データベースシステム 101−13
（1995. 1. 27）

文書データベース管理システムXebecの概要

中津山 恒　京嶋 仁樹　奥村 洋　安松 一樹　安藤 俊明
内田 剛　千葉 和也　沼田 賢一　上林 憲行

富士ゼロックス (株) システム・コミュニケーション研究所

現在, ODA やSGML などの文書アーキテクチャの国際規格や, 文書エディタ/ワードプロセッサなどが多数存在するため, 電子文書の編集可能な形での交換・共有が難しく, 再利用を阻害している. Xebec は, このような問題を解決する文書データベース管理システムで, 文書アーキテクチャや文書タイプを仮想化して電子文書を管理する. ユーザは, 文書アーキテクチャや文書タイプを意識せずに文書の格納と検索を行なうことができる. Xebec は, 文書に付与された属性および論理構造の要素のタイプ・属性・テキスト内容を複合的に利用した高度な検索機能をもつ. 本稿では, 機能とデータモデルを中心に, Xebec の概要を説明する.

An Overview of Xebec Document Database Management System

Hisashi Nakatsuyama　Masaki Kyojima　Yoh Okumura　Kazuki Yasumatsu　Toshiaki Ando
Go Uchida　Kazuya Chiba　Kenichi Numata　Noriyuki Kamibayashi

Systems and Communications Lab., Fuji Xerox Co.,Ltd.
Godo-cho 134, Hodogaya-ku, Yokohama-shi, Kanagawa, 242 Japan

Since there are some international standards for document architecture, such as ODA and SGML, and a lot of proprietary document formats of word processors, it is difficult to interchange and share electronic documents in editable forms and thus documents can hardly be reused. Xebec is a document database management system that is a solution for this problem and virtually incorporates document architectures and document types. Users of Xebec can store and retrieve documents without considering the differences between document architectures or document types. Xebec provides a sophisticated document query function, by which users can specify queries using attributes associated with documents, types, attributes, and text contents of logical structures. This paper describes an overview of Xebec document database management system, focusing on its functionalities and data model.

1 背景

製品開発, 部品調達, 大規模なシステムのマニュアル作成などの現場では, 膨大な文書をいかにして効率的に管理するかが問題となっている. オフィスでも, 文書作成をワードプロセッサで行なうのが一般化するにつれ, 電子文書の性質を利用して, 文書の再利用を図りたいという要望が強くなっている. このため, 文書データベース管理システム (文書 DBMS) の研究開発が迫られている.

多くの場合, 電子文書はワードプロセッサの内部形式で表現されている. このような文書は, 他のアプリケーションから利用できない. 形式が公開されている場合でも, フォントやレイアウトなどの指定のための情報と文書の内容が混在していることが多いため, フォント指定などを除いた, 本来の文書内容を抽出するのが難しい. また, 文書の構造を表現するためのデータモデルが複雑であることも問題を一層難しくしている。

1.1 構造化文書

構造化文書では, 文書は通常, 章や段落などのオブジェクトの階層構造で表現される. この構造化された内容は論理構造と呼ばれる. 論理構造は人間にも機械にも認識される. 論理構造には特定の処理向けの情報, たとえばフォントやレイアウトの指定が埋め込まれず, 代わりに各要素が何であるかを示す情報が付与されている. フォーマッタなどの応用プログラムは, その情報を利用して要素ごとに必要な処理を行なう. このように, 構造化文書では, 特定の処理に依存しない形で文書が表現される.

構造化文書では, 文書のひながた (テンプレート) を定める文書クラスが存在する. 文書クラスは構文規則に相当し, そこから作られる文書 (文書インスタンス) の論理構造に現れ得る要素と, 要素がもち得る下位構造とを定める. 文書クラスと文書インスタンスの例を図 1と図 2 に示す. 文書クラスをユーザが定義できる文書アーキテクチャもあるが, 事前に文書クラスが定まっているものもある. 明示的な文書クラスがないアーキテクチャでは, 唯一の文書クラスがあらかじめ定められていると考えてよい.

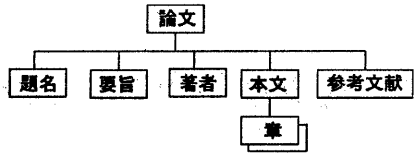

図 1: 文書クラスの例

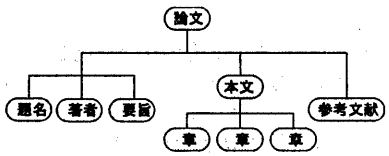

図 2: 文書インスタンスの例

1.2 文書アーキテクチャ

文書を表現する情報構造のことを文書アーキテクチャという. 現在, さまざまな文書アーキテクチャが存在しており, 再編集可能な形での文書の交換 (以下, 単に文書交換と呼ぶ), 文書の内容の再利用を阻害する要因となっている.

構造化文書の国際規格として, ODA[1] や SGML[5] などが制定されている. ODA では, テキスト, 図形, イメージといった内容体系を定めている. SGML は, ODA とは対照的に, 特定の内容体系を定めず, 文書アーキテクチャを定めるための記法を定義しているので, メタアーキテクチャと言える. SGML の場合, DTD ごとにひとつの文書アーキテクチャを定めていると言ってもよい. 使用する内容アーキテクチャや要素集合をあらかじめ定義して, SGML のサブセッティングを行ない, 文書アーキテクチャを定めている規格は数多い. 米国では, 国防総省の CALS(Continuous Acquisition and Lifecycle Support) [3] をはじめとして, 航空業界, 出版業界, 自動車業界, 製薬業界などで SGML をもとにした文書の標準が定められ, 運用されている.

1.3 問題意識

文書を管理する現場では，さまざまな文書アーキテクチャ/文書クラスの文書の管理が必要になる．製品開発をしている部門について考えてみると，そこで書かれる製品の仕様書のほかに，マニュアルの原稿，部品調達のための文書などを交換し，管理しなければならない．文書交換の相手には他の部門や会社もある．その文書の性質や交換相手によって文書アーキテクチャや文書クラスが異なることがある．たとえば，マニュアルにはOSF DTD(計算機ベンダの文書型) を用いるが，製品の解説記事にはAAP DTD(米国出版業界の文書型) を用いる必要があるかもしれない．これらはいずれもSGMLをベースに定められたものであるが，文書アーキテクチャとしては別個のものである．このように，部門や会社の都合で，そこで管理する文書アーキテクチャ/文書クラスを統一するのは実際には無理であることが多い．この場合，異種混合文書データベースを管理しなければならないことになる．

Xebec は構造化文書を管理対象とする文書DBMSで，異種混合文書データベースの管理と，構造化文書がもつ情報を複合的に利用した高度な検索機能の提供とを目標としている．

2 機能概要

以下に，Xebecの機能の概要を示す．

- 異種混合文書データベースの管理
  複数の文書アーキテクチャ/文書クラスの複数の文書群からなる，異種混合データベースを管理できる．現在プロトタイプ中のシステムでは，Akane[6], CALS[4], RTF[2] の文書を管理する．[1]
- 単一の検索空間
  文書間の関係はユーザの意図 (検索式) によって定まると考え，キャビネットやドロワといった文書間の関係を階層的な形で静的に表現するメタファは採用せず，単一の検索空間を提供している．
- 論理構造を複合的に利用した検索
  タイプ，属性，テキスト内容といった論理構造の要素に関する条件と，親子関係や祖孫関係といった要素間の関係とを複合的に組合せた条件に基づく，高度な検索機能を提供する．
- 文書の識別子および属性による管理
  著者，文書名，要旨といった書誌的情報を表す属性により，文書を検索することができる．また，一旦検索した文書の識別子を用いて，文書の取り出し，変更 (上書き) などの操作ができる．文書の識別子は，一意で不変である．
- 複数のプラットフォームへの対応
  サーバ/クライアント方式により，多様な環境でユーザが作業できるようにする．Xebecアプリケーションインタフェース (API) が定義され，APIにしたがってクライアントを作成する．
- マルチユーザ環境での使用
  Xebec固有のユーザ/グループ管理機能をもっており，ひとつの会社あるいは部門で運用できる．グループは，ユーザおよびグループをメンバにできる．
- データベースの規模
  数十万から数百万文書を管理する．
- セキュリティ
  文書ごとにアクセスを制御する．アクセス制御リストは，ユーザとグループの情報を使って指定される．
- グラフィカルなユーザインターフェース
  ユーザが利用するツールだけではなく，データベース管理者向けのツールもグラフィカルユーザインターフェースを備えている．

[1] 日本語EUCコードが使えるよう，CALSのSGML宣言は変更している．

3 データモデル

Xebecの最初のプロトタイプであるXebec V1(以下，単にXebecと呼ぶ) では，ユーザはもとの文書アーキテクチャが何であったかを意識せずに文書を管理できる．このため，データベース中では，特定の文書アーキテクチャには依存しないデータモデルによって文書インスタンスを表現している．もとの文書データもデータベース中に格納され，データベース中の文書インスタンスとともに管理される．

3.1 文書スキーマ

文書スキーマは, 文書インスタンスが取り得る構造と, 文書および各要素の意味を規定するもので, 文書のテンプレートと言ってよい. 文書スキーマは, 報告書, 特許明細などの表現したい文書の種別ごとに, データベース管理者が定義することができる.

3.1.1 スキーマ定義

文書スキーマは, スキーマ名, 管理属性, 意味記述, タイプ定義の 4 つ組で定義される.

文書スキーマはスキーマ名で識別されるので, スキーマ名は一意である.

スキーマ管理属性は, スキーマ自身を管理するための属性である. どのような属性があるのかはXebecがあらかじめ定義しており, 文書スキーマを定義したときにその値を指定する.

意味記述は, そのスキーマがデータベース中でどのような意味をもつのかを示すもので, ユーザ間でスキーマの意味を共有するために用いられる. また, 自動処理にも利用される.

タイプ定義は, 文書スキーマから生成し得る文書インスタンスの構造を記述する.

3.1.2 タイプ定義

タイプ定義は, タイプ名, 意味記述, 属性定義, 内容定義, 構造定義からなる. 構造定義は, ノードと構造生成子による有向グラフで表現される. ノードには内容部を持つものと持たないものがある. 葉ノードはタイプ付けされたノードで, かならず内容部を持っている. 葉でないノードも内容部をもってよい.

タイプ定義はタイプ名で識別される. タイプの名前空間はひとつのスキーマで閉じているので, 異なるスキーマに属する同名のタイプ定義には意味的な関係はない.

意味記述は, そのタイプがもつ意味を表現する. これには, ユーザが読む情報と, マシンプロセッサブルな情報とがある. 前者は, ユーザが意味を理解するのを助けるコメントで, 自然言語で記述する. 後者は, 章, 節, 段落, 箇条書, 注, 引用, 参考文献などの汎用的な構成要素の列挙型で表現され, 自動処理に利用される.

属性定義は文書インスタンスのノードの持つ属性を定義するもので, 属性名と属性値の型の組で定義される. 許される属性値の型は, 文字列型, 数値型, 日時型, 人名型, 列挙型である.

内容定義は, 内容体系の型を指定する. 内容体系の型は, テキスト, 図形, イメージ, 数式, 表である. 内容体系を持たない場合は, 型 NULL で表現する.

構造定義は構造生成子とタイプ定義で表現される. 構造生成子を, 表 1に示す.

表 1: 構造生成子

構造生成子	意味
AGG	順列
SEQ	順序
OPT	有無
CHO	選択
REP	繰返し

文書スキーマの例を, 図3に示す. この文書スキーマから作られた文書インスタンスでは, ルートの直下に "前付", "後付", "謝辞" という 3 つのノードがある. "前付" は "題名", "要旨", "著者" からなる. "本文" は複数の "章" からなる. "後付" は, "参考文献" と "謝辞" からなる.

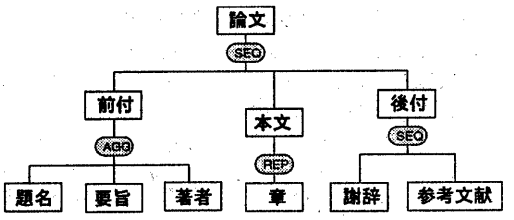

図 3: 文書スキーマの例

3.2 文書クラス

Xebecは, 5.1節で述べる対応づけのため, 管理対象の文書アーキテクチャで定義された文書クラスも管理している.

文書クラスは、文書スキーマのデータ構造にSGMLの包含(inclusion)と排除(exclusion)を追加したデータ構造で表現され、もとのデータと対にして管理される。管理対象の文書アーキテクチャで定義された文書クラスを格納すると、そのデータが解析され、Xebecのデータモデルで表現された文書クラスが生成される。

3.3 文書インスタンス

文書インスタンスは、文書の書誌的な情報を表現する管理属性と、文書内容を構造化した論理構造とからなる。文書インスタンスは、いずれかのスキーマをテンプレートとして生成される。したがって、論理構造は、スキーマ定義に従うものしか許されない。

論理構造は、有向順序木で表現される。各ノードは文書スキーマのいずれかのノードから作られたものである。文書スキーマのノードを、論理構造のノードのタイプと呼ぶ。ノードは、タイプの定義にしたがって、属性、内容をもつことができる。葉ノードは必ず内容をもっており、葉でないノードも内容をもつことができる。

内容体系には、テキスト、図形、イメージ、数式、表があるが、Xebecデータモデルで表現されるのはテキストだけで、それ以外の内容はもとの文書での表現そのものが格納される。

テキスト内容中に入れ子になって現れる内容は、下位構造として表現される。入れ子になって現われる内容もまた、有向順序木である。

文書インスタンスがもつ論理構造は、もとの文書アーキテクチャでの論理構造とは必ずしも一致しない。たとえば、Akaneでは箇条書は文字内容体系で表現されるが、これでは各項目を単位とするような処理を行いにくいので、箇条書を論理構造とすることがある。RTFでは段落などいくつかの論理構造しか記述できないが、それ以外の構造をもつことがある。

文書アーキテクチャによっては、ひとつの文書を、文書ファイルと補助的な別のファイルに分けて保持するものがある。例えば、SGMLの文書ファイルは、外部実体を参照してひとつの文書を表すこともある。また、Akaneの文書ファイル形式では、イメージを別の外部ファイルとして持つ[8]。データベース内では、もともと複数の文書ファイルで表現された文書であっても、ひとつの文書インスタンスとして操作できる。

4 文書操作機能

文書の検索には、文書ごとに付けられる管理属性に関する条件と、論理構造に基づく文脈情報とを用いることができる。文脈情報とは、タイプ・属性・テキスト内容といったノード内の情報と、親子関係・祖孫関係といったノード間の情報とを、複合的に組み合わせたものである。[7] 文脈情報を用いた検索条件により、検索結果を高度に絞り込むことができる。

文脈情報を用いた検索式の例を図4に示す。この検索式は、章見出しに“文書”という文字列を含む章で、図見出しに“DBMS”という文字列を含む図をもつものを検索する。

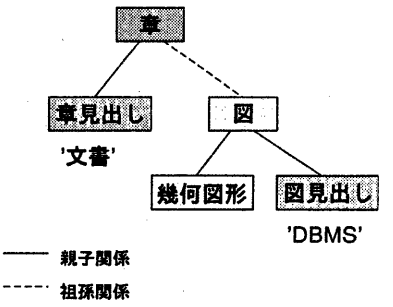

図 4: 文脈条件の例

検索結果は、文書または文書部品の識別子の列である。検索の結果得られた識別子を、ブラウジングや次の検索条件に利用できる。

管理対象の文書アーキテクチャで表現された文書あるいは文書の部品を、ひとつの文書インスタンスとして格納する。文書を更新するには、既存の文書の識別子を指定して文書を格納する。

削除の単位はインスタンスである。文書部品のみを削除することはできない。

5 文書アーキテクチャ変換と対応づけ

本節では、操作時に発生する、文書アーキテクチャの変換と、対応づけと呼ばれる処理について説明する。

5.1 対応づけの概要

3章で述べた通り, Xebec はもとの文書アーキテクチャのデータとこれに対応する文書インスタンスの両方を保存する. したがって, 文書を格納するときにはその文書に対応するインスタンスを生成しなければならない. このように, 特定の文書アーキテクチャの文書から Xebec の文書インスタンスを生成する処理を内部文書化 (internalization) と呼ぶ. 逆に, 文書を取り出すときには, Xebec の文書インスタンスから特定の文書アーキテクチャの文書を生成する. この処理を外部文書化 (externalization) と呼ぶ.

内部文書化は, 対応づけと呼ばれる処理により論理構造を再構成し, 得られた論理構造を, Xebec のデータモデルで表現することで行なう. 内容のうち, テキストだけがデータモデルで表現される. 一方, 外部文書化は, 対応づけにより論理構造を再構成し, 得られた論理構造を指定された文書アーキテクチャで表現することで行なう. この際, テキスト以外の内容は, もとの文書中のデータから, 指定された文書アーキテクチャのデータに直接変換される. 外部文書化のうち, もとの文書アーキテクチャと指定された文書アーキテクチャが異なる場合をとくに文書アーキテクチャ変換と呼ぶ.

対応づけは, 対応づけ規則と呼ばれる規則を, 対応づけ元の文書の論理構造に対して評価することによって行なわれる. 対応づけ規則は, 文書スキーマと文書クラスのもつ情報と, 個々の文書インスタンスがもつ情報とを用いて記述される.

Xebec では, 要素とその出現順序が似通っている文書スキーマと文書クラスだけを対応づけることができる. これは, 文書スキーマや文書クラスの設計者の意図とかけ離れた文書に対応づけることを避けるためである. また, 対応づけによって, テキストや図表などの文書の内容が欠落することはない.

対応づけの例を, 図5に示す. この例の入力は図1の文書クラス, 出力は3のスキーマである. 出力側の論理構造には, 入力側にはなかった, "前付", "後付", "謝辞" という3つのノードが付与されいる. "題名", "要旨", "著者" は集約化されて "前付" となり, "参考文献" は "謝辞" と集約化されて "後付" となる.

対応づける規則をいくつか用意すれば, 文書を格納するときに, 複数の文書アーキテクチャ/文書クラスの文書を, 同一の文書スキーマに対応づけることができる. 同様に, 取り出すときには, ある特定の文書スキーマの文書を, 複数の文書アーキテクチャ/文書クラスに対応づけることもできる. 図6は, ある文書スキーマの文書インスタンスを検索対象とし, 検索結果を Akane で取り出す例である. もとの文書アーキテクチャが何であったかに関わらず, 所望の文書アーキテクチャ/文書クラスで取り出すことができる.

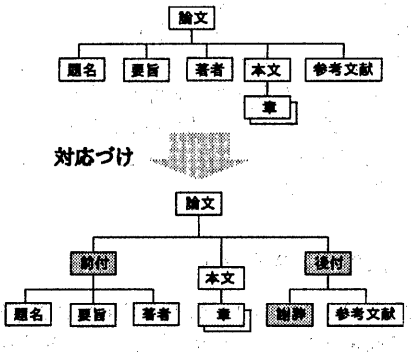

図 5: 対応づけの例

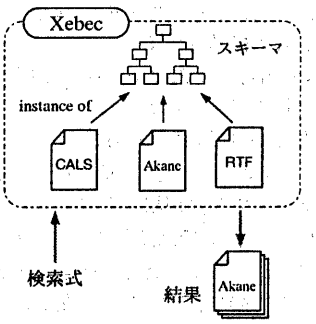

図 6: 複数の文書アーキテクチャの文書を同一スキーマに対応づけた例

5.2 対応づけ規則

対応づけ規則は文書スキーマと文書クラスをもとに記述される. いずれかの文書アーキテクチャの

文書部品を対象とする場合には，その部品についてだけ対応づけ規則を記述する．対応づけ規則は，対応づけ元，対応づけ先，および部品の根の3つ組に対して，高々ひとつだけ定義することができる．対象が文書部品でないときには，根は指定しない．

表 2: 対応づけ規則

	対応づけ元	対応づけ先	根
文書格納	文書クラス	文書スキーマ	—
部品格納	文書クラス	文書スキーマ	指定
文書取出	文書スキーマ	文書クラス	—
部品取出	文書スキーマ	文書クラス	指定

5.3 対応づけ処理の構成要素

対応づけ処理は，内容分割，対応処理，補完処理という3つの処理からなる．これらの処理の結果，生成された構造が出力側の文書クラスやスキーマの制約をみたしていれば，対応づけは成功である．

文書アーキテクチャや文書クラスの制約上，本来は論理構造の部分木あるいは部分木の列であるべき部分が，単一の文字内容にまとめられていることがある．たとえば，Akane では箇条書の各項目が単一の段落に入れられることがある．このような文字内容を分割し，本来の論理構造を抽出する処理を内容分割処理と呼ぶ．

入力インスタンスの論理構造の各ノード/属性/内容が，出力インスタンスの論理構造上のどのノード/属性/内容に対応するかを決定し，出力インスタンスの論理構造を生成する処理を対応処理と呼ぶ．

対応処理によって生成された出力インスタンスが，出力側のスキーマや文書クラスの制約を満たさない場合に，制約を満たすよう必要なノードや部分木を補う処理を補完処理と呼ぶ．

6 プロトタイプ

プロトタイプは，図7に示すように，カーネルとストレージシステムからなるサーバと，グラフィカルなユーザインタフェースを提供するクライアントからなっている．

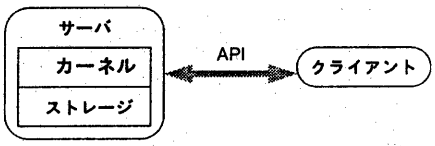

図 7: Xebec V1 プロトタイプの構成

サーバとクライアントの通信のため，アプリケーションプログラムインタフェース (API) が定義されており，これを使って所望の環境で動作するクライアントを作ることができる．API はテクスチュアルな言語ではなく，Xebec が提供する．C++のクラス群を使い，遠隔手続き呼び出し (RPC) のセマンティクスで通信する．たとえば，文書スキーマを定義するには，文書スキーマを表現するオブジェクト群を生成し，それを引数としてスキーマ定義のためのオペレーションを呼び出す．

ネットワークを介して授受されるデータにはネットワーク表現を使用しており，サーバとクライアントのホストの機種が異なっていても問題なく使用できる．

プロトタイプの開発環境を表3に示す．[2] また，画面イメージを図8に示す．

表 3: プロトタイプの開発環境

OS	Solaris 2.3
言語	C++
UI	OpenLook Interface Toolkit
ストレージ	ObjectStore

7 おわりに

本稿では，文書データベース管理システム Xebec について，機能とデータモデルを中心に述べた．

今後は，最初のプロトタイプである Xebec V1 で実際に文書を管理し，機能と性能の評価を行なう．と

[2] Solaris, OpenLook, ObjectStore は，それぞれ Sun Microsystems, UNIX System Laboratories, Object Design の登録商標です．

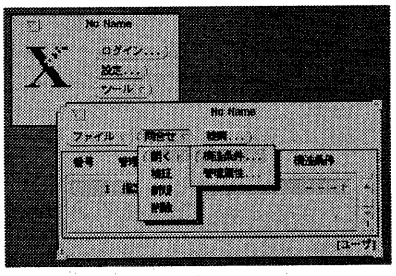

図 8: Xebec プロトタイプ

くに, 異種混合文書を扱う上で重要な, 文書アーキテクチャの仮想化の方法について重点的に検討する.

謝辞

小部正人副主任研究員には, 日頃研究を進める上で大変お世話になっている. 清水宏行氏は, 1994 年 7 月まで Xebec 研究グループのリーダを務められ, 研究の遂行に心を砕かれた. お二人への感謝の意をここに表する.

参考文献

[1] International Standardization Organisation. *Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format. ISO 8613*, 1989.

[2] Microsoft. *Rich Text Format. Microsoft Word Technical Reference. Chapter 10.*

[3] US Department of Defence. Military standard: Automated interchange of technical information (MIL-STD-1840), 1987.

[4] US Department of Defence. Military specification: Markup requirements and generic style specifications for electronic printed output and exchange of text. (MIL-M-28001B), June 1993.

[5] International Standardization Organisation. Information processing systems - text and office systems - standard generalized markup language (SGML). ISO 8879. 1986.

[6] 屋内恭輔, 保科孝之, 田口安男, 小林晴法, 西田賢一, 黒澤宏. 構造化ドキュメントエディタ Akane. 富士ゼロックス テクニカルレポート, pp. 98-105. 1993.

[7] 中津山恒, 楠本浩二, 村田真. 構造化文書の文脈情報に基づく文書操作システム. In *92-TCG-5-7*. 情報処理学会, January 1993.

[8] 富士ゼロックス (株). Akane 文書ファイルフォーマット説明書 - CForm 説明書 -, 1993.