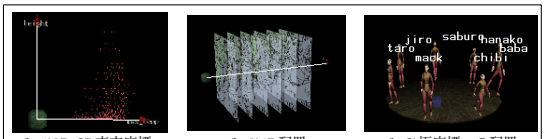
実装したフレームワークは，仮想実体集合の配置やこれらに対する演算機能を，組立て可能な要素機能部品群に分解して提供する．GM ファンクション・ボックスは，2 種の空間属性管理機能（intrinsic 型，extrinsic 型）と演算機能（operation 型）に分類される．**表 3** に GM ファンクション・ボックスの分類とその部品化例を示す．詳細は以下で述べる．ユーザは GM ファンクション・ボックスを組み立て，自由に配置法を定義したり演算を導入することができる．

〈空間属性管理機能〉 空間属性は形状や色，透明度，大きさ等を表す intrinsic 属性と位置を表す extrinsic 属性とに分類される[3]．intrinsic 属性の管理機能としては，任意の属性値に基づきカラーリングやスケーリング，透明度の変更を行う機能やデータ全体の概観を得るため各レコードを単純形状で表示する機能等がある．表 3 (a) の部品例は，これらの機能を部品化したものである．これらの一部の部品機能例を**図 6** (a) に示す．extrinsic 属性の管理機能としては，直交座標の各座標軸や極座標の動径や回転軸，レコードの類似度



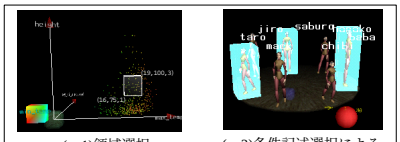
(a−1)カラーリング (a−2)形状変更

(a) intrinsic型



(b−1)2D, 3D直交座標への配置 (b−2)1D配置 (b−3)極座標への配置

(b) extrinsic型



(c−1)領域選択 (c−2)条件記述選択によるハイライト

(c) operation型

図 6 各種 GM ファンクション・ボックスの機能例
Fig. 6 Example functions provided by different types of GM function boxes.

に基づく近傍配置機能等がある．表 3 (b) の部品例は，これらの機能を部品化したものである．これらの一部の部品機能例を図 6 (b) に示す．

〈演算機能〉 operation 型 GM ファンクション・ボックスは，演算機能とその結果データ集合の実体化機能を提供する．GM ファンクション・ボックスは，演算の種類（選択，射影，結合等）と手法（視覚的と非視覚的），演算結果の表示方法（選択結果のハイライトや色の反転等）を基に部品化する．表 3 (c) に部品化例を示す．たとえば選択演算部品は，図 6 (c-1) に示すように，extrinsic 型 GM ファンクション・ボックスにより配置された実体化集合から空間領域を指定して視覚的に選択を行う部品や，図 6 (c-2) に示すように，条件を直接記述してその結果レコードをハイライトさせる選択部品等がある．operation 型 GM ファンクション・ボックスは，演算結果データを内部スロットに保持し，ユーザの要求に応じてこれらを部品化する．

4.1.4 GeometryManagerBox

GeometryManagerBox は，スロット #addFunction に登録された GM ファンクション・ボックスの機能を起動して実体化集合に適用する．GM ファンクション・ボックスの持つ機能は，スロット #addFunction へのスロット結合により追加され，結合解除により削除される．GM ファンクション・ボックスの追加・削除・パラメータ変更等のカスタマイズ結果は，即座に表示に反映される．

図 7 に GM ファンクション・ボックスのカスタマイズ例を示す．図 7 (a) は GeometryManagerBox に直交座標の原点を示す部品 CoordinateOriginBox と，異なるパラメータを設定した 2 つの座標軸部品 CoordinateAxisBox を登録して検索した結果である．図 7 (b) では，スロット #addFunction にもう 1 つ座標軸部品（extrinsic 型）とカラーリング部品（intrinsic 型）を追加して 3D 座標へプロットしている．図 7 (c) では，これに領域選択部品（operation 型）を追加して選択範囲を決定し，領域内のレコード集合を実体化している．図 7 (d-1) に示すように，実体化されたレコード集合は他の任意のツールへドロップして詳細を表示したり，図 7 (d-2) で示すように，座標配置部品等，他のツールを導入して，さらに解析を行うことができる．

4.2 仮想実体化のプロセス

4.1 節で部品化した要素機能部品群を図 3 のように機能合成し，仮想実体化フレームワークを実現する．以下に，DB レコードの仮想実体化プロセスを示す．

(1) DBProxyBox のスロット #query に入力された検索文はスロット #search へのアクセスをトリガとして DBMS に送信され，その評価結果であるレコード配列がスロット #resultList に格納される．

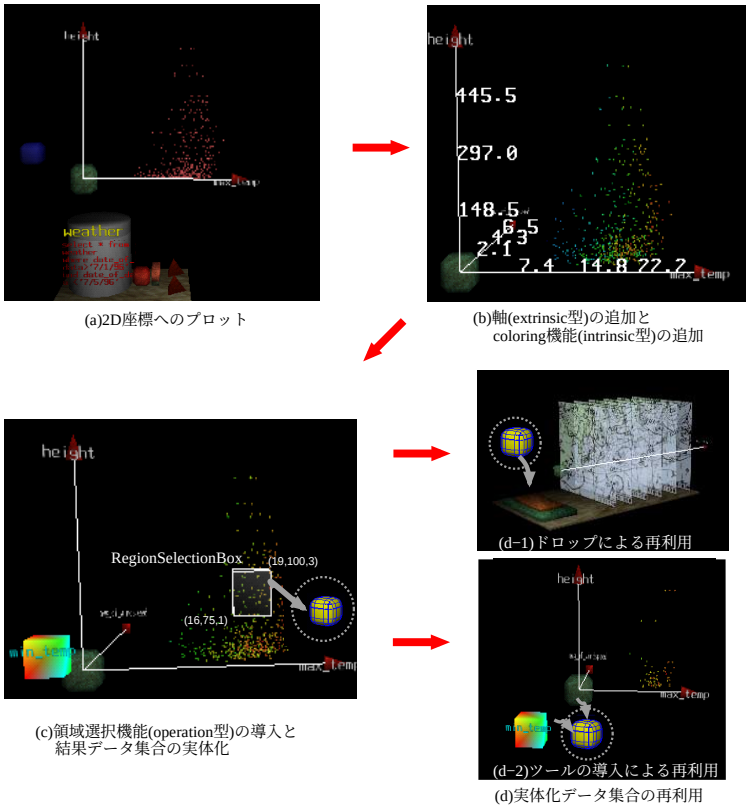
(2) DBProxyBox のスロット #resultList に結合された RecordManagerBox は各レコード情報を保持するレコードスロットを生成する．同時にスロット #registration に登録された表現モデルを複製し，レコードスロットに結合して各レコードを実体化する．

(3) 実体化されたレコード集合の情報は GeometryManagerBox に渡される．GeometryManagerBox はスロット #addFunction に登録された intrinsic 型，もしくは extrinsic 型の GM ファンクション・ボックスに基づき実体化集合の空間属性管理を行う．

(4) ユーザは，必要に応じて operation 型の GM ファンクション・ボックスを GeometryManagerBox に登録し，実体化集合に対する演算を導入できる．導入した演算の結果データ集合はユーザの要求に応じて実体化され，他のツールと組み合わせて再利用できる．

5. レコード表現モデル

4 章で述べたフレームワークは，DB スキーマやユーザの要求に応じてカスタマイズしたレコード表現モデルを基に仮想実体を生成する．本章ではまず，レコード表現モデルの部品構成法について説明する．次に，3D 部品アーキテクチャを基盤とすることによるレコード表現の多様性について議論した後，レコード表現の



(a)2D座標へのプロット
(b)軸(extrinsic型)の追加と coloring機能(intrinsic型)の追加
(c)領域選択機能(operation型)の導入と 結果データ集合の実体化
(d)実体化データ集合の再利用

図 7 GM ファンクション・ボックスのカスタマイズ例
Fig. 7 Example customizations of geometrical management functions and example uses of operation functions.

カスタマイズ手法について述べる．そして最後に，入れ子構造を含むレコードを実体化するための表現モデルの部品構成法について述べる．

5.1 表現モデルの部品構成法

表現モデルは，レコード情報を保持する基盤部品と属性表現部品群から構成される．この基盤部品と属性表現部品の機能連携により，各属性とその表現部品とのマッピングが定義される．

〈レコード情報保持部品〉 表現モデルは，レコード情報を保持する RecordBox を基にして定義される．RecordBox は，レコード属性配列を保持するスロット#record とその各属性情報を保持する属性スロットを持つ．各属性スロット値とスロット#record の対応する各要素値はつねに同値を保つ．RecordBox はスロット#addSlot やスロット#deleteSlot に属性名と型が指定されると，属性スロットの追加や削除を行う．

〈マッピング定義〉 ユーザは RecordBox の各属性スロットのデータ型に応じて，同じデータ型のスロットを持つ部品を属性表現ボックスとして結合し，表現モデルを定義する．この属性スロットと属性表現ボックス群のスロット結合がマッピング定義に相当する．

〈表現モデルの例〉 **図 8** に，属性 name，radius，rotation，orbit，revolution，surface を持つクラス planet のレコード表現モデルの例を示す．図 8 (a) は表現モデル部品構造とマッピング定義を表している．RecordBox は，クラス planet の持つ各属性をスロットとして保持する．RecordBox の子ボックスとして，惑星と同じ動作をするアニメーション合成部品を用いた．図 8 (b) に，合成ボックスの部品構成を示す．RecordBox の持つ属性スロット#revolution に周回運動速度を制御する回転部品，スロット#orbit に伸縮部品，スロット#rotation に自転速度を制御する回転部品，スロット#radius に赤道半径を制御するスケーリング部品，そしてスロット#surface に ImageBox をマッピングしている．

この表現モデルを RecordManagerBox に登録して，クラス planet に対する検索を行った結果を**図 9** に示す．各惑星レコードの属性値は RecordBox から属性表現ボックスに渡され，各値に応じて，公転速度，軌道半径，自転速度，赤道半径，表面テクスチャの異な

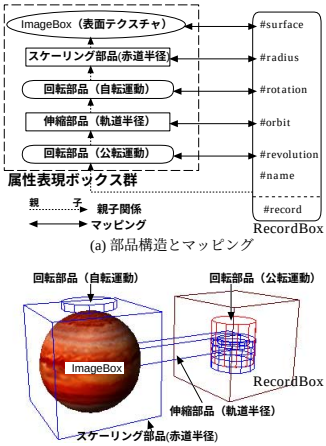


図 8 表現モデルの例
Fig. 8 An example record representation model.

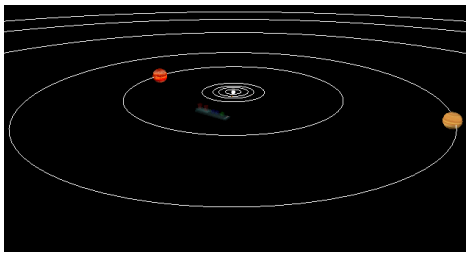


図 9 アニメーションによる仮想実体化
Fig. 9 The animated nine planets generated from database records.

る惑星アニメーションが生成される．

5.2 レコード表現の多様性

ユーザは RecordBox の持つ属性スロットに，属性と同じデータ型のスロットを持つ任意のボックスを結合することができる．TextBox，ImageBox 等の基本ボックスのほか，ユーザの定義したアニメーションや数値解析を行う合成ボックス等，様々なボックスを属性表現ボックスとして使用できる．

たとえば，数値属性をアニメーション部品のパラメータとして使用すれば，図 9 で示したように，各レコードを異なる速度を持つアニメーション物体として実体化できる．また，DB アクセス部品を属性表現ボックスとして使用すれば，同一ないしは別の DB から関連データを逐次検索するナビゲーションを実現できる．

従来の視覚化システムでは，システムが提供する数種の表現しか使用できない．これに対して Intelligent-Box では，マルチメディア・データ，アプリケーション・ツール等の組合せにより，多様なレコード表現を

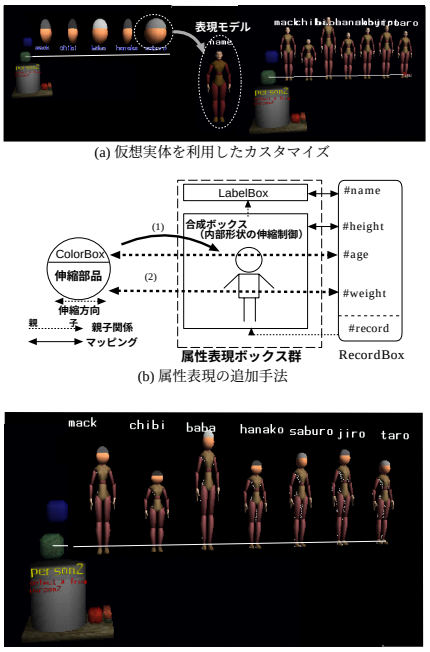


図 10 表現モデルのカスタマイズ例
Fig. 10 An example customization of a representation model.

定義できる．さらに，既存のボックスのみならず，将来開発されるあらゆるボックスも同じフレームワーク内で利用可能である．これにより，ユーザの要求に応じた多様なレコード表現を実現することができる．

5.3 表現モデルのカスタマイズ

表現モデルのカスタマイズは，特定 DB データの多面的観察や異なる DB データを視覚化する場合に行う．表現モデルは合成ボックスとして定義されているため，ユーザは，属性表現ボックスの追加，削除，交換等のカスタマイズができる．3 章で述べたように，カスタマイズは画面上で直接操作して行うことができ，その結果動作をモードを切り換えずに確認できる．

たとえば，**図 10** (a) 右は，個人情報 DB の height 属性と name 属性を用いたレコード表現，図 10 (a) 左は，name 属性と weight 属性，age 属性を用いたレコード表現を表している（図 5 参照）．いま，図 10 (a) 左の仮想実体のうち，weight 属性と age 属性にマッピングされた部品を分解し，図 10 (a) 右の表現モデルに追加する．図 10 (b) にこのカスタマイズ手法を示す．図 10 (b) 左の部品群は，図 10 (a) 左の表現モデルを分解した部分構造を表している．(1) これを図 10 (b) 右の表現モデルの人型形状の頭部品と交換し，(2) RecordBox の持つスロット #weight，スロット #age とスロット結合してカスタマイズする．この表現モデルを Record-

表 4 構造木ノード記号
Table 4 Node types used in tree structured schemata.

記号	データ型
○	基本データ型
⊕	レコード型
⊗	コレクション型

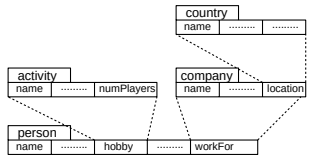


図 11 レコード型属性を含むスキーマの例
Fig. 11 An example schema with record type attributes.

ManagerBox に再登録することで図 10 (c) の結果が得られる．これらの作業はすべて画面上での直接操作により行える．例で示した属性表現部品の追加のほか，削除，交換等も同様にしてカスタマイズできる．

5.4 入れ子リレーションの表現モデル

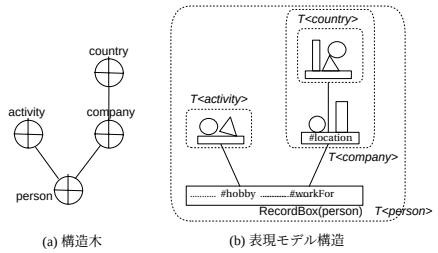
OODB や ORDB には，レコード型属性やコレクション型属性等，複雑な構造を持つ属性を含むレコードが格納されている．レコード型属性やコレクション型属性を再帰的に含む入れ子構造を入れ子リレーションと呼ぶ[2]．入れ子リレーションは**表 4** に示す 3 種類のノードからなる構造木で表すことができる[14]．

この節では，入れ子リレーションに対する表現モデルを，その構造木に基づいて構築する手法を提案する．

5.4.1 レコード型属性

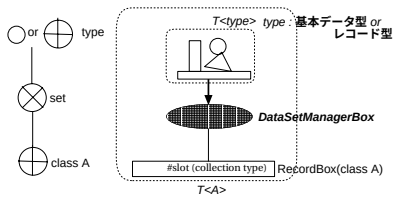
図 11 にレコード型属性を含むスキーマ例を示す．クラス preson の属性 hobby はレコード型である activity 型，属性 workFor は company 型のデータを持つ．また，クラス company の属性 location は country 型データを持つ．このスキーマの構造木を**図 12** (a) に示す．ただし，基本データ型の属性ノードは省略している．この構造木で表される入れ子リレーションの表現モデルの構造を図 12 (b) に示す．ここで，$T\langle type\rangle$ はレコード型，もしくは基本データ型 *type* の表現モデルを表す．本稿では以下この表記を用いる．

図 12 (a) の構造木において，先端ノードのクラスから順に，クラス country の表現モデル $T\langle country\rangle$ を定義し，$T\langle company\rangle$ の RecordBox が持つ属性スロット #location にこれをつなぐ．この $T\langle company\rangle$ を $T\langle person\rangle$ の RecordBox が持つ属性スロット #workFor に結合し，$T\langle activity\rangle$ を同じ RecordBox が持つ属性スロット #hobby につないで $T\langle person\rangle$ を定義できる（図 12 (b)）．



(a) 構造木 (b) 表現モデル構造

図 12 レコード型属性を含むスキーマの構造木とその表現モデル構造
Fig. 12 A tree structured schema and its representation model.



(a) 構造木 (b) 表現モデル構造

図 13 コレクション型を含むスキーマの構造木とその表現モデル構造
Fig. 13 A tree structured schema using collection type attributes and its representation model.

5.4.2 コレクション型属性

コレクション型属性の要素型には，基本データ型，もしくはレコード型が使用される．コレクション型属性を持つスキーマの構造木を**図 13** (a) に示す．図 13 (b) は，コレクション型属性を持つクラス A の表現モデル構造を表している．構造木のコレクション型ノードに対応する箇所には DataSetManagerBox を結合する．DataSetManagerBox は，4.1.2 項で述べた RecordManagerBox の機能を拡張し，レコード型以外のデータ配列を実体化可能にしたボックスである．

図 13 (a) の構造木において，コレクション型属性の要素型ノードに対応する表現モデル $T\langle type\rangle$ を定義し，DataSetManagerBox に登録する．この DataSetManagerBox を $T\langle A\rangle$ の RecordBox が持つコレクション型属性スロットに結合して $T\langle A\rangle$ が定義できる．

5.4.3 高次な入れ子リレーションの表現モデル

ここでは，**図 14** (a) の構造木で表される高次な入れ子リレーションの表現モデル $T\langle A\rangle$ の定義手法について述べる．構造木の先端ノードから順に，そのノードの型に応じて表現モデルを定義し，その表現モデルを 5.4.1 項で述べたようにして RecordBox のレコード型属性スロット，または 5.4.2 項で述べたようにして DataSetManagerBox に登録して，この DataSetManagerBox を RecordBox のコレクション型属性ス

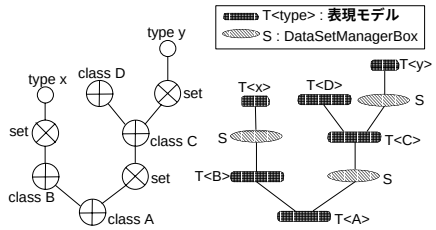


図 14 高次な入れ子リレーションの構造木と表現モデル構造
Fig. 14 A tree structured schema with hierarchically nested relations and its representation model.

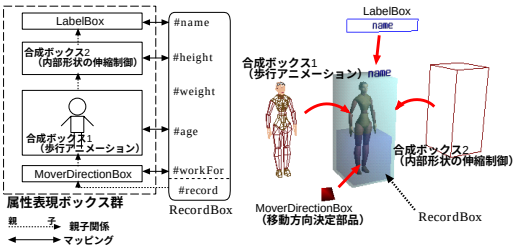


図 15 クラス person の表現モデル例
Fig. 15 A class 'person' and its representation model for virtual reification.

ロットに結合する．これを構造木のルート・ノードまで繰り返すことにより $T\langle A\rangle$ を定義できる．

この $T\langle A\rangle$ の部品構造を図 14 (b) に示す．図で示したように，高次な入れ子リレーションを仮想実体化するための表現モデルも，その構造木と同じ構造に部品を組み立てることで定義することができる．

6. 仮想実体化例

この章では，提案する仮想実体化フレームワークによる DB レコードの仮想実体化例を紹介する．

6.1 個人情報データの仮想実体化

最初に，属性として name，age，height，weight，workFor を持つクラス person のレコードの仮想実体化例を示す．これに対する表現モデル $T\langle person\rangle$ の部品構造とマッピング例を**図 15** (a) に示す．合成ボックス 1 の人形は，各関節部に回転部品を埋め込んだ歩行アニメーション合成ボックスである．スロット #age を人形の歩行速度制御部品にマッピングしている．また，合成ボックス 2 は，自分の形状内部に含まれるボックスの形状を相対的に伸縮させる合成ボックスである．これは RecordBox のスロット #height にマッピングされている．RecordBox のスロット #name は，文字列表示を行う LabelBox にマッピングしている．図 15 (b) に実際の部品構成を示す．この表現モデルを RecordManagerBox に登録し，weight，age の 2 属性を設定した座標軸部品を座標原点部品と組み合わせて，クラス person に対して検索を行うと**図 16** (a) の結果が得られる．各レコードは RecordBox に渡され，RecordBox は各属性値をマッピングされた属性表現ボックスに送信する．height 属性値は，合成ボックス 2 に渡され人形の高さを変更する．また，age 属性値は，合成ボックス 2 の歩行速度制御部品に渡され歩行アニメーションの速度を変更する．各レコードは，height 属性値に応じて高さ，age 属性値に応じて歩行速度が異なる人型形状を持つインタラクティブなアニメーション部品として実体化され，weight 属性と age 属性の値に応じて 2 次元の直交座標上に配置される．

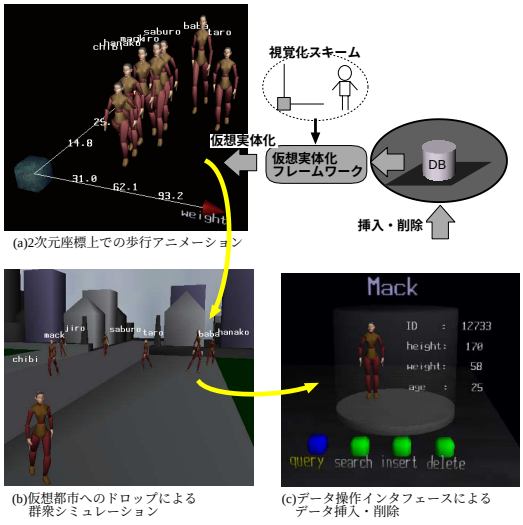
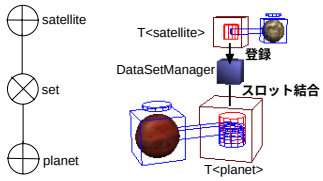


図 16 個人情報データの仮想実体化例
Fig. 16 Virtual reification of personnel database records.

RecordBox の workFor 属性にマッピングされた MoverDirectionBox のスロットには，オフィス名データが格納される．MoverDirectionBox は，同データを持つ MoverDestinationBox の方向へ移動する部品である．いま，各仮想オフィスにそのオフィス名を入力した MoverDestinationBox を埋め込んだ仮想都市があるとする．図 16 (b) のように，人形をこの仮想都市に直接操作によりドロップすると，属性 workFor の値に応じて自分のオフィスへ向かって歩き出す簡単な群衆シミュレーションを実現することができる．

図 16 (c) は，仮想都市内を歩く人形をコピーして 4.1.1 項で述べた 3D フォーム・インタフェースの 3D フォームとして使用した例である．ユーザは，ボタン部品を押してこのレコードを DB から削除したり，データを直接入力して新規データを挿入することができる．

このように，DB レコードをアニメーション部品と



(a) 構造木 (b) 表現モデルの構成

図 17 クラス planet の構造木と表現モデル例
Fig. 17 A tree structured schema for the class planet and its representation model.

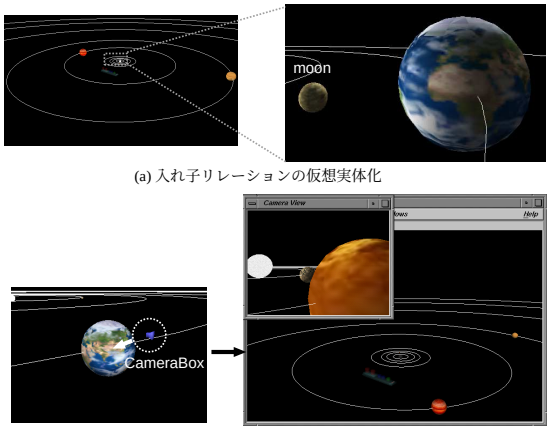
して仮想空間内に実体化でき，かつ，仮想実体そのものを直接，異なる仮想空間に持ち込むことが可能である．他のツールとの連携により，仮想実体化された DB データを再利用することができ，仮想空間での編集を DB に再蓄積することも可能である．

6.2 惑星データの仮想実体化

簡単な入れ子リレーションを持つ惑星 DB の仮想実体化例を示す．クラス planet はクラス satellite のコレクション型属性を持つ．**図 17** にその構造木と表現モデル構成を示す．クラス planet，クラス satellite は赤道半径，自転周期，公転軌道半径，公転周期，表面テクスチャを属性に持つ．表現モデル $T\langle planet\rangle$，$T\langle satellite\rangle$ は，各々図 8 で示す部品構造とマッピングが定義されている．$T\langle planet\rangle$ は $T\langle satellite\rangle$ を登録した DataSetManagerBox を RecordBox のコレクション型属性スロット #satellites に結合して定義する．

この表現モデルを RecordManagerBox に登録してクラス planet に対して検索すると，各惑星のデータが 5.1 節と同様にして仮想実体化される．実体化された各惑星の RecordBox のスロット #satellites には，その惑星を周回する衛星群のデータが格納される．この衛星データ集合は，RecordBox のスロット #satellites に結合された DataSetManagerBox に送信される．DataSetManagerBox は各衛星のデータを保持するスロットを生成し，登録された $T\langle satellite\rangle$ を複製して衛星データを仮想実体化する．このプロセスがすべての惑星について行われて太陽系アニメーションが生成される（**図 18** (a)）．この仮想の太陽系空間において，ユーザは地球の任意の地点にカメラ部品を導入することができる（図 18 (b)）．カメラ部品は，ユーザの指定するレンズ面からのビューを別ウィンドウに表示する機能を持つ．図 18 (c) は，地球のある地点に導入したカメラ部品のウィンドウを開いた様子である．このウィンドウには，地球のカメラを取り付けた地点から見た他の惑星の運動の様子が表示される．

このように，仮想空間においてユーザは目的に応じて任意のツールを導入し，仮想実体化された DB レコードと自由に連携することが可能である．



(a) 入れ子リレーションの仮想実体化

(b) カメラ部品の導入 (c) カメラ部品から見たビューの表示

図 18 惑星データの仮想実体化例
Fig. 18 Virtual reification of the planet database.

7. おわりに

本稿では，ユーザによるカスタマイズが可能な任意の視覚化スキームを用いて，様々な DB レコードをインタラクティブな 3D オブジェクトとして仮想空間内に実体化するためのフレームワークを提案した．

従来の視覚化システムがデータド メインに特化した数種の視覚化スキームのみを提案しているのに対し，本研究ではカスタマイズ可能な視覚化スキームを提供することにより，様々な DB データの視覚化を可能にした．IntelligentBox の 3D 部品アーキテクチャを基盤とすることで，直接操作可能な部品を用いた視覚化スキームのカスタマイズ，アニメーションをはじめとする多様なレコード表現が可能であることを示した．

本研究では，OODB や ORDB に蓄積された入れ子構造の DB レコードに対しても，部品合成による容易なレコード表現の定義を可能とするため，その部品構成法を提案した．木構造スキーマの表記法，および，それに対応する表現モデルの表記法を提示することにより，提案する部品構成法の容易性を示した．

ユーザによる DB レコードの柔軟な解析・利用操作を支援するため，本研究では DB レコードを仮想的な「もの」として扱うための枠組みを示した．仮想空間や DB レコードの仮想実体に対する任意のアプリケーション・ツールの導入，および，それらのツールと DB レコードとの自由な連携を可能にすることにより，ユーザがその仮想空間で柔軟な解析・利用操作を行うことができることをいくつかの実体化例を通して

示せたと考える．

参 考 文 献

1) Ahlberg, C. and Wistrand, E.: IVEE: An Information Visualization and Exploration Environment, *Proc. IEEE Visualization'95*, IEEE (1995).
2) Atzeni, P. and Antonellis, V.D.: *Relational Database Theory*, Benjamin/Cummings Publ. Comp., Redwood City, California (1993).
3) Benedikt, M.: *Cyberspace: Some Proposals in Cyberspace: First Steps*, MIT Press (1991).
4) Benford, S., Snowdon, D., Greenhalgh, C., Ingram, R., Knox, I. and Brown, C.: VR-VIBE: A Virtual Environment for Co-operative Information Retrieval, *Computer Graphics Forum*, Vol.14, No.3, pp.349–360 (1995).
5) Bray, T.: Measuring the Web, *Computer Networks and ISDN Systems*, Vol.28, No.7-11, pp.993–1005 (1996).
6) Card, S.K., Robertson, G.G. and York, W.: The WebBook and the Web Forager: An Information Workspace for the World-Wide Web, *Proc. ACM CHI 96 Conference on Human Factors in Computing Systems*, pp.111–117 (1996).
7) Chalmers, M. and Chitson, P.: Bead: Explorations in Information Visualization, *Proc. ACM SIGIR'92, published as a special issue of SIGIR forum*, pp.330–337 (1992).
8) Eick, S.G., Steffen, J.L. and Sumner Jr., E.E.: Seesoft – A Tool For Visualizing Line Oriented Software Statistics, *IEEE Trans. Softw. Eng.*, pp.957–68 (1992).
9) Freeman, E. and Fertig, S.: Lifestream: Organizing your electronic life, *Proc. AAAI Fall Symposium on AI Applications in Knowledge Navigation* (1995).
10) Hemmje, M., Kunkel, C. and Willet, A.: LyberWorld – A Visualization User Interface Supporting Fulltext Retrieval, *Proc. ACM SIGIR'94* (1994).
11) Krohn, U.: *VINETA: Navigation Through Virtual Infomation Spaces*, AVI: Advanced Visual Interfaces, Italy (1996).
12) Mackinlay, J.D., Robertson, G.G. and Card, S.K.: The Perspective Wall: Detail and Context Smoothly Integrated, *Proc. ACM CHI'91 Conference on Human Factors in Computing Systems and Graphics Interface, ACM SIGCHI*, ACM-Press (1991).
13) Mariani, J. and Benford, S.: Populated Information Terrains: Virtual Environments for Sharing Data, Technical Report CSCW/4/94, Lancaster University, Computing Department (1994).
14) Massari, A., Saladini, L., Hemmja, M. and Sisinni, F.: Virgilio: A Non-Immersive VR System To Browse Multimedia Databases, *Proc. Intl. Conf. on Multimedia Computing and Systems*, IEEE (1997).
15) Okada, Y. and Tanaka, Y.: IntelligentBox: A Constructive Visual Software Developement System for Interactive 3D Graphic Apprications, *Proc. Computer Animation '95*, pp.114–125, IEEE (1994).
16) Wright, W.: Information animation applications in the capital markets, *Proc. InfoVis'95.*
17) 岡田義広，田中 譲：対話型 3D ソフトウェア構築システム—IntelligentBox，コンピュータソフトウェア，Vol.12, No.4, pp.374–384 (1995).

(平成 12 年 6 月 20 日受付)
(平成 12 年 9 月 27 日採録)

(担当編集委員 有澤 博)

大東 誠

1973 年生．1997 年北海道大学工学部電気工学科卒業．1999 年同大学院電子情報工学専攻博士前期課程修了．現在，同大学院電子情報工学専攻博士後期課程に在学中．データベース視覚化，アプリケーション・フレームワーク等の研究に従事．

田中 譲（正会員）

1972 年京都大学工学部電気工学科卒業．1974 年同大学院電子工学専攻修士課程修了．工学博士．1974 年北海道大学工学部助手．1977 年同講師．1985 年同助教授を経て，1990 年同教授，現在に至る．1996 年北海道大学知識メディアラボラトリー長．この間，1985 年 10 月より 1 年間，IBM 社 T.J. ワトソン研究所客員研究員．ソフトウェア科学会，人工知能学会，米国 IEEE 各会員．データベース理論，データベース・マシン，並列処理アーキテクチャ，メディア・アーキテクチャ等の研究に従事．「コンピュータ・アーキテクチャ」（雨宮氏との共著，オーム社）等の著書あり．1994 年に IntelligentPad の開発に関して日経 BP 技術賞大賞受賞．