半構造オブジェクト：ユビクタスデータモデル

倉光君郎[†,☆] 坂村健[†,☆]

インターネット時代のデータは，どこにでも存在する．電子商取引やデジタルライブラリなど現在のデータアプリケーションは，異なる組織の間でのデータの交換や統合を必要としている．そのデータプラットホームも，従来のデータベースから，XMLやスマートカードまで多様化している．我々の目標は，これらの上でデータプラットホームに依存しない統一的なデータ表現と問合せを定式化し，データアプリケーションの統合を促進することである．半構造オブジェクトは，ユビクタスデータモデルとして設計され，不規則で変化しやすい情報構造に対する自己記述的な表現力，データ流通や結合の単位としてのオブジェクト構造，そして概念と表現の分離などの特徴を持つ．本論文では，半構造オブジェクトの実装として，2つの主要なデータプラットホーム，XMLとリレーショナルデータベース上の実装とその相互運用性について議論する．

Semistructured Object: A Ubiquitous Data Model

Kimio Kuramitsu[†,☆] and Ken Sakamura[†,☆]

In the Internet age we have data anywhere. Data applications, like electronic commerce and digital library, need the exchange and integration of data between different organizations. Data platforms for that have became ubiquitous; relational databases, XML, and smartcard. The objective of this study is to model common data and queries for such platforms in a platform-independent fashion, and then to facilitate the integration of data applications. As a ubiquitous data model, we here formulize semistructured object, which has modeling facilities with self-describing irregular information structures of the real-world entities, moving and combing objects that different authors create, differentiating between concept and structure for semantic interoperability. This paper will discuss data interoperability between two major platforms, relational database and XML, using our implementation of ubiquitous data.

1. はじめに

インターネットは，世界に変化を与え続けている．我々は，Webや電子メールを通して，いつでもどこでも様々な情報を交換することが可能である．今日では，ほとんどすべての社会活動（ビジネス，教育，政府サービスなど）が，インターネットとその情報流通による大きな変革に直面している．しかしWebや電子メール上の情報は，十分に構造化されておらず，提供者ごとに表現は不統一である．そのため，収集した情報を手作業で再編集，そして管理しなければならないケースも少なくない．インターネット上でより効率良く社会サービスを連携するためには，構造的な情報，つまりデータの流通が強く求められる[2]．

† 東京大学
University of Tokyo
☆ 現在，東京大学大学院情報学環
Presently with Interfaculty Initiative in Information Studies, University of Tokyo

インターネット上のデータに対する要望は，1990年代後半から始まるXML（eXtensible Markup Language）への高い関心にも重なる[4]．XMLは，W3Cが標準化を進めるWeb上の標準フォーマット[6]であり，記述者に構造とセマンティクスを自由に拡張することを認めている．今日，電子商取引やデジタルライブラリにおいて，情報を交換するデータハブとして期待ができる．しかし，XMLが標準化するのはシンタックスのみである．伝統的なデータベースモデルや異なる組織間でのデータセマンティクスの相互運用性の課題を残している．

我々は，インターネット時代の新しいデータ技術を考える鍵はデータの遍在性（ubiquity）であると考える．つまり，データアプリケーションの立場から見れば，データはデータベースやXML，スマートカードなど様々なデータプラットホーム上に遍在する．そこで遍在するデータを論じるため，まずデータプラットホームに依存しない共通データモデルが必要となる．本論文の目的は，ユビクタスデータモデルとして，そ

のデータ表現と問合せの定式化を試みることである．

半構造オブジェクトは，本論文において定義するユビクタスデータモデルである．半構造オブジェクトは，その名前の示すとおり，半構造データ（semistructured data）とオブジェクトモデル（object model）の 2 種類のデータモデルをベースに設計されている．データは，オブジェクト構造で記述され，第 3 者の記述したオブジェクトをシンタックス上もセマンティクス上も結合することができる．半構造オブジェクトの構造自体は，自己記述的であり，不規則な情報構造を柔軟に表現することができる．オブジェクトへのアクセスは，統一されたビューを通して行う．データアプリケーションは，ビュー上でクラス制約を検証することも可能になる．

我々は，先行研究として，インターネット商取引向けのカタログ記述言語 PCO を設計した．PCO の成果は，論文 13) ですでに報告されている．本論文の貢献は，先行研究の成果や知見を精錬し，ユビクタスデータモデルとして定式化を行った点である．半構造オブジェクトは，特に概念と表現を明確に分離できるように改良が行われた．本論文では，それに加えて今日の代表的なデータプラットホームである，XML とリレーショナルデータベース上のデータに対して，相互運用の議論を行う．

本論文の構成は次のとおりである．2 章では，関連するデータモデルをまとめ，我々のユビクタスデータモデルの設計目標をまとめる．3 章では，半構造オブジェクトと問合せを定義する．4 章では，XML 上の半構造オブジェクトのコーディングとして X#を述べる．5 章では，リレーショナルデータベース上の半構造オブジェクトのマッピングを論じる．6 章では，論文を総括する．

2. ユビクタスデータ

インターネット時代に適したデータのあり方とは何か？ この問いが，ユビクタスなデータをモデル化する動機である．本章では，まずデータモデルの歴史と論点を振り返り，我々の目指している新しいデータモデルの設計目標を設定する．

2.1 データモデルの歴史

データモデルは，データをエンコーディングから独立させ，その構造と問合せを定式化して議論する土俵である．その起源は，1970 年に Codd によって提案されたリレーショナルモデル[10]といえる．リレーショナルモデルは，今日のデータベース技術に最も大きな影響を与えているが，同時に，多くの研究者から情熱的に改良が試みられてきた．

データモデルの論点の 1 つは，「いかに実世界の情報構造を正しく表現するか？」といえる．この点において，リレーショナルモデルは，Codd 自身も指摘[11]したとおり，セマンティクスが貧弱である．多くの研究者は，リレーションだけでなくよりリッチな構造を用いて，実世界の情報構造を記述しようと試みてきた．その試みは，集合や順序，階層構造など，様々なセマンティクスを持った構造[12]の拡張につながる．この方向への拡張は，1980 年代後半にオブジェクト指向モデルとして統一されることになった[3),17)]．実世界の情報は，すべてオブジェクトで表現される．オブジェクトは，クラス–サブクラス（IS-A）関係で分類され，各オブジェクト間の関係は，コンポジション（HAS-A）で表現される．

1990 年代は，半構造データ（semistructured data）と呼ばれる新しいデータモデルの流れ[1]が登場した．特徴は，不規則で変化に富んだ実世界の情報構造を，無理にクラスで分類するのではなく，その構造自体をラベル付きの木構造で自己記述する点である．スタンフォード大学の Object Exchange Model（OEM）モデルは，半構造データの代表的な定式化モデルである．TSIMMIS [7]や YAT [9]は，半構造データをベースにした代表的なデータ管理システムである．現在では，自己記述的☆なデータ構造の表現力と柔軟性が広く認められ，HTML や XML の普及が示すとおり，ネットワーク上でデータを交換するのに適したデータモデルとして利用される．

2.2 ユビクタスデータの設計目標

過去のデータモデルは，データはつねにデータベース上に存在することを前提として議論されてきた．一方，ユビクタスデータは，様々なデータベース上に遍在することを想定する．マルチデータベース（multi-database）やヘテロジニアスデータベース（heterogeneous database）との大きな違いは，データがインターネットを通して，データベースや分散したアプリケーションの間であらかじめ予測不可能な形で移動する点である．我々は，ユビクタスデータに求められる要求を次のように設定する．

1. **データプラットホーム独立.** データプラットホームは，データシンタックスを定義し，そのアクセス手段を提供するシステムである．データベー

☆ オブジェクト指向データモデルの分野においても，柔軟なオブジェクト構造の研究は行われ，Object Specialization などの手法が提案された．半構造データは，外部からのスキーマ制約を必要としない点が大きな違いである．

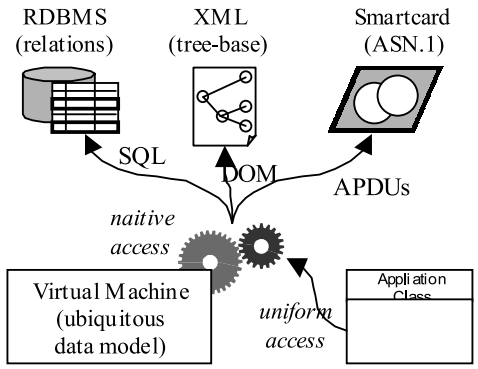

図 1 ユビクタスデータモデルのアーキテクチャ
Fig. 1 Architecture of ubiquitous data model.

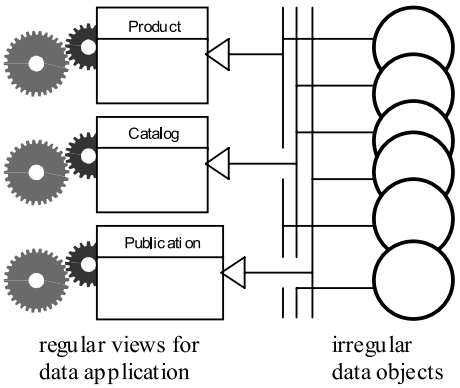

図 2 統合ビューモデル
Fig. 2 Integrated view.

スや XML，ASN.1 などがこれに相当する．ユビクタスデータは，これらのデータプラットホームの上にどこでも存在でき，かつデータセマンティクスから独立してデータ相互運用できる必要がある．

2. **記述者とアプリケーションの分散独立性．** ユビクタスデータの記述者と利用者は，インターネットを通して分散的に独立しており，その関係も動的である．データは，記述者の記述要求に従って表現されるべきであり，逆にデータアプリケーションの要求に従って，データ検証が行われるべきである．

我々は，データプラットホーム独立性を満たすため，仮想プロセッサ（virtual processor）モデルを導入する．**図 1** は，その概念的アーキテクチャを示している．ユビキタスデータは，各データプラットホームごと異なるシンタックスで表現されるが，その上で統一されたデータモデルを持つ．データモデルが統一されているため，異なるシンタックスはそのセマンティクスに依存することなく，機械的に相互変換することができる．さらに，このアーキテクチャは，Java プログラミング実行環境におけるバーチャルマシンに類似している．ユビキタスデータへの問合せは，各データプラットホームのネイティブ問合せに翻訳されて実行される．データアプリケーションは，そのデータプラットホームから独立して，ユビクタスデータへの操作を行うことができる．

我々は，データの多様性と分散アプリケーションの多様性のギャップを埋めるため，統合ビュークエリープロセッサモデルを導入する．**図 2** は，異なる記述者が表現した構造の異なるデータをアプリケーションから統合ビューを用いてアクセスする様子を表している．ユビクタスデータは，ビューを通してアクセスされることで，自動的にアプリケーション側のデータ仕様を検証することができる．

3. 半構造オブジェクトと問合せ

半構造オブジェクトは，実世界に存在する「もの」の不規則で変わりやすい情報構造を表現し，そのデータをネットワーク上で共有するためのデータモデルである．本章では，まずはじめに議論に必要な基本的な用語のインフォーマルな定義を行う．続いて，半構造オブジェクトのデータ構造と問合せの定式化を行う．

3.1 準　　備

データ（data）は，複数のデータ値（values）からなる構造体である．次は，3 つのデータ値のタプルからなるデータの例である．

[“CD Audio”, 1000, 9.85] ··· (3.1)

データ型（datatype）は，データとデータの構造を識別する情報である．たとえば，リレーショナルモデルでは，型スキーマとして，あらかじめ型定義を行うことで，データとデータ値の構成を明らかにする．

[String, Integer, Double] ··· (3.2)

名前（name）は，データ独立（data-independent）な方法によるデータ値へのアクセスパスである．リレーショナルモデルでは，名前も型スキーマ同様，タプルであらかじめ定義される．

[Product, PriceYEN, PriceUSD] ··· (3.3)

概念（concept）は，データ値の実世界コンテクストにおける意味を表す．通常，データ値とデータ値の関係（relation）によって明確にされる．知識表現の分野では，述語（predicate）やステートメント（statement）を用いて表現することもある．次は，ステートメントによるデータ値の意味の明確化である．

Product is sold for PriceYEN or PriceUSD ··· (3.4)

重要なのは，データ名自体が必ずしもデータ値の

意味を正確に表現するものではない点である．我々はデータ値に対してその意味に対応する自然言語上の単語を名付けることが多いが，これはあくまでもデータ上の値を区別する識別子である．

最後に，構造データと半構造データの違いをはっきりさせる．本論文では，半構造データは「名前やデータ型の情報☆が自己記述的（self-describing）に含まれるデータ」のことを呼ぶ．次は，構造データ (3.1) とそのスキーマ (3.2)，(3.3) を含んだ半構造データである．

{[Product, String, “CD Audio”],
[PriceYEN, Integer, 1000],
[PriceUSD, Double, 9.85]} ··· (3.5)

半構造データでは，必ずしも名前やデータ型の両方を持つ必要はない．たとえば，HTMLでは，すべての値は文字列リテラルと見なし，データ型は存在しない．XMLでは，データ型を後から導入したため，XMLスキーマ言語によって，各データ値にデータ型を追加している．

3.2 概念とデータ型

半構造オブジェクトでは，プロパティ（property）と名付けた自己記述的な単位でデータ値を表現する．

定義 3.1 プロパティは，$[l, t, v]$ の三つ組（triple）で表記される．このとき，$v \in Vt$ はデータ値である．$l \in \mathbf{L}$ は，データ値 v の意味を表す概念ラベル（concept label）であり，$t \in \mathbf{T}$ はデータ値 v のクラスを表すデータ型である． □

我々のモデルの特徴は，データ値に対して，概念ラベルを導入した点である．概念ラベルは，データ値を識別する名前ではなく，その概念的な意味を表す．つまり，概念が同じデータ値は，そのデータ値の表現（データ型）が違っても同じラベルがつけられる．次は，(3.5) の Price を，同じ概念のデータと異なった表現で記述したものである．

[Price, Integer/Currency-Yen, 1000]
[Price, Double/Currency-USD, 9.85] ··· (3.6)

概念ラベルの導入は，2つの新しいユニークな結果をもたらす．1つは，ラベルとデータ型の関係である．名前ラベルの場合，ラベルは名前として，データ値に連想されているため，ラベルとデータ型の関係は1対1であり，かつ固定的である．これに対し，概念ラベルはデータ値やその形式を表すデータ型から独立させ，その関係も1対1に固定しない．たとえば，記述者によっては，Price という概念は，次のような形式で表現することもできる．

[Price, String, “オープンプライス”] ··· (3.7)

もう1つの特徴は，データ値の表現方法の多様性をサポートするための，実世界型（real-world type）の導入である．データ型は，一般的には計算機セマンティクス，つまり記憶装置上のエンコーディングを基礎として導入されてきた．しかし，半構造オブジェクトの表現対象は，あくまでも実世界の実体や関係である．そこで，実世界セマンティクスを含んだ表現構造，つまり単位やスケール，パターンやボキャブラリセットなどに着目する．半構造オブジェクトの上では，データ型は計算機型と実世界型のペアで表現される．

Datatype	**Value (Encoded)**
String/Phone	03-5841-2484
Long/Currency-usd	18.00
Long/Date	2001-04-21
Image/Jpeg	MIME (Base64)
Class/ClassName	（複合オブジェクト）

半構造オブジェクトは，データモデルから概念モデルを明確に独立させるため，概念ラベルの意味とデータ型のセマンティクスを区別する．

定義 3.2 概念と型の独立性．データ値は，概念ラベルに依存することなく，データ型の変換関数のみで変換することができる．つまり，今，データ型変換関数 $\mathrm{cvt}(t_1, t_2, v_1)\ v_2$ が定義されていれば，いかなる概念ラベル l に対して，$[l, t_1, v_1]$ は，$[l, t_2, v_2]$ に変換することができる． □

3.3 オブジェクトと識別性

我々は，半構造オブジェクトにおける自己記述的なオブジェクト構造をここで定義できる．

定義 3.3 半構造オブジェクト（semistructured object）は，プロパティの集合である．半構造オブジェクト（属性数 n 個）は，$\{[l_1, t_1, v_1], [l_2, t_2, v_2], \ldots, [l_n, t_n, v_n]\}$ である．ここで，n は自己記述的に決定できる．さらにオブジェクト上のプロパティの順序は，意味がない． □

オブジェクト上では，複数のプロパティに対して同一の概念ラベルを割り付けることが認められる．そのため，概念ラベルは，プロパティを識別する名前にはならない．ここで，新たにプロパティの名前を次のように定義する．

定義 3.4 プロパティ名（property name）は，ラベルと型の組で構成される．我々は，名前を表記するため，$l(t)$ のように，ラベルの後にカッコ付きで型を追加す

☆ 半構造データは，その代表例として，HTMLがさかんに参照されたため，プレーンテキストのような構造なしデータが混在したデータとの解釈もある．本論文では，スキーマ参照なしにデータ構造が解析できるデータと位置付ける．

る記法を用いる．オブジェクト上では，プロパティ名は区別可能（distinct）であると想定する． □

ユビクタスデータは，異なった組織の間でデータが移動し，異なった著者によるデータが構造的にも意味的にも連結する必要がある．オブジェクトは，その移動と連結の単位である．

定義 3.5 オブジェクトは，データ値の一種である．オブジェクトどうしの連結は，コンポジション（composition）として表現され，オブジェクト間の関係は概念ラベルとして表現される． □

次は，プロパティ値にオブジェクトが含まれた複合オブジェクトの例である．オブジェクトのデータ型は，クラス（後述）で指定することができる．

{ [name, String, “Kimio Kuramitsu”],
[age, Integer/age, 28],
[birthday, Long/date, 1972-12-24]
[birthday, Class/date,
{[**year, Integer/year, 1972**],
[**month, Integer/month, 12**]
[**day, Integer/day, 24**]}] } ··· (3.8)

3.4 ビュー仕様とクラス

半構造オブジェクトは，記述者に対して，スキーマ的な外部制約を受けることなく，概念とデータ型を組み合わせる柔軟な表現を認めている．これに対し，データアプリケーションからのアクセスは，あらかじめ定義された固定的なデータ構造が望ましい．データアプリケーションからのデータ構造を固定化するものがビュー仕様である．

定義 3.6 ビュー（view）仕様は，アプリケーションから固定的にアクセスするための型付きプロパティの仕様である．プロパティ名の集合で表現される．$N_v = \{l_1(t_1), l_2(t_2), \ldots, l_n(t_n)\}$ また，オブジェクト全体のビューをオブジェクトビュー（N_o）と呼ぶ．あるビュー（N_v）がオブジェクト上ビューであるというのは，$N_v \subseteq N_o$ を満たすときである． □

アプリケーションは，ビュー仕様に対して名前を付けることで，あらかじめクラスとして定義しておくことができる．

定義 3.7 クラス（class）は，名前付きのビューである．クラス定義関数を f_t，クラス名を c とすれば，クラスは $f_t : c \rightarrow N_c = \{l_1(t_1), l_2(t_2), \ldots, l_n(t_n)\}$ で表記できる． □

注意：半構造オブジェクトモデルは，オブジェクト構造とビュー仕様が必ずしもスタティックにマッチングしなくてもよい．本論文では紙面の都合上，その機構の詳細を報告できないが，半構造オブジェクトは，定義 3.2 の型変換に概念モデルを組み合わることで，翻訳ベースの動的な型検証システムを拡張できるように設計されている．

3.5 アクセスパスと問合せ

半構造オブジェクトに対する問合せは，次の 2 種類に大きく分類できる．

1. **オブジェクト選択．** オブジェクト集合（object group）からある条件を満たすオブジェクト集合を取り出す．
2. **プロパティ操作．** ある特定オブジェクト（または，あるオブジェクトグループ）上のプロパティを操作すること．

本論文では，最も基本となる，オブジェクト操作とプロパティ操作の基本となるアクセスパス（access path）について述べる．

3.5.1 オブジェクト識別子

ユビクタスデータは，組織を越えて移動するデータである．その移動したデータは，組織の壁を越えて識別できなければならない．ただし，半構造オブジェクトモデルでは，必ずしもすべてのオブジェクトがオブジェクト識別子を持つことを義務付けられていない．複合オブジェクトは，名前なしオブジェクトも認められる．半構造オブジェクトは，オブジェクト識別子がある場合のみ，独立移動可能な（fragment）データと見なされる．

定義 3.8 オブジェクト識別子（object identifier）は，オブジェクトをインターネット上で識別する識別子である．著者識別子とオブジェクト名，ハッシュ識別子からなる． □

著者識別子は，電子メールアドレスなど，インターネット上のユニークな名前を用いる．オブジェクト名は，各オブジェクトの作者が自由に名付けることができる．これに対し，ハッシュ識別子は，暗号関数を用いてインスタンスごとに計算され，オブジェクトを一意に識別する識別子である．ここで，著者識別子とオブジェクト名を，a，o とし，ハッシュ関数を hash() とすれば，ハッシュ識別子 h の計算は次のとおりである．

$$\mathrm{h} = \mathrm{hash}([\mathrm{a}, \mathrm{o}, l_1, t_1, v_1, l_2, t_2, v_2, \ldots, l_n, t_n, v_n])$$

半構造オブジェクトの参照では，著者識別子とオブジェクト名で指定した場合は，同一名中で最新のオブジェクトが動的に参照される．これに対し，ハッシュ識別子付きで参照する場合は，スタティックに参照オブジェクトが決定される．ハッシュ識別子は，さらに，電子署名を生成する役割も果たす．

3.5.2 プロパティパス

半構造オブジェクト上のプロパティは，複合オブジェクトが含まれる場合，ツリー構造に類似したネスト構造になる．ネスト構造上のプロパティ値を直接アクセスするため，我々はプロパティ名を拡張して，プロパティパスを定義する．

定義 3.9 プロパティパス（property path）は，プロパティ名のドット記法の表現で表される．例. l1(t1).l2(t2) □

プロパティパスを使うことで，オブジェクト上のプロパティ値を指定して選択することが可能になる．プロパティパスは，データ型変換に対する問合せにも応用することができる．次の例は，プロパティパスによるデータ変換が含まれた select 文と，オブジェクト (3.8) に対する問合せ結果である（問合せ言語は，一般の SQL のセマンティクを継承しているものとする）．

```
select birthday(Class/Date).year(String/
Japanese-year)
    where birthday(Long/date) < 1970-01-01
```

→ “昭和 47 年”

上記の問合せは，プロパティパスと型変換の関係を示している．半構造オブジェクトへの問合せでは，データへの静的な問合せに，動的な型変換システムを組み合わせることが可能となる．

4. X#

X#は，XML 上の半構造オブジェクトの表現である．X#の特徴は，RDF [16] と異なり，拡張されたデータモデルをコーディングするため，独自の要素や文法を導入しない点である．任意の XML 文書を半構造オブジェクトとして解釈する方針を採用し，その解釈をサポートするいくつかの XML 属性を定義する．本章は，X# のオブジェクト，クラス定義，そして Java 言語ベースのアプリケーションインタフェースについて述べる（現在，X#仕様書やプロトタイプ実装は，公開に向けて準備中である）．

4.1 オブジェクト

X#では，プロパティ[label, type, value] は，XML エレメントで表現する．

$< labelxs : type = \text{“}type\text{”} > \boldsymbol{value} < /label >$

名前空間 xs:は，半構造オブジェクトの構造を表すための特別なセマンティクスを表す．xs:type は，プロパティの自己記述的な型の表現であり，省略した場合，データ型は String 型と解釈される．オブジェクトは，XML 要素でプロパティを囲むことで表現される．式 (3.8) の X#表現は，次のとおりである．

```
<person xs:oid="(Object-Identifeir)">
 <name xs:type="String"> Kimio Kuramitsu </name>
 <age xs:type="Integer/age"> 28 </age>
 <birthday xs:type="Long/date"> 1972-12-24
 </birthday>
 <birthday xs:type="Class/Date">
  <year xs:type="Integer/year"> 1972 </year>
  <month xs:type="Integer/month"> 12 </month>
  <day xs:type="Integer/day"> 24 </day>
 </birthday>
</person>
```

上例では，<person> は，オブジェクトコンストラクタであるが，その要素名は意味がない．代わりに，xs:oid によるオブジェクト識別子が，コンストラクタを表す．同様に，<birthday xs:type="Class/Date"> 要素も,（名前なし）オブジェクトである．ここでは，データ型 Class が，オブジェクトコンストラクタを表している．

半構造オブジェクトでは，オブジェクト上での同一プロパティ名（定義 3.4）の衝突は認めていないが，X#ではセットとして解釈する．たとえば，次の X#表現は，

```
<object xs:oid="(Object-Identifier)">
  <author> 倉光君郎</author>
  <author> 坂村健</author>
</object>
```

半構造オブジェクトとして，次のように解釈される．

[author, String, {“倉光君郎”, “坂村健”}] .

X#は，基本的に任意の XML 文書を半構造オブジェクトにマッピングできるように設計されている．ただし，通常の XML 文書は，必ずしも概念ラベルとデータ型の分離が行われていない．半構造オブジェクトとして，正しくコーディングされた XML 文書（つまり X#データ）は，その要素名に X#の名前空間（http://ubiquitous.jp/XSharp/）を割り当てる．

4.2 クラス定義

W3C は，XML の構造スキーマ言語として XSD（XML Structure Definition）の仕様化を行っている．XSD 仕様は，半構造オブジェクトのシンプルなクラス定義（定義 3.7）にはオーバスペックであるが，X#では次のように XSD 用の検証ツールとの相互運用性を図ることが可能である．

```
<xsd:complexType name="Date">
  <xsd:element name="Year" type="xsd:integer"
    xs:type="Integer/year"/>
  <xsd:element name="Month" type="xsd:integer"
    xs:type="Integer/month"/>
  <xsd:element name="Year" type="xsd:integer"
    xs:type="Integer/year"/>
</xsd:complexType>
```

4.3 Java アプリケーションへのインタフェース

X#プロセッサは，XML 上の半構造オブジェクトを操作するプロセッサで，大量のオブジェクトを同時に扱うより，個々のオブジェクトを操作することを目的に実装されている．オブジェクトの XML への入出力は，標準の SAX2.0/DOM2.0 を利用する．データアプリケーション側からのアクセスは，すべてビュー（View）を通して行う．ビューは，Java2 コレクションフレームワークの Map インタフェースを実装し，プロパティパスをキーにして，データ値の操作（get/put）を行うことができる．次は，半構造オブジェクトをアクセスする Java コードのサンプルである．

```
SemiObject obj
  = SemiObject.getInstance(org.w3.dom.Element);
Map view = obj.getView(ViewSpec);
view.put("age(Integer/age)", new Integer(28));
Long bd = (Long)view.get("birthday(Long/date)");
```

X#プロセッサは，ビューによってオブジェクトの構造検証やアクセス制約を付加することができる．つまり，ビュー仕様を満たさないオブジェクトのビューは生成できないし，逆にビュー仕様外のプロパティへのアクセスは制限される．さらに，X#プロセッサは，統合ビューアーキテクチャを採用し，ビューを生成するとき，ビュー仕様を満たすように自動的に型変換を行うように拡張できる．

5. リレーショナルデータベースの評価

今日の情報システムにおいて，大量のデータを効率良く処理することは必須である．リレーショナルデータベースは，多くの研究者やデータベースベンダの長年にわたる継続的な改良により，きわめて優れたパフォーマンスを実現する．リレーショナルデータベースは，大量の半構造オブジェクトを効率良く管理するために適したデータプラットホームといえる．また，データ構造の観点から見れば，半構造オブジェクトのビュー（定義 3.6）とリレーショナルモデルのリレーション[6]は同じである．つまり，アプリケーション単位のデータ統合では，ビューを直接リレーションに置き換えることができる．

本章では，半構造オブジェクトの不規則な情報構造（オブジェクトビュー）をそのままデータベースに蓄積し，そこから条件に従ってオブジェクトグループを選択する手法とその評価を行う．

5.1 リレーションへのマッピング

半構造オブジェクトは，自己記述的でプロパティの構成は不規則である．この構造を直接，固定的なリレーションにマッピングするのは難しい．データ型ごとに，リレーションを分割して，プロパティを蓄積する方法を示す．

Step1. 半構造オブジェクトは，オブジェクト識別子，概念ラベル，データ型，データ値の四つ組からなるフラットなタプルに展開する．複合オブジェクトは，プロパティパス同様，ドット表記で概念ラベルを連結して表現する．次は，式 (3.8) のフラット構造への展開である（&1234 は，オブジェクト識別子の内部表現である）．

```
[&1234, name, String, "Kimio Kuramitsu"]
[&1234, age, Integer/age, 28]
[&1234, birthday, Long/date, 1972-12-24]
[&1234, birthday.year, Integer/year, 1972]
[&1234, birthday.month, Integer/month, 12]
[&1234, birthday.day, Integer/day, 24]
```

Step2. 半構造オブジェクトのフラット表現をデータ型ごとに分類し，それぞれ [オブジェクト識別子，概念ラベル，データ値] からなるテーブルに格納する．たとえば，Integer/month 型のリレーションは，Integer_month_tbl (oid int32, label text, value int32) になる．

5.2 クエリー変形

リレーション上にデータ型分割して蓄積された半構造オブジェクトは，直接，SQL を用いて問い合わせることができない．そのため，ユビクタスデータモデルの仮想プロセッサモデルに従い，半構造オブジェクトへの問合せを SQL に変形する必要がある．オブジェクト上のプロパティに対する条件が 1 つだけの場合：

```
select xs:oid
where birthday.month(Integer/month) = 12
```

は，次のように SQL 変換する．

```
select oid from Integer_month_tbl
where label = 'birthday.month' and value = 12
```

このクエリー変形の方法では，オブジェクト上のプロパティに対する条件が複数になった場合，SQL 上ではテーブルをまたいだ処理が必要になる．このときは，テーブルごとに条件処理を行い，その結果をユビクタスデータプロセッサ上で結合することで高速化する．

5.3 実験と評価

我々は，リレーショナルデータベースに対するユビクタスデータプロセッサを Java 言語でプログラムして，オブジェクトを検索する実行効率の評価実験を行った．

5.3.1 実験データ

我々は，実験評価用データとして，民間企業から提供を受けた書籍カタログを用いた．**図 3** は，書籍カタログを X#で記述したサンプルである．本論文では，

```
<?xml version="1.0" ?>
<Book xs:oid="00000001-00CE0A01:8875A41C-366CD440-88 F12C2A-6F4C9289">
  <Title>電脳新世紀</Title> <Subtitle>インターネットの新しい未来</Subtitle>
  <Author>Mark Stefik </Author> <Editor>Mark Stefik </Editor>
  <Translator>近藤智幸</Translator> <Editor>石川千秋 </Editor>
  <Size> A5 版</Size> <Page xs:type="Integer">496</Page>
  <Price xs:type="Integer/currency-yen">2500 </Price>
  <ISBN xs:type="String/isbn"> 4-89362-159-9</ISBN>
</Book>
```

図 3 X#版書籍カタログの例
Fig. 3 An example of book catalog in X#.

表 1 実験結果
Table 1 Experimental results.

	SELECT oid	ヒット数	ユビクタス プロセッサ	ネイティブ プロセッサ
TEST1	WHERE Price(Integer/currency-yen) = 800	313	91 ms	81 ms
TEST2	WHERE Price(Integer/currency-yen) between 780 and 800	678	170 ms	160 ms
TEST3	WHERE Author(String) = '中村雄二郎'	2	20 ms	20 ms
TEST4	WHERE Author(String) LIKE '中村'	461	20 ms	20 ms
TEST5	WHERE Author(String) LIKE '中村' AND Price(Integer/currency-yen) between 780 and 800	43	280 ms	210 ms

この種のデータを 10 万件用いて，オブジェクトの検索効率を評価する．さらに，ネイティブ SQL との検索効率を比較するため，書籍カタログのスキーマを次のとおり定義し，測定を行った．

```
create table book (id int4, title text,
   subtitle text, series text, author text,
   editor text, translator text,
   illustrator text, photo text,
   publisher text, year int4, format text,
   pages int4, price int4, isdn text);
```

5.3.2 実装サーバ

我々は，評価用のリレーショナルデータベースとして，フリーウェアでかつ多くの実用実績のある PostgreSQL を用いた．実験サーバ環境は，次のとおりである．

実験サーバ環境

PentiumIII 600 MHz (Dual)
Redhat Linux 6.2J / PostgreSQL-7.02
8 GB Ultra-Wide SCSI / 512 MB DIMM

PostgreSQL サーバへのアクセスは，すべて JDBC ドライバを経由して行う．また，テーブル間の結合は，ユビクタスプロセッサ上の専用の集合計算ライブラリを用いている．

5.3.3 実 験 結 果

我々は，5 種類の検索式を各 100〜300 回試行し，その処理時間を計測した．**表 1** は，各検索式の 1 回の検索にかかる時間の平均を表している（それぞれのテーブルは，BTREE インデックスを作成してパフォーマンスチューニングを施している）．リレーショナルデータベース上の半構造オブジェクトは，単純な問合せの場合，ネイティブリレーションに対してほとんどオーバヘッドなく，実用的な速度で検索できることが示された．

6. 結 論

我々は，インターネット時代のデータモデルを考える鍵は，データの遍在性であると考える．つまり，データは様々なところに存在し，また組織の間を自由に移動できなければならない．そのためには，単純にシンタックスを統一するより，より抽象的なデータモデルの統一が重要である．我々は，様々なデータプラットホームの上で利用可能である新しいデータモデル，半構造オブジェクトをモデル化した．本論文では，半構造オブジェクトの XML 上のコーディング X#と，リレーショナルデータベース上の蓄積を示し，両者の相互運用性を示した．

半構造オブジェクトのもう 1 つの特徴は，概念モデルを導入しやすいように，概念とデータ構造を独立させた点である．今後は，半構造オブジェクトをベース

にした異種データや意味相互運用性に関して，議論を深めていくつもりである．

謝辞 本研究を進めるにあたり様々なご意見，ご討論をいただいた重定如彦氏と村上直氏（東京大学大学院理学系研究科）に深謝いたします．

参 考 文 献

1) Abiteboul, S., Buneman, P. and Suciu, D.: *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers (1999).
2) Adam, N. and Yesha, Y.: Strategic Directions in Electronic Commerce and Digital Libraries: Towards a Digital Agora, *ACM Computing Surveys*, Vol.28, No.4, pp.818–835 (1996).
3) Bancilhon, F., Delobel, C. and Kanellakis, P. (Eds.): *Building an Object-Oriented Database System: The Story of O2*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers (May 1992).
4) Biron, P.V. and Malhotra, A.: XML Schema Part 2: Datatypes, *W3C Candidate Recommendation* (2000).
5) Bosak, J.: XML, Java, and the future of the Web, *Proc. GCA XML Conference 97* (1997).
6) Bray, T. and Paoli, J.: Extensible Markup Language (XML) Specification 1.0, *W3C Recommendation* (Feb. 1998).
7) Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J.: The TSIMMIS Project: Integration of Heterogeneous Information Source, *IPSJ Conference* (Oct. 1994).
8) Chen, P.P.: The Entity Relationship Model—Toward a Unified View of Data, *ACM Trans. Database Syst.*, Vol.1, No.1, pp.9–16 (1976).
9) Cluet, S., Delobel, C., Simeon, J. and Smaga, K.: Your Mediators Need Data Conversion!, *Proc. ACM SIGMOD98*, pp.177–188 (June 1998).
10) Codd, E.F.: A Relational Model for Large Shared Data Banks, *CACM*, Vol.13, No.6, pp.377–387 (1970).
11) Codd, E.F.: Extending Database Relations to Capture More Meaning, *ACM Trans. Database Syst.* (1979).
12) Hull, R. and King, R.: Semantic database modeling: Survey, applications, and research issues, *ACM Computing Survey.* Vol.19, No.3, pp.201–260 (1987).
13) 倉光君郎，坂村 健：PCO：インターネット上における分散的なスキーマに対応した複合 EC コンテンツ記述言語，情報処理学会論文誌，Vol.41, No.1, pp.110–122 (2000).
14) Kuramitsu, K. and Sakamura, K.: Distributed Object-Oriented Schema for XML-based Electronic Catalog Sharing Semantics among Businesses, *Proc. 1st International Conference of Web Information Systems Engineering* (*WISE2000*) (2000).
15) Lassila, O. and Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification, *W3C Recommendation* (1999).
16) Papakonstantinou, Y., Gupta, A., Garcia-Molina, H. and Widom, J.: Object exchange across heterogeneous information sources, *Proc. 11th International Conference on Data Engineering*, pp.251–260 (Mar. 1995).
17) Sciore, E., Siegel, M. and Rosenthal, A.: Using Semantic Values to Falilitate Interoperability Among Heterogeneous Information Systems, *ACM Trans. Database Syst.*, Vol.19, No.2, pp.254–290 (1994).
18) Stoneraker, M.: New Data Models, *Reading in Database Systems*, pp.369–373 (1988).
19) Thompson, H.S., Beech, D., Maloney, M. and Mendelsohn, N. (Eds.): XML Schema Part 1: Structures, *W3C Recommendation* (Sep. 2000).
20) Yokota, K., Tsuda, H. and Morita, Y.: Specific Features of a Deductive Object-Oriented Database Language Quixote, *Proc. ACM SIGMOD Workshop on Combining Declarative and Object-Oriented Databases* (*SIGMOD'93 WCDOOD*), pp.89–99 (May 1993).

(平成 13 年 6 月 29 日受付)
(平成 13 年 8 月 2 日採録)

(担当編集委員 宝珍 輝尚)

倉光 君郎（正会員）

1972 年生．1996 年東京大学工学部卒業（機械情報工学）．1998 年同大学大学院理学系研究科修了（情報科学）．2000 年同大学院博士課程中退．現在，東京大学大学院情報学環助手．電子商取引，半構造データ，インターネットマーケッティング技術に興味を持つ．IEEE，ACM 各会員．

坂村　健（正会員）

東京大学大学院情報学環教授．1984 年より TRON プロジェクトリーダーとして新しい概念に基づくコンピュータ体系の構築に精力を注ぐ．TRON は現在，携帯電話，デジタルビデオ，エンジン制御等，組み込みシステムの世界で最も使われている OS である．さらに最近はプロジェクトの最終目的である超機能分散システムの研究を進めている．