オブジェクト指向データベースによるWWW-DB連携システムの高速化とWWWコンテンツ管理への応用

安村義孝†

オブジェクト指向データベース技術を利用したWWW-DB連携システムの高速化方式を提案し，その実装方法と応用アプリケーションについて述べる．WWWコンテンツ管理のような高度なWWWアプリケーションでは，複雑な構造を持つデータを格納可能なデータベースが必要であり，不特定多数の利用者から高速にアクセスできるWWW-DB連携システムが要求される．本システムでは，これらのデータを複合オブジェクトとしてオブジェクト指向データベースに格納し，拡張スクリプト言語によって複合オブジェクトをHTML文書形式に容易に変換する仕組みを提供する．さらに，高速レスポンスを実現するために，セッション管理とマルチスレッド制御を統合して，HTTPサーバと同一プロセス空間でオブジェクトキャッシュを共有する．性能評価の結果，CGIを利用した従来の連携方法に比べて数倍の性能向上を確認した．

High Performance WWW-DB Integration System using Object-Oriented Database and its Application for WWW Contents Management

YOSHITAKA YASUMURA†

The speedup methods of a WWW-DB integration system using object-oriented database technologies are proposed, and its implementation and its application are described in this paper. It is necessary to store complicated data into databases for advanced WWW applications such as WWW contents management. These applications demand WWW-DB integration systems which can process quickly a request from any user. In this system, the complicated data is represented as a composite object and is stored in an object-oriented database. The object can be easily converted into the form of an HTML document using a script language extended for composite objects. Moreover, session management and multi-thread control are integrated with the efficient sharing of the object cache on the HTTP server process in order to have high speed access to databases. As a result of its performance evaluation, this system is several times as fast as a conventional system using CGI.

1. はじめに

インターネットやイントラネット上で情報システムを構築する場合，WWW（World Wide Web）は必須なものになりつつあり，最近では従来のソフトウェアをWWW技術と連携させることが多くなってきた．これらのシステムを構築する際には，情報を共有するためのデータベースをサポートすることが重要である．そのため，様々なWWW-DB連携システムが開発されているが，ほとんどのWWW-DB連携システムは関係データベース管理システム（RDBMS）を採用している[6)10)]．WWWブラウザを利用してユーザがWWW-DB連携システムに検索条件を与え，それが内部的にSQL（Structured Query Language）の問合せ文に変換されてデータベースへアクセスし，タプルの属性データとして得られた問合せ結果は何らかの方法でテキストデータに変換されてHTML（HyperText Markup Language）文書に埋め込まれる．HTTP（HyperText Transfer Protocol）サーバとのインタフェースにはCGI（Common Gateway Interface）を利用しているものが多い．

RDBMSはデータベースサーバ上に存在するデータベースをRDBMS内のバッファ上でのみ扱い，データベースクライアントからはデータベースサーバにSQLやストアドプロシージャコールなどの要求を出すという処理手順になるため，WWW-DB連携システムでもデータベースサーバとHTTPサーバとの通信がデータベースアクセス要求の度に必要になってくる．それに対して，オブジェクト指向データベース管理システ

† NEC C&Cメディア研究所
C&C Media Research Laboratories, NEC Corporation

ム（OODBMS）はデータベースクライアント側にオブジェクトキャッシュを持ち，アプリケーションの実行に必要なオブジェクト（またはそのオブジェクトを含むページ）が最初にデータベースにアクセスした際にオブジェクトキャッシュにロードされる．もし，そのオブジェクトがスワップアウトされなければ，次のアクセス時にもオブジェクトキャッシュ上に存在することになるため，冗長な通信コストを避けることができる．

一方，CGIを利用したWWW-DB連携システムでは，データベースアクセス処理のためのプロセスが起動されるため，同時に複数のユーザからのリクエストを処理しなければならないシステムでは，高性能のサーバマシンが必要になってくる。そのため，幾つかの商用HTTPサーバには独自のサーバAPIを持つものもある．サーバAPIはCGIのようにHTTPサーバ上で実行させるアプリケーションモジュールに利用されるが，毎回新しいプロセスを起動させるのではなく，HTTPサーバと同一のプロセス空間で実行する．ユーザからのアクセス要求はHTTPサーバプロセスのスレッドに割り当てられることになる．したがって，それらのアプリケーションスレッドは同じメモリアドレスを共有することが可能になる．

以上のことを考慮して，OODBMS PERCIO[8)12)]を利用したWWW-OODB連携システムを開発した[16)17)]．PERCIOはマルチスレッド/マルチトランザクションに対応したOODBMSであるため，サーバAPIを経由したデータベースへのマルチスレッドアクセスが実現できる．WWW上でOODBMSにアクセスするためのアーキテクチャはいくつか研究されており[13)15)]商用システムも開発されている[1)11)]が，それらはデータベース内のオブジェクトをいかにしてHTML文書に埋め込むかということに主眼を置いている．本システムの様に，サーバAPIを利用してWWW-DB連携機能がHTTPサーバと同一プロセス空間で実行したり，オブジェクトキャッシュを共有したりすることで，システム全体の性能向上を目的にしたものは提案されていない．

本稿では，主にWWWコンテンツ管理に適用した事例を考え，本システムの特長について詳細に説明する．また，文書管理データベースによる基本的な性能評価の結果，本システムはCGIを利用する従来のシステム構成よりも数倍高速であることを確認した．第2節でWWW-DB連携システムのアプリケーションとしてWWWコンテンツ管理を取り上げ，それを実現するためのWWW-DB連携システムに期待される機能について触れる．第3節では，本稿で提案するWWW-OODB連携システムのシステム構成と内部処理の流れを述べる．主要機能である複合オブジェクト変換とマルチスレッド制御については，それぞれ第4節と第5節で詳細に説明する．第6節は本システムの性能評価結果に関する議論をする．最後の第7節で本稿をまとめる．

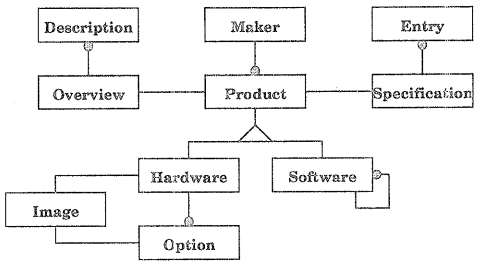


図1 製品情報データベースのスキーマ例
Fig. 1 Example schema of product information database

2. WWWコンテンツ管理

ここでは，WWW-DB連携システムのアプリケーション事例としてWWWコンテンツ管理を取り上げる．通常のWWWサーバにおいてはコンテンツデータをファイルとして保管し，各OSが提供するファイルシステムのディレクトリツリーと対応させて管理しているが，コンテンツのデータ量が多くなるにしたがって，新しいWWWページを追加したり，WWWページ間のリンク張り替えなどの保守作業が煩雑になってしまう．そこで，DBMSを活用することで大規模WWWサーバにおけるコンテンツ管理を容易にすることが望まれる．コンテンツデータをデータベースに格納することで様々なパターンの検索が可能になり，また，コンテンツデータの保守のためにDBMS保守ツールの利用や，専用のデータベースアプリケーションを作成することもできるようになる．ただし，データベースに格納されたコンテンツデータをWWWブラウザに表示可能なようにするための仕組みを提供しなければならない．

WWWコンテンツ管理の具体例として，パソコン関連の製品情報データベースを考える．各製品はハードウェアとソフトウェアに分類し，それらの製品情報をデータベースに格納する．製品情報データベースのオブジェクト指向モデルによるクラス構造を表すスキーマ例を図1に示す．オブジェクト指向モデルではコンテンツデータの入れ子構造を容易に表現でき，OODBMSはこのような複雑なデータ構造を持つコンテンツデータを管理することができる．これらのデータは複合オブジェクトを利用してデータベースに格納する．WWWコンテンツ管理にWWW-DB連携システムを適用する場合には，WWWブラウザ上にコンテンツデータを表示で

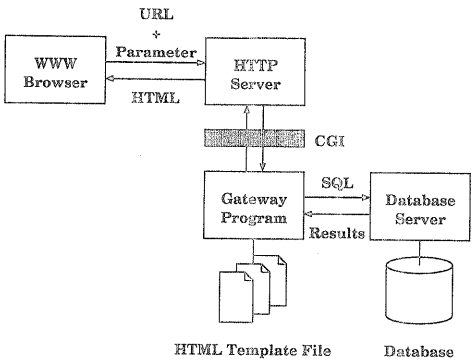


図2 WWW-DB連携システム
Fig. 2 WWW-DB integration system

きるようにするために，複合オブジェクトをどのように扱うかが重要になってくる．

典型的なWWW-DB連携システムのシステム構成を図2に示す．このWWW-DB連携システムでは次のような手続で処理が進められる．

(1) ユーザはHTMLのFORMタグを利用してHTTPサーバへアクセス要求を出す．その要求はURL（Uniform Resource Locator）と，データベースへの問合せ条件のためのパラメータを含んでいる．
(2) HTTPサーバはユーザからのアクセス要求を受け取ると，ゲートウェイプログラムのためのプロセスを生成する．
(3) 生成されたプロセス内で，HTTPサーバより渡されたパラメータからSQLの問合せ文を生成する．
(4) 生成された問合せ文をデータベースに対して発行し，検索結果をゲートウェイプログラムに返却する．
(5) 結果データを適切なテキスト形式に変換し，HTMLテンプレートファイル内で指定された位置に埋め込んでHTML文書を生成する．
(6) HTTPサーバにより，生成されたHTML文書をユーザに返却する．

上記のように，データベースの検索結果であるデータを使ってHTML文書を動的に生成することが可能であるが，いくつかの問題点がある．複合オブジェクトの様な複雑な構造を持つデータを扱う際には単一のSQLによる問合せでは不十分なため，ストアドプロシージャが利用されることがあるが，何度もデータベースへの問合せを発行すると効率が悪くなる．ゲートウェイプロセスとデータベースサーバの通信数が極端に増えてしまうからである．また，ユーザからのアクセス要求を受理する度にゲートウェイプログラムのためのプロセスを起動するのは，同じセッション内で何度もアクセスされる場合には冗長になってしまう．これらの問題に対して，CGIの利用をベースに考えられた高速化のためのシステム構成[4]で解決することは根本的に不可能であり，専用ポートを利用したプロトコルのやり取り[14]では汎用性がない．

一方，ゲートウェイプログラムはHTML文書を返却した後に終了してしまうため，セッションを継続するためには継続中のセッションに関する情報をどこかに保存しておく必要がある．保存されたセッション情報はそれ以降の同一セッション内のアクセス要求がある際に再利用される．しかし，データベースのオープン/クローズやトランザクションのスタート/コミットはアクセス要求がある度に行わなければならない．データベースのバッファ内に検索結果のデータを一時的に残しておくことができないので，絞り込み検索や複数ページの結果生成，更新処理の確認などのセッション継続を必要とする処理を行うためには，アプリケーションプログラムで工夫をしなければならないが，マルチユーザアクセスなどを考慮するとこれらの処理は煩雑になってしまう．

3. WWW-OODB連携システム

WWWコンテンツ管理のような高度なWWWアプリケーションを開発できる基盤となり，前節で述べた従来のWWW-DB連携システムが抱える問題点を回避するために，次の方針にしたがってWWW-OODB連携システムを設計・開発した．

- 商用HTTPサーバに提供されているサーバAPIとOODBMSが管理するオブジェクトキャッシュを利用してセッション継続を実現し，高速レスポンスを要求するWWWアプリケーションにも利用可能にする．
- データベースアクセスの処理手順は複数の方法（プログラミング言語）で記述でき，外部クラスライブラリなどとして存在するOODBMSが持つ拡張機能を利用できるようにする．
- 互いに様々な関連がつけられた大量のコンテンツデータが管理でき，そのコンテンツデータを使ってWWWサービスを容易に提供できるようにする．

本節では，上記のことを考慮してWWW-OODB連携システムのシステム構成について述べる．

3.1 システム構成

WWW-OODB連携システムのシステム構成を図3に示す．本システムはOODBMSとしてPERCIOを利用しており，PERCIOが管理するデータベースにアクセ

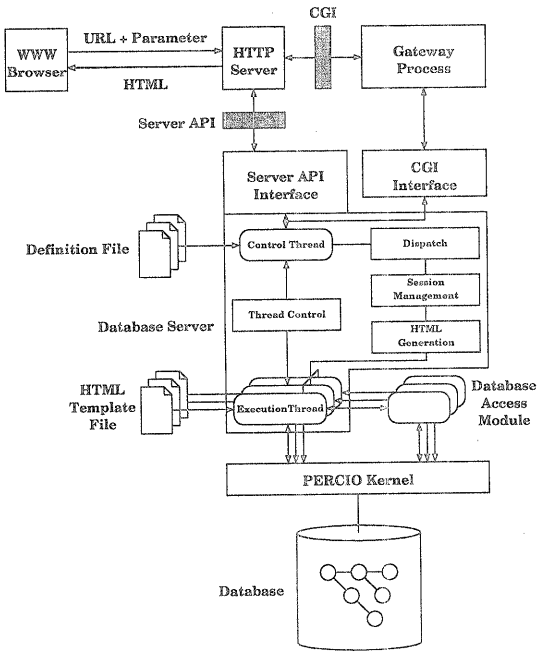


図3 システム構成
Fig. 3 System architecture

スすることを前提にしている．PERCIO は ODMG[2] の標準仕様に準拠した OODBMS であり，C++ バインディングと Java バインディングに基づいたデータベースプログラミング言語を提供している．また，SQL によるアクセスも可能である．

複数のプログラミング言語によるアプリケーション開発ができるようにするために，データベースアクセスを行うコンポーネントをその他の処理を行うコンポーネントとは分離した構成にした．WWW-DB 連携機能を処理するメインのモジュールはデータベースサーバであり，ユーザからのアクセス要求を振り分けたり，セッションの管理や HTML 文書の生成などを行う．データベースへの問合せなどの個々のアプリケーション機能に対応する処理はデータベースアクセスモジュールで行う．サーバ API としては ISAPI (Internet Server API)[5] をサポートする．また，汎用性を考慮して，HTTP サーバから CGI を利用した本システムとの連携を可能にするゲートウェイプロセスも用意している．

データベースサーバ内ではマルチスレッドで動作し，ユーザからのアクセス要求は個別のスレッドによって処理される．HTTP サーバから渡されたアクセス要求は制御スレッドで受理し，それに対応する定義ファイルを読み込む．そして，制御スレッドから実行スレッドに処理が移り，定義ファイルで指定されたデータベースアクセスモジュールがデータベースサーバに動的にロードされ，データベースアクセスの処理が行われる．データベースアクセスの結果データはテキスト形式に変換され，HTML テンプレートファイルの指定された場所に埋め込まれる．このようにして生成された HTML 文書をユーザ側の WWW ブラウザに返却する．

データベースサーバは次のようなモジュールから構成される．

- インタフェースモジュール
  HTTP サーバからサーバ API または CGI により URL とパラメータを受け取り，それらを制御スレッドに渡す．データベースアクセス後に生成された HTML 文書をユーザに返却する際にも各インタフェースを通して HTTP サーバに渡す．データベースサーバはサーバマシン上で常駐しているため，CGI を利用する場合はプロセス間通信でゲートウェイプロセスとやり取りを行う．また，パラメータ解析と文字列操作もこのモジュールで行い，その際にセッション情報とディレクトリ情報を取得する．
- ディスパッチモジュール
  データベースアクセス要求を適切な実行スレッドに割り当てる役割をする．各セッションは実行スレッドを保持しており，継続中であるセッションの実行スレッドがそのセッションのアクセス要求を扱う．新規のセッションの場合には，新たに実行スレッドが生成されることになる．実行スレッドの数は WWW サーバ管理者によって制限することも可能である．もし実行スレッド数が制限を越えてしまったときには，次のアクセス要求はキューに貯えられ，任意の実行スレッドが終了するまで待たされることになる．
- セッション管理モジュール
  連続するデータベースアクセス要求を一連のトランザクション処理として扱うためのセッション情報を管理する．セッション情報は実際にはセッション識別子としてデータベースサーバと WWW ブラウザの間で受け渡されることにより，セッションの継続をシステム側で保証する．同一のセッション内では，データベースのオープン / クローズ，トランザクションのスタート / コミットが任意の時点で発行可能である。もし指定された時間内にユーザからのアクセス要求がない場合には，自動的にそのセッションが破棄される．
- HTML 文書生成モジュール
  定義ファイルで指定された HTML テンプレートファイルの記述にしたがって，ユーザに返却する

HTML文書がこのモジュール内で生成される．データベースアクセスモジュールで取得した結果データを埋め込むために，HTMLテンプレートファイルには様々な制御構文が含まれる．

- スレッド制御モジュール
  ディスパッチモジュールで生成された実行スレッドを制御する．実行スレッドはデータベースへのアクセスやHTML文書を生成する主体である．実行スレッドが終了するまで，データベースサーバの実行スレッド情報と実行スレッドの処理に必要なリソース情報を維持する．

3.2 データベースアクセス

ユーザからのデータベースアクセス要求において，実行スレッドが利用するデータベースアクセスモジュールとHTMLテンプレートファイルは定義ファイルで指定する．データベースアクセスモジュールはソフトウェアコンポーネントであり，任意のクラスエクステントに対する問合せやオブジェクト遷移などのデータベースアクセス処理が記述されている．これらデータベースアクセスのコードは通常のPERCIOアプリケーションのコードと同様であるが，データベースのオープン/クローズやトランザクションのスタート/コミットを明示する必要はない．これらのコードはActiveXコントロール[3]としてコンポーネント化され，データベースアクセスを行う前にデータベースサーバに動的リンクされることになる．

データベースアクセスモジュールをActiveXコントロールにすることで，そのコードを記述するためのプログラミング言語には任意のものが利用できるようになる．PERCIOは現在のところC++とJava，SQLのためのAPIを持っており，これらのAPIを利用してActiveXコントロールを生成することで，データベースアクセスモジュールとして適用することができる．さらに，エクステント集合に対する問合せ結果を取得するような典型的なデータベースアクセスコードは，予め汎用のActiveXコントロールとして提供している．それらのActiveXコントロールを利用すると，プログラミング言語によるコードの記述をすることなく，WWWアプリケーションを容易に実現することが可能になる．

本システムのデータベースアクセスの仕組みを図4に示す．実行スレッドがデータベースアクセスモジュールを動的にロードした後に，そのコードを呼び出すことで目的のデータがデータベースから取得される．PERCIOデータベース上に構築された外部クラスライブラリもデータベースアクセスモジュール内で呼び出すことが可能である．HTMLテンプレートファイルによりHTML文書を生成する際には，実際のデータ値を取得するためにさらなるデータベースアクセスが行われる．これらの手続きはデータベースサーバが持つデータベースアクセス関数を使って処理される．

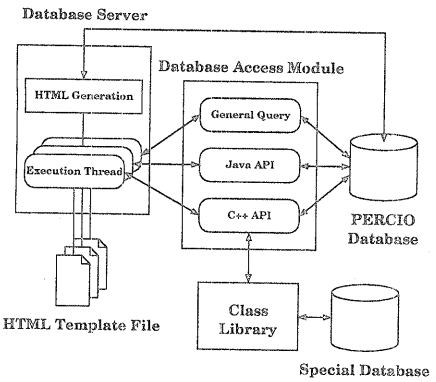


図4 データベースアクセス
Fig. 4 Database access

データベースアクセスの結果データは集合かオブジェクトへのリファレンスとしてデータベースアクセスモジュール内で取得される．このリファレンスはHTML文書生成時に利用され，結果データから必要なデータ値が取得される．HTMLテンプレートファイルを記述する際，データベースアクセスモジュールを作成したプログラミング言語に依存せずに結果データにアクセスするためのインタフェースが必要である．このため，データベースアクセスコードのいくつかの変数名（オブジェクト変数名）をActiveXコントロールのプロパティとしてエクスポートする．これにより，データベースアクセスモジュールの外部から共通のAPIでその変数値を取り出すことができるようになる．これらの関係はデータベースサーバ内のシステムテーブルで管理し，HTMLテンプレートファイルからHTML文書を生成する時に参照される．

4. 複合オブジェクト変換

データベースアクセスモジュールでデータベースにアクセスした後，HTMLテンプレートファイルの内容にしたがってWWWブラウザに返却するためのHTML文書を生成する．HTMLテンプレートファイルはHTMLを拡張したスクリプト言語で記述し，記述されたコードを解析しながらデータベースアクセス結果から必要なデータ値を埋め込んでいくのである．

4.1 拡張スクリプト言語

HTMLテンプレートファイルのフォーマットはIIS (Internet Information Server) [5]上に実装された

IDC (Internet Database Connectivity) の HTML 拡張ファイルに基づいている．HTML 文書をどのように生成するかを制御するキーワードに加えて，オブジェクト指向データベース内のオブジェクトリファレンスやコレクションを扱うためにその仕様を拡張した．具体的には以下のような変数が HTML テンプレートファイルで指定可能である．

- オブジェクト変数
  データベースアクセスモジュールからエクスポートされたオブジェクトへ参照するために利用される変数である．オブジェクト変数名は“<$”と“$>”で囲まれる．
- パラメータ変数
  WWW ブラウザから配送されてきたパラメータを取得するために利用される変数である．パラメータ変数は“<$”と“$>”で囲まれ，名前の先頭に“pdt.”を追加する．
- HTTP 変数
  HTTP サーバに接続している WWW ブラウザやオペレーション環境などの情報を保持するために利用される変数である．さらに，クライアントから送信された全てのヘッダ情報も利用することが可能である．HTTP 変数はオブジェクト変数と同様に“<$”と“$>”で囲まれる．
- システム変数
  集合演算を制御したり，セッション管理を行うために利用される変数である．

任意のオブジェクトのメンバ変数値を HTML テンプレートファイルに埋め込む時は必ずオブジェクト変数が使われる．オブジェクト変数はオブジェクトやコレクションへのリファレンスであるので，そのメンバ変数値を取得するためにドット式 (Dot Expression) を利用できるようにスクリプト言語を拡張した．オブジェクト変数にオブジェクトが束縛された場合には次のような式を利用する．

<$ (オブジェクト変数名) . (メンバ変数名) $>

この拡張タグに指定されたメンバ変数の型が何であれ，取得したデータ値を文字列に変換し，HTML テンプレートファイルの拡張タグが存在する位置に埋め込む．一方，オブジェクト変数にコレクションが束縛された場合には，foreach 文を利用する．このフォーマットは次の通りである．

```
<$ foreach (代入文) $>
<$ do [ (終了条件) ] $>
    (HTML テキスト)
<$ done $>
```

ここで，foreach 拡張タグの (代入文) は“(変数名) in (オブジェクト変数名)”の形式であり，(オブジェクト変数名) によって参照されるコレクションと，そのコレクションから 1 つずつ取り出した要素を (変数名) に束縛する．この変数は do と done の拡張タグで囲まれた範囲において，オブジェクト変数と同様に扱うことができる．これらの処理はコレクション内の要素がなくなるまで繰り返されるが，do 拡張タグの (終了条件) に“until (数値)”の形式で最大ループカウントを指定すると，多くても (数値) に指定された値だけループするように制限できる．

4.2 HTML 文書生成

複合オブジェクト変換の例として WWW コンテンツ管理における HTML 文書生成を考える．製品情報データベースではメーカ別に製品データが蓄積されており，各製品の概要とその記述というような複合オブジェクトを構成する．この複合オブジェクトを辿って HTML 文書を生成するために，foreach 文を利用して HTML テンプレートファイルを作成する (図 5)．foreach 文がネストされている場合には，深さ優先でその複合オブジェクトにアクセスされる．この規則を適用することで，複合オブジェクトの入れ子構造がそのまま，生成される HTML 文書に反映されるようになる．

HTML 文書生成の内部処理を図 6 に示す．複合オブジェクトの基点となるベースオブジェクトがデータベースアクセスモジュールによって取得されると，HTML 文書生成モジュールがアクセスできるように ActiveX コントロールのプロパティとしてエクスポートする．ベースオブジェクトを取得すると，そのオブジェクトのメンバ変数や下位のオブジェクトがアクセスされるかもしれないため，オブジェクトキャッシュにはいくつかの必要となるオブジェクトもロードする．HTML テンプレートファイルにドット式によるオブジェクトリファレンスがあれば，それらのオブジェクトはオブジェクトキャッシュにあるため，主記憶上でのみオブジェクトの遷移が行われることになる．

5. マルチスレッド制御

現在の仕様において HTTP は状態を保持しないプロトコルであるため，他の WWW サーバ系システムと同様にセッション継続の管理を独自に実現しなければならない．本システムでは，セッション情報はシステムでユニークなセッション識別子であり，セッションが継続されている間はこのセッション識別子が WWW ブラウザとデータベースサーバの間で受け渡される．また，WWW ブラウザのエラー等を検出することができない

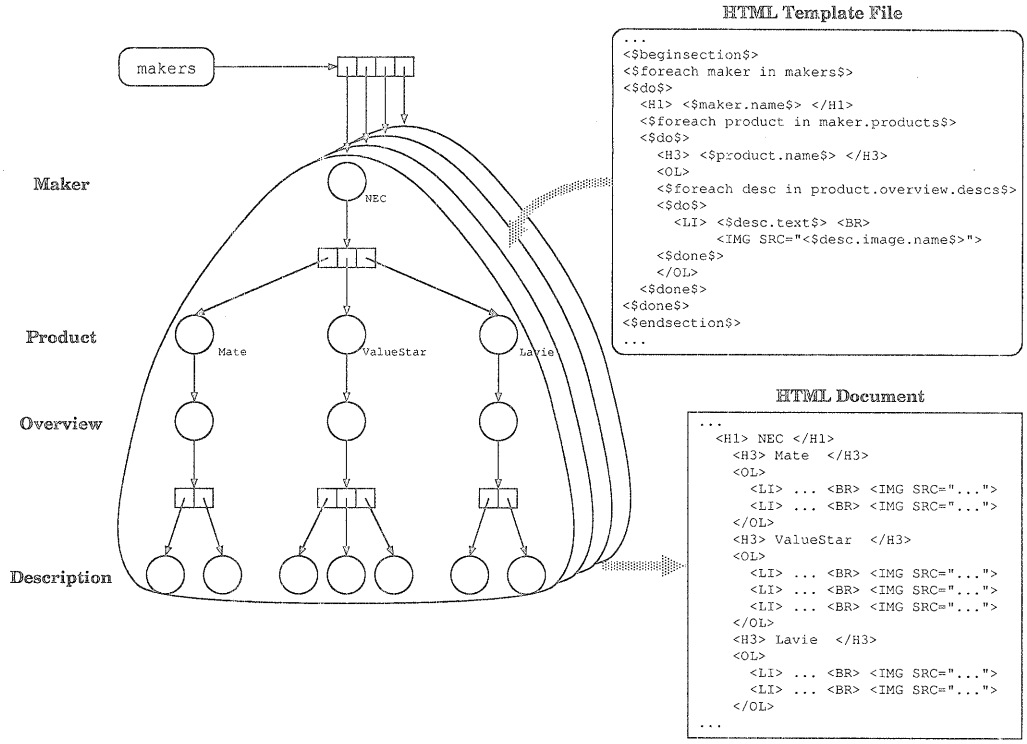


図5 複合オブジェクト変換
Fig. 5 Composite object conversion

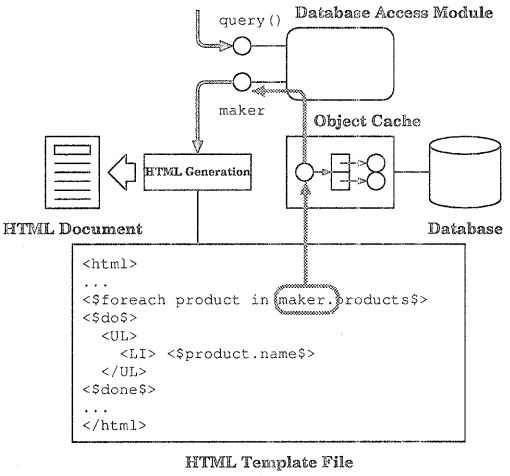


図6 HTML文書生成
Fig. 6 HTML document generation

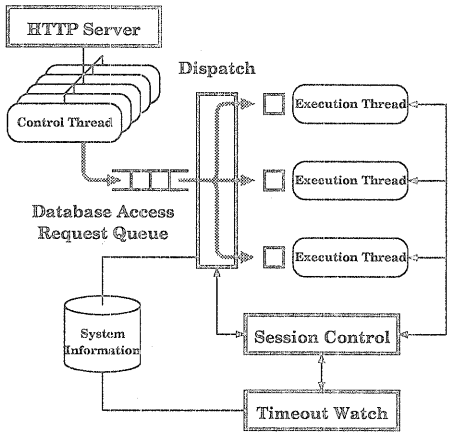


図7 マルチスレッド制御
Fig. 7 Multi-thread control

ため，指定された時間内に任意の処理要求が来なければセッションを強制的に終了させる機構も必要である．この場合，データベースサーバ内ではトランザクションのアボートとデータベースのクローズの処理が自動的に行なわれる．

5.1 セッション管理

各セッションはデータベースサーバ内の実行スレッドで処理が進む．実行スレッドのためのマルチスレッド制御を図7に示す．HTTPサーバは，WWWブラウザからのアクセス要求を受理したときに制御スレッドを生成して処理を開始し，HTML文書をWWWブラウザに返却するとこの制御スレッドは終了する．実行スレッ

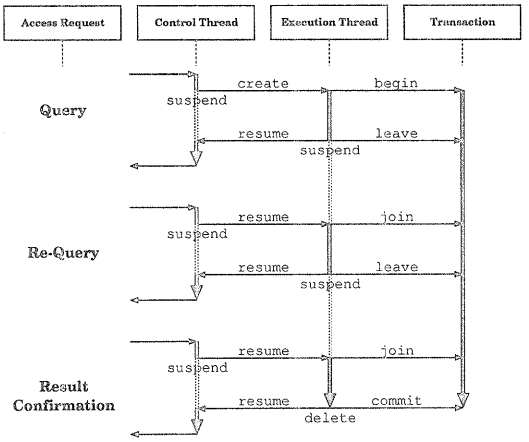


図8 セッション継続のスレッド制御
Fig. 8 Thread control in session continuous

ド数が設定情報として登録している最大スレッド数を越えていなければ，新しい実行スレッドを生成してそのスレッドに制御を移す．もし越えたときには，新しく受理したアクセス要求をデータベースアクセス要求キューに格納し，任意の実行スレッドが終了するまでそのアクセス要求の処理を延期する．しかし，処理待ちの時間がタイムアウトの時間を越えると，そのアクセス要求は削除されてWWWブラウザにはエラーが返却される．

セッション情報以外にも次のような資源の状態を保持しておかなければならない．

- HTMLテンプレートファイルで扱かう変数名を管理するオブジェクト変数テーブル
- PERCIOカーネルがデータベースアクセスのために利用するオブジェクトキャッシュ
- 継続されるトランザクション
- 上記トランザクションを実行しているスレッド

データベースアクセスモジュールでエクスポートされる変数と関数の名前はオブジェクト変数テーブルに登録してある．このテーブルをセッション内で共有することで，複数のWWWページにまたがる処理を実現している．また，セッションを開始したスレッドとそのスレッド内で実行するトランザクションは，オブジェクトキャッシュとトランザクションの実行状態を保持するために，HTML文書を生成した後にも残される．次のアクセス時には再びそのスレッドに処理を割り当てられるため，同一セッションで同じトランザクションを利用することが保証される．

5.2 セッション継続

継続するセッション内の制御スレッドと実行スレッド，トランザクションの関係を図8に示す．垂直方向において，制御スレッドと実行スレッドの実線はそれらのスレッドが実行中であることを表し，破線はそれらのスレッドが停止中であることを表している．トランザクションはコミットするまでは常にアクティブである．実行スレッドは`join`メンバ関数を呼ぶことでトランザクションオブジェクトに接続し，`leave`メンバ関数を呼ぶことでその接続を外す．これら`join`メンバ関数と`leave`メンバ関数はトランザクションクラスのメンバ関数である．また，水平方向の線は各スレッドで発行される演算である．この例では，1）検索要求を発行，2）他の条件を付け加えた後に再検索要求を発行，3）結果を確認，の3つの演算がある．各アクセス要求の手続きは次のように行なう．

(1) HTTPサーバによってデータベースアクセスを行うための制御スレッドが生成される．
(2) 新規のセッションの場合は新しく実行スレッドを生成するが，継続するセッションであれば該当する実行スレッドを再開する．
(3) 制御スレッドを中断した後に実行スレッドに制御を移す．
(4) 新規セッションの場合はデータベースアクセスのためのトランザクションを生成するが，継続セッションの場合は該当するトランザクションを実行スレッドに接続する．
(5) データベースアクセスが終了すると，セッションを継続する場合はトランザクションから接続を外し，セッションを終了する場合はトランザクションをコミットする．
(6) 制御スレッドを再開し，セッションを継続する場合は実行スレッドを中断するが，セッションを継続しない場合は実行スレッドを終了する．
(7) 生成されたHTML文書を返却し，制御スレッドを終了する．

6. 性能評価

本システムの基本性能を評価するために，マルチクライアント環境での性能測定を行った．性能比較の対象は以下のものである．

- サーバAPIを利用する本システム（`wapi`）
- CGIを利用する本システム（`wcgi`）
- PERCIOデータベースを直接アクセスするCGIプログラム（`cgi`）

`wapi`は本システムの特長を全て利用するものであるが，`wcgi`はWWW-DB連携機能をHTTPサーバと同一プロセス空間で実行することはできない．また，`cgi`は従来のシステム構成を想定したものであるが，WWW-DB連携機能は直接コーディングしてある．

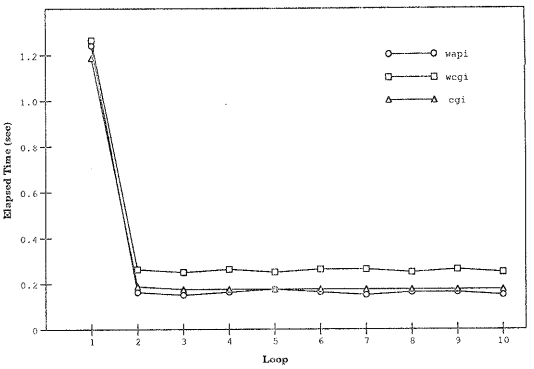


図9 ループ数による性能評価結果（1クライアント）
Fig. 9 Performance evaluation by loop (1 client)

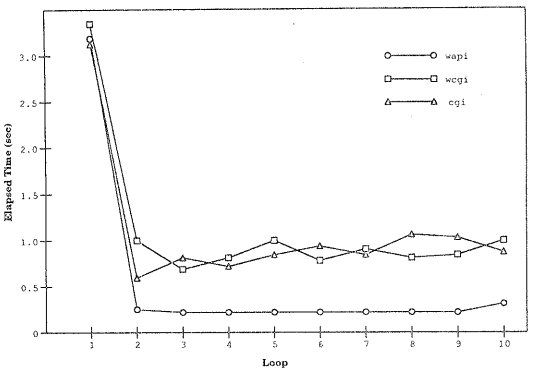


図10 ループ数による性能評価結果（3クライアント）
Fig. 10 Performance evaluation by loop (3 clients)

性能測定に利用したサーバマシンはNEC製Express 5800/160Pro（CPU: Pentinum Pro 200MHz x 2、主メモリ: 128Mバイト，OS: Windows NT Server 4.0）である．サーバと各クライアントはイーサネットで接続し，プロキシサーバは利用しない．クライアントプログラムは1つのWebMasterと複数のWebClientから構成する．これはHTTPサーバのベンチマークプログラムであるWebStone[7]を参考に構築した．WebMasterは全てのWebClientを同期させ，同時にアクセス要求を発行することを保証する．

性能評価のために利用したアプリケーション事例はHTML文書管理である．データベースに格納されるオブジェクトは，文書が10,000件，著者が1,000件，文書の概要が10,000件，キーワードが500件であり，全ての著者オブジェクトにはデータベース内でユニークな名前を付与する．これらのオブジェクトは互いに関連づけられている．HTML文書は外部ファイルで，そのファイル名だけを文書オブジェクトに登録する．性能測定の演算は、著者名により著者データを検索し，その著者名と組織名を含んだHTML文書として返却するものである．

1クライアントと3クライアントにおける測定結果をそれぞれ図9と図10に示す．これらは同じ演算を10回繰り返して測定したものである．ループ1以降の全てのループの測定結果がループ1のものよりかなりよくなっている．これはほぼ全ての必要なデータがデータベースからオブジェクトキャッシュに取り込まれているためである．1クライアントの場合はループ1後の全てのループで安定した性能が得られているのに対して，3クライアントの場合はwcgiとcgiの性能が不安定になっている．wcgiとcgiはCGIプログラムのための新規プロセスの生成による影響を受けているものと考えられる．一方，ループ1では全ての測定対象で同一の性能が得られた．wapiにおけるデータベースアクセスモジュールの動的リンクなど，付加的な処理手続きが性能差にはそれほど表れていないことが分かる．

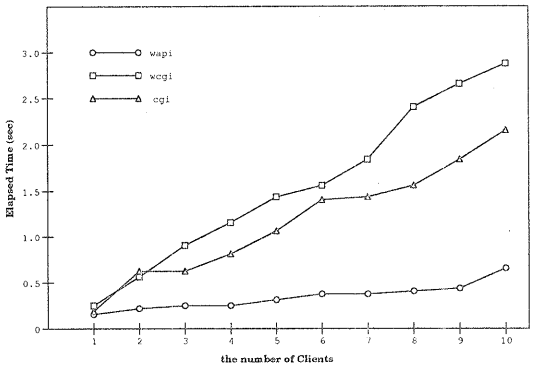


図11 クライアント数による測定評価結果（ループ2）
Fig. 11 Performance evaluation by clients (loop 2)

次に，クライアント数による測定結果を図11に示す．図9と図10からループ2以降の測定結果にそれほど大きな相違がないため，図11ではループ2の測定結果のみを描いてある．図11を見るとwapiの性能が最も優れていることは明らかである．wcgiとcgiの場合はクライアント数が増えると経過時間が極端に上昇してしまう．1クライアントと10クライアントでの測定結果を比較すると，wapiの性能低下の割合はwcgiとcgiのそれと比べてそれほど落ちてはいない．動的にHTML文書を生成する場合，同時アクセス数が増えると極端に性能が低下することが知られている[9]が，本システムを利用すればそれを回避できる．また，wcgiではWWW-DB連携機能のための手続きが含まれているため，その性能はcgiよりも悪くなる．

7. 結　　論

本稿では，WWWコンテンツ管理のような高度な

WWWアプリケーションに適用できるようなWWWとOODBを統合する方法とその高速化方式，それに基づいて開発したWWW-OODB連携システムの性能評価について述べた．OODBに格納された複合オブジェクトは拡張スクリプト言語の制御構文により，その構造を保持したままHTML文書形式に変換することができる．データベースアクセス処理はActiveXコントロールとして生成するため，その記述言語は何を利用してもよい．また，データベース連携機能はHTTPサーバが提供しているサーバAPIを利用してHTTPサーバと同一プロセスで実行するため，セッション管理とマルチスレッド制御を統合してある．マルチクライアント環境における基本的な性能評価の結果，本システムはCGIを利用した従来のシステム構成より数倍高速であることを確認した．アプリケーションサーバを利用した大規模WWWアプリケーションにも，本システムの高速化技法が役立てると考えられる．

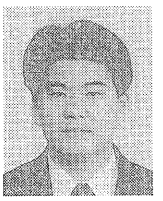
謝辞 本システムの設計および開発において，NEC第二コンピュータソフトウェア事業部およびNEC情報システムズ（株）の開発メンバには多大な協力をして頂き感謝致します．また，NEC第二コンピュータソフトウェア事業部の鶴岡邦敏部長には日頃から有益な御助言を頂いており感謝致します．

参考文献

1) Ardent Software, Inc.: *O_2Web*, http://www.ardentsoftware.com/
2) Cattell, R.G.G. and Barry, D.: *The Object Database Standard: ODMG 2.0*, Morgan Kaufmann (1997).
3) Chappell, D.: *Understanding ActiveX and OLE*, Microsoft Press (1996).
4) Hadjiefthymiades, S.P. and Martakos, D.I.: Improving the performance of CGI compliant database gateways, *Proc. 6th World Wide Web Conference*, pp. 573–585 (1997).
5) Microsoft Corp.: *Internet Information Server*, http://www.microsoft.com/japan/product/iis/
6) Nguyen, T. and Srinivasan, V.: Accessing Relational Databases from the World Wide Web: *Proc. 1996 ACM SIGMOD*, pp.529–540 (1996).
7) Mindcraft, Inc.: *WebStone*, http://www.mindcraft.com/benchmarks/webstone/
8) NEC Corp.: *PERCIO*, http://www.ace.comp.nec.co.jp/product/db/percio/p_main.htm
9) 日経BPシステムラボ: イントラネットPCサーバー－導入・活用・最適化－，日経BP (1996).
10) 日経データプロ: WWW-データベース連携システム構築法，日経BP (1996).
11) Object Design, Inc.: *ObjectForms*, http://www.odi.com/
12) 鶴岡邦敏，木村裕，波内みさ，安村義孝: オブジェクト指向データベース管理システムPERCIOの開発と今後の課題，電子情報通信学会論文誌，Vol. J79-D-I, No. 10, pp. 589–596 (1996).
13) Varela,C., Nekhavev,D., Chandrasekharan,P., Krishnan, C., Govindan, V., Modgil, D., Siddiqui, S., and Nickolavev, D.: DB:Browsing Object-Oriented Databases over the Web, *4th World Wide Web Conference* (1995), http://www.w3.org/pub/Conference/WWW4/Papers2/282/
14) Yamamoto, S., Kawasaki, R., Motoda, T., and Tokumaru, K.: Internet/Intranet Application Development System WebBASE and Its Evaluation, *IEICE Trans. Inf. & Syst.*, Vol. E81-D, No. 12, pp. 1450–1457 (1998).
15) Yang, J.J. and Kaiser, G.: An Architecture for Integrating OODBs with WWW, *5th World Wide Web Conference* (1996), http://www5conf.inria.fr/fich_html/papers/P31/Overview.html
16) 安村義孝: WWW-OODB連携システムを用いたコンテンツ管理の実現，情報処理学会第55回全国大会講演論文集 (3), 6X-3 (1997).
17) 安村義孝: オブジェクト指向データベースによるWWW-DB連携システムの高速化と性能評価，情報処理学会研究報告，Vol. 98, No. 57, 98-DBS-116(1), pp. 57–64 (1998).

(平成11年6月27日受付)
(平成11年9月27日採録)

(担当編集委員 加藤 和彦)



安村 義孝（正会員）

1992年慶應義塾大学大学院理工学研究科計算機科学専攻修士課程修了．同年日本電気（株）入社．現在C&Cメディア研究所主任．商用データベースの性能評価，オブジェクト指向データベース管理システム，WWW-DB連携システム，ディレクトリサーバなどの研究開発に従事．情報処理学会，日本ソフトウェア科学会各会員．