WWW-RDB 連携システムの開発

畑田 稔[†] 遠藤裕英[†]

WWW (World-Wide Web)-RDB (Relational Database) 連携システムの提案と評価を行った. RDB とのインタフェースは, HTML (HyperText Markup Language) ファイルと WWW ブラウザからは見えないサーバ上の sql (Structured Query Language) ファイルで記述される. キャッシュ機構の装備および CGI (Common Gateway Interface) を経由せず, RDB とのインタフェースを WWW サーバーに内蔵させることにより高性能化を実現した. WWW ブラウザで実現できるユーザー・インタフェースには限界があるため, よりユーザーフレンドリーなインタフェースに対応するため, RDB ビュアを開発した. 数種類のクエリについて性能を実測・評価し, 実用上十分な性能が得られることを明らかにした. オーバオールなレスポンスタイムは 1.16～2.74 秒であった. キャッシュ機構を働かせた場合, レスポンスタイムは 2.74 秒から 1.14 秒に短縮した.

Development of Relational Database Access over WWW

MINORU HATADA[†] and HIROHIDE ENDOH[†]

This paper discusses the architecture and performance of RDB (Relational database) access over WWW (World-Wide Web). HTML (HyperText Markup Language) file and sql (Structured Query Language) file which is invisible for user, are used to retrieve information. The implemented system has the caching mechanism and WWW Server-integrated RDB interface to improve performance. We have also developed a RDB viewer to support user-friendly interface, because current publicly-available WWW browsers do not allow sufficiently user-friendly interface. Proposed architecture are evaluated for several queries. The overall response time has been observed a 1.16 sec to 2.74 sec. Cache architecture has reduced response time from 2.74 sec to 1.14 sec.

1. はじめに

WWW (World-Wide Web) の普及により, WWW ブラウザがユーザーインタフェースの標準の地位を築きはじめ, DBMS (Database Management System) の分野でも, WWW との連携の動きが高まっている. DBMS を中心とする企業システムに WWW を導入するメリットは大きい. DBMS のフロントエンドとして WWW を利用すれば, アプリケーションを開発・配布する手間が大幅に削減できる. WWW サーバー側に 1 つのアプリケーションを開発するだけで, WWW ブラウザが使えるすべてのマシンから利用でき, ユーザーインタフェースの統一も図れる. WWW ブラウザを利用できるプラットホームは, パソコンからワークステーションまで幅広い[1),2)].

このような利点に着目して, 航空券予約システム, 情報検索システムなど WWW とデータベースの連携システムが研究, 開発されている[3),4),10),11)]. また, WWW と大規模な DBMS との連携はデジタルライブラリなどの分野で研究が進んでいる[5)].

この市場を狙って, 米国の Oracle 社, Sybase 社などのデータベース・ベンダーが, 95 年後半に WWW と DBMS を連携するツール[6),7)]を発表している. 国内でも 96 年前半から提供を開始する.

著者らは, エンドユーザーコンピューティングの観点から, Microsoft 社の RDB (Relational Database) 管理システム「Microsoft Access」☆ (以下では MS-Access と略記する) を以前から使用してきた. これまで, スタンドアロン環境で用いてきたが, WWW サーバーと密な連携を図るために, 今回, MS-Access を対象として, WWW-RDB 連携システムのプロトタイプを開発した.

本システムは以下の特徴を持つ.

(1) CGI (Common Gateway Interface) を用いず,

† 日立製作所システム開発研究所情報センタ
Systems Developments Laboratory, Information System Center, Hitachi, Ltd.

☆ Microsoft Access は米国 Microsoft Corp. の商品名称である.

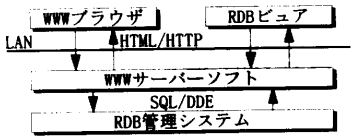

図1 WWW-RDB 連携システム
Fig. 1 Structure of RDB access over WWW.

WWW サーバー自体にその機能を持たせたこと，およびキャッシュ機能を備えることにより，高性能なシステムを実現した．

(2) HTML (HyperText Markup Language) ファイルに SQL (Structured Query Language) コマンドを書くのではなく，HTML ファイルとは別のファイルに SQL コマンドを記述する方式をとることにより，クライアント側で SQL コマンドを改ざんできないようにした．

(3) 現在，普及している WWW ブラウザでは実現できない，よりユーザーフレンドリーなインタフェースにも対処するため，イベント駆動型の RDB ビュアを併せて開発した．

本論文では，2 章で，WWW-RDB 連携システムの構成および動作原理について述べる．次に 3 章で，データベース操作を中心として WWW-RDB インタフェースについて説明する．4 章では，定型業務を対象として，よりユーザーフレンドリーなインタフェースを実現するために開発した RDB ビュアのアーキテクチャについて述べる．5 章で，提案システムの機能および性能評価を行い，提案システムの妥当性を検証する．

2. WWW-RDB 連携システムの概要

2.1 システム構成

WWW-RDB 連携システムの構成を図 1 に示す．主なプログラムは WWW サーバーソフトおよび RDB 管理システムである．

WWW サーバーソフトは，WWW ブラウザからのリクエストに応じて，HTML ファイルを WWW クライアントへ送信する機能のほか，RDB 管理システムに SQL コマンドを送り，検索結果を受け取ったり，データベースの更新（レコードの追加，削除，更新など）を行う機能を有している．

2.2 DDE インタフェース

WWW クライアントからのリクエストに応じて，HTML 形式のファイルとか，イメージ・ファイルなどを WWW クライアントへ送信するのが，WWW サーバーソフトの基本機能である．フォームを通して，WWW ブラウザのユーザーが入力したデータを処理するプログラムは，通常「バックエンド」プログラムと呼ばれ，WWW サーバーソフトとは独立したアプリケーション・プログラムである．このときの WWW サーバーソフトとバックエンドプログラムのインタフェースが CGI と呼ばれる．

RDB 管理システムとのインタフェースについても，1 つのバックエンドプログラムとして，構築することができる．しかし，本研究では，オーバーヘッドを削減するために，CGI を用いず，RDB 管理システムとのインタフェースを内蔵した WWW サーバーソフト全体を開発した．

上述のように WWW-RDB 連携には，CGI は用いていないが，WWW サーバーから見れば，RDB 管理システムは一種のバックエンド・プログラムである．WWW サーバーソフトとデータベース管理システムとのデータ交換には，Microsoft 社の Windows NT3.5, Windows 95☆の DDE (Dynamic Data Exchange) を用いた．DDE は，共有メモリによってアプリケーション間でデータを交換する，一種のプロセス間通信である．

DDE 機能をサポートする API (Application Programming Interface) では，文字列を直接，引数にすることはできず，事前に文字列ハンドルを作成し，DDE が完了した後，これを解放する必要がある．これに対して，Visual BASIC では，文字列を直接，引数にできるなど，より単純な DDE インタフェースが用意されている．本システムでは，C 言語で，これと同等の DDE インタフェースを構築した．DDE クライアントとして使用する関数を表 1 に示す．DDE としての観点からは，RDB 管理システムが DDE サーバー，WWW サーバーソフト側が DDE クライアントとなる．したがって，DDEInitiate 関数に与えるアプリケーション名は，この場合，RDB 管理システムとなる．

2.3 WWW サーバー–クライアント インタフェース

WWW のサーバーとクライアント間のインタフェースは HTML および HTTP (HyperText Transfer Protocol) で規定される．これらのうちで，WWW と RDB 連携に関係が深いのは，フォーム機能である．

フォームの HTML ソースの例を以下に示す．

☆ Windows NT, Windows 95 は米国 Microsoft Corp. の登録商標である．

表1 DDE サポート関数
Table 1 DDE support function.

関数名	引数	機能概要
DDEInitiate	application: アプリケーション名 topic: トピック名	DDE を開始する.
DDEExecute	channel: チャネル command: コマンド文字列	コマンドを送る.
DDERequest	channel: チャネル item: データ アイテム名	アイテム情報を求める.
DDEPoke	channel: チャネル item: データ アイテム名 data: サーバーに送るデータ	データを送る.
DDETerminate	channel: チャネル	チャネルを閉じる.

図2 フォームの表示例
Fig. 2 An example of display screen of form.

```
蔵書検索
<Form Method=POST Action=/sql/
                              findbook.sql>
  <Select Name=Field>
    <Option>書名<Option>著者<Option>発行所
  </Select>
  <Input Type=text Name=Word><p>
  <Input Type=submit Value=検索実行>
</Form>
```

このブラウザ表示例を図2に示す. たとえば, 検索項目として, プルダウンメニューから「書名」を選び, 検索語として,「WWW」を入力ボックスに設定して,「検索実行」ボタンをクリックすると, WWW ブラウザは, URL (Uniform Resource Locator) として,「/sql/findbook.sql」を, またデータとして「Field=書名&Word=WWW」をリクエストとして, WWW サーバーに送る☆. ここで, Field は, 図2に示されているように, プルダウンメニューに付けられた名前である. 同様に Word は, 入力ボックスに付けられた名前である. つまり, 名前=値 がセットとなり, 複数の項目があればそれらがアンパサンド (&) でつながれる.

入力ボックスは Type, Name, Value などの属性を持つことができる.「text」型の入力ボックスでは Value は初期値として, 表示される. また, Type には, 次の例のように「hidden」型がある.

```
<Input Type=hidden Name=SQL Value=
                              SQL_Command>
```

WWW ブラウザ画面には表示されないが, サーバーにはデータとして「SQL=SQL_Command」が送られる.

2.4 WWW サーバーソフトの処理概要

ユーザーが選んだフィールド名「書名」および入力した「WWW」から,「書名」に「WWW」を含むレコードを取り出したい場合, WWW ブラウザからの情報に基づいて「Where 書名 Like '*WWW*'」という抽出条件を作り, 全体として, SQL コマンド

```
Select 図書番号, 書名, 著者, 発行所
From 図書台帳 Where 書名 Like '*WWW*'
```

を作成して, RDB 管理システムに送ればよい☆☆.

SQL コマンドは「hidden」型で渡すことができる. この方式では, WWW ブラウザのダウンロード機能を使用して, HTML ファイルをクライアントに取り込み, SQL コマンドを書き換え, そして, この書き換えた SQL コマンドをサーバーに送り込むことができる. 通常, 企業内のデータベースシステムでは, 個々のユーザーが自由に SQL コマンドを改ざんして, データベースにアクセスできるのは好ましくないとの判断から, 本システムでは,「hidden」型による SQL コマンドおよびレスポンス情報の引き渡しは行っていない. SQL コマンドは, HTML ファイルとは別の sql ファイルと名づけたファイルに記述している. sql ファイルは HTML ファイルと同様に, サーバーに格納される. 蔵書検索 HTML ソースが参照している sql ファイル「findbook.sql」の例を以下に示す.

☆ 実際は, 漢字コード, 記号は1バイトのデータが%に続いて16進数2桁の合計3文字にコード化されて送られる. この場合, "Field=%8F%91%96%BC&Word=WWW" がサーバーが受け取るデータである.

☆☆ Like 述語に使うワイルドカード文字が ANSI SQL とは異なる. MS-Access SQL では, 任意の1文字, 任意長の文字列を「?」,「*」で表すが, ANSI SQL では, それぞれ「_」(アンダースコア),「%」で表す.

表2 ステートメント識別子
Table 2 Statement identifier.

名称	役割
Database	リレーショナルデータベース名
SQL, SQLn	SQL コマンド．n は数値 1, 2,···, 9
Format	選択クエリで得られたレコードの表示形式
Response	SQL コマンドの実行結果としてもどす情報
Break	選択クエリでチェック NG のときもどす情報
Cache	キャッシュ対象にする．
ReNew	キャッシングをクリアする．

表3 システム変数
Table 3 System variable.

名称	役割
_Records	選択クエリで抽出されたレコード数
_Now	このSQL コマンドの実行開始日時
_Date	このSQL コマンドの実行日
_IPA	クライアントのIP アドレス

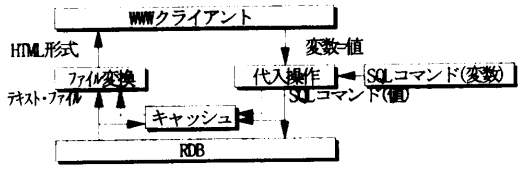

図3 WWW-RDB インタフェース
Fig. 3 Interface between WWW and RDB.

```
Database=WWW
SQL=Select 図書番号, 書名, 著者
  From 図書台帳 Where Field Like '*Word*'
Response=<Title>検索結果</Title>
  <H2>検索結果 (該当 _Records 冊)</H2>
```

ここで，Database，SQL，Response はステートメントの種別を表す．ステートメント識別子はこれらを含め，表2 に示すように 7 種ある．また，レスポンスデータに含まれる _Records はシステム変数である．本システムのシステム変数の一覧を表3 に示す．

図3 に示すブロック図を基に，WWW サーバーソフトの処理概要を以下に述べる．

ステップ 1：

2.3 節で述べたように，クライアントからのリクエストに URL が含まれている．URL から，リクエストされた sql ファイルの格納ディレクトリを求め，このファイルからステートメントを読み込む．

sql ファイル名の次に ? を付け，カンマ (,) 区切りの引数を持たせることができる．sql ファイル中の %1, %2, ··· は引数 1，引数 2，··· を表しており，URL に伴う引数の値に変換する．

ここでは，内容の判断は行わず，n 個のステートメントに分割する．

ステップ 2：

WWW クライアントから送られてきたデータをアンパサンド (&) を区切りとして，複数のステートメントに分割して，コード化された文字列を元に戻して，ステップ 1 で得たステートメントに付け加える．

ステップ 3：

ステートメントはすべて <name>=<value> の形となっている☆．

名前と値のペアに分解して，値を名前によって定められたエリアに格納する．

名前が予約語でなければ，変数名=値のペアであり，SQL コマンドおよびレスポンス中のこの「変数名」を「値」に置き換える．同様に，SQL コマンドおよびレスポンス中のシステム変数をそのときの値に置き換える．

ステップ 4：

SQL コマンドを実行する．1 回のリクエストに複数の SQL コマンドが存在しうるが，選択クエリはせいぜい 1 つである．

選択クエリの場合，Cache=1 のときは，この選択クエリの実行結果がキャッシュにあるかどうか調べる．キャッシュにあれば，そこから，該当レコードを取り出す．その他のケースでは，データベース管理システムに SQL コマンドを送り，該当レコードを抽出する．Cache=1 のときは，SQL コマンドと抽出されたレコードセットをキャッシュに格納する．

次にレコードセットを指定された表形式に変換して，先頭にレスポンス情報をおいた HTML ファイルを作成する．先頭のレスポンス情報にシステム変数 _Records が含まれていたら，抽出されたレコード数に置換する．

アクションクエリだけの場合には，データベース管理システムから戻される情報はない．SQL コマンドが正常に実行されたら，sql ファイルから与えられたレスポンスデータをクライアントに戻す．sql ファイルで定義しているレスポンスデータに，フォームからの入力変数，およびシステム変数が含まれていた場合，実際の値に置き換えたデータがクライアントに戻される．

アクションクエリで ReNew ステートメントがあれば，指定されたテーブル名がキャッシュされている SQL コマンド中に含まれていないか調べて，含まれていた場合には，その SQL コマンドとレコードセットを破棄する．

☆ <name> には 等号 (=) は含むことはできず，<value> には含むことができる．つまり，最初の等号 (=) が<name>と<value>の区切りとみなされる．

2.5 キャッシュ

データベースの内容は絶えず更新されているが，すべての情報が頻繁に変化しているというわけではない．滅多に変化しない情報を毎回，SQL コマンドを実行して，取り出しているのは，効率の良い方法とはいえない．このため，本システムでは，キャッシュ機構を備えた．

SQL コマンドはデフォルトではキャッシュ対象とはならない．sql ファイルに「Cache=1」という宣言が置かれたときのみがキャッシュ対象となる．「Cache=1」の場合，初回の実行では，SQL コマンド（クエリ）の実行が行われ，RDB システムから戻されたデータセット（リザルト）を WWW クライアントに送出すると共に，クエリ/リザルトの対を，サーバーのメモリにも格納する．次回のアクセスでは，SQL コマンドの照合で，キャッシュにあることが分かるため，このデータセットが WWW クライアントに戻される．キャッシュ対象は，直接，テーブルからレコード抽出する選択クエリ（From に続くパラメータがテーブル名）に限定する．

SQL コマンドの解析から，それぞれのキャッシュされたデータセットがリレーショナルデータベース上のどのテーブルから取り出されたものかは明らかとなる．このテーブルに変化があった，すなわち，Insert, Delete, Update などの更新クエリが実行された時に，キャッシュデータを無効として，廃棄している．したがって，その後の最初のアクセスで，SQL コマンドの再実行が行われ，新たなデータセットがキャッシュされることとなる．具体的には「ReNew=tablename」というステートメントがあったとき，テーブル「tablename」から抽出されたキャッシュ情報を廃棄する．

WWW-RDB インタフェースに含まれるキャッシュ管理プログラムは，キャッシュされている各データセットについて，最終アクセス時刻を記録している．キャッシュするとき，データセット格納エリアが足りない場合には，最終アクセス時刻が最も古い順から，必要な空きエリアが得られるまで廃棄する．

3. WWW-RDB インタフェース

前章では，選択クエリを例として，WWW と RDB とのインタフェースを述べた．データベースのインタフェースとしては，条件を満たすレコードの抽出だけでなく，新規レコードを挿入（追加）したり，既存レコードの内容を修正する機能が必要である．

選択クエリについては，すでに述べたため，省略し，本章では，新規レコードの追加と既存レコードの更新を実行するアクションクエリについて説明する．

図 4 レコード挿入フォームの表示例
Fig. 4 An example of display screen to insert record.

3.1 レコードの追加

RDB のテーブル「図書台帳」に新規レコードを追加（挿入）するための入力を受付ける HTML ファイル「insert.htm」を以下に，画面の表示例を図 4 に示す．

```
<Form Method=POST Action=/sql/insert.sql>
    図書名<Input Type=text Name=Title><p>
    著者名<Input Type=text Name=Author><p>
    発行所<Input Type=text Name=Pub><p>
    <Input Type=submit Value=レコード追加>
 </Form>
```

これと対をなす insert.sql ファイルを以下に示す．

```
Database=WWW
SQL=Insert into 図書台帳（図書名，著者名，発行
        所）
        Values（'Title','Author','Pub'）
Response=<a href=/insert.htm>
        レコード追加フォームへ</a><p>
```

これは SQL コマンドとして，Select コマンドの代わりに Insert into コマンドが使われるということを除いて，選択クエリと大差がない．アクションクエリを実行後，Response で宣言したデータ（HTML 形式）が WWW サーバーから WWW ブラウザに戻される．

3.2 レコードの更新

図書番号を指定して，図書台帳のレコードを更新する例を取り上げる．図書番号の入力フォームは，これまでの例から自明のため，説明を省く．データ修正 sql ファイル「modify.sql」を以下に示す．

```
Database=WWW
Format=Update
SQL=Select * From 図書台帳 Where 図書番号='%1'
Response=<H2>レコード更新</H2><p>
 図書番号: TID<p>
 <Form Method=POST Action=/sql
                        /update.sql?%1>
  図書名
   <Input Type=text Name=Title Value=:
                        図書名><p>
```

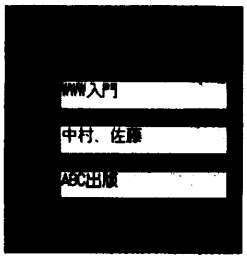

図5 レコード更新フォームの表示例
Fig. 5 An example of display screen to update record.

```
著者名
 <Input Type=text Name=Auth Value=:
                              著者名><p>
発行所
  <Input Type=text Name=Pub Value=:
                                  発行所><p>
  <Input Type=submit Value=レコード更新>
</Form>
```

まず，引数として与えられた図書番号（たとえば，1234）に対するレコードの抽出を行う．

Response データに含まれる入力ボックスの属性 Value に設定している「:図書名」,「:著者名」,「:発行所」は，それぞれのフィールドの値に置き換え，WWW ブラウザに戻される．その結果，これらが初期値となって，図5のように表示される．

入力ボックスの文字列を変更し,「レコード更新」ボタンをクリックすると，以下に示す update.sql ファイルがコールされる．

```
Database=WWW
SQL=Update 図書台帳 Set 図書名='Title',
   著者名='Auth', 発行所='Pub'
   Where 図書番号='%1'
Response=レコードを更新しました
```

入力したデータと SQL コマンドとのバインディングが行われる．すなわち，引数の値 1234 が %1 に代入される．Title, Auth, Pub にも代入が行われ，その結果得られた SQL コマンドが RDB 管理システムへ送られ，レコードの更新が実行される．

4. RDB ビュア

WWW ブラウザからデータベースにアクセスするときのユーザー・インタフェースは，リクエスト画面とレスポンス画面が分離，独立している．リクエスト/レスポンスが1～2回で終わる場合，このようなインタフェースで支障はないが，何度もリクエスト/レスポンスのやり取りをしないと，1つの処理が終わらない場合，画面が何度も切り替わるのは，操作性が悪く，ユーザーフレンドリーなインタフェースとはいえない．

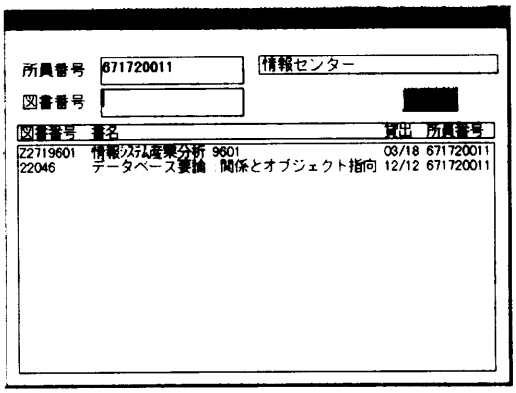

図6 RDB ビュア表示画面例
Fig. 6 An example of RDB viewer display screen.

WWW-RDB 連携システムの適用事例として，図書管理システムを開発した．図書の新着案内，蔵書検索などは，リクエスト/レスポンスが1～2回で終わるため，WWW ブラウザからサービスが受けられるようにしている．

一方，後述するように，図書の貸出処理では，1冊の貸出でも，5～6回のリクエスト/レスポンスが発生する．その都度，画面が切り替わっていては，使い勝手が悪い．このため，図書室の図書貸出管理用パソコンには，ユーザーフレンドリーなインタフェースを実現するために，別途開発した RDB ビュアをインストールして，この上で図書貸出管理サービスを実現している．

図書貸出処理画面を図6に示している．最初は，入力ボックス，リストボックスには何も表示されていない．カレット（縦棒）が所員番号入力ボックスの左端でブリンクしている．

所員番号をキー入力する（あるいは，所員番号のバーコードをバーコードリーダーで読み込ませる）と所属，氏名を問い合わせるリクエストをサーバーに送出する．サーバーからレスポンスが戻るとその右のボックスに表示する（図では，情報センターとなっているが，実際のシステムでは，所属と氏名を表示する）．次にカレットは図書番号の入力ボックスに移り，図書番号の入力を待つ．

図書（書籍，雑誌）に貼られたバーコードを読み込むと，今回の図書を先頭に，この人が借りている図書の一覧を表示する．その後，カレットは再び，図書番号の入力ボックスに移り，図書番号の入力を待つ．

このような GUI は WWW ブラウザでは実現できない．また，RS-232C を通して，バーコード入力を受

表4 コントロール
Table 4 Control.

タイプ	用途
static	文字列の表示に用いる.
edit	文字列の入力ボックス. データ表示にも使う.
listbox	複数レコードの表示に使用する.
combobox	入力ボックスとリストボックスの組合せ
button	ユーザからの指示を受けるためのボタン
variable	データの保管場所

け付ける機能もない. RDB ビュアは, このようなインタフェースを実現するために, 開発したものである.

RDB ビュアは, 起動したとき, WWW サーバーから定義ファイルをダウンロードして, この内容に従って動作する.

RDB ビュアと WWW サーバー間の通信プロトコルは当然 HTTP であるが, 定義ファイルの記述形式は, HTML ではない. RDB ビュアの作りやすさを考慮して, Windows のプロファイル (通常, 拡張子 INI をつけたファイル) に準じたものとした.

コントロールの種類は表4に示す6つである.

入力ボックスに文字列をキー入力して「Enter」キーを押したとか, マウスでボタンをクリックしたとか, バーコードを読み込ませたといった, イベントの発生で, プログラムを実行する, いわゆる「イベント駆動型」のプログラム実行機能を有している.

プログラムで使える関数を表5に示している.

図書番号のバーコードを読み込んだとき実行するプログラムを図7に示す. 処理概要は以下のとおりである.

Step 1: 書籍かどうか調べる.
Step 2: 書籍であれば, Step 5 に進む.
Step 3: 雑誌かどうか調べる
Step 4: 雑誌でなければ, エラー表示後 Step 10 へ
Step 5: すでに貸出中 (返却処理もれ) でないか調べる
Step 6: 貸出中でなければ Step 8 へ
Step 7: 以前の貸出に関して返却処理を行う
Step 8: 今回の貸出を記録する
Step 9: 今回の貸出を先頭に貸出図書リストを表示
Step 10: 次の図書番号の入力を待つ

Step 1, 3, 5, 7, 8, 9 では, クラインアントとサーバー間でリクエスト/レスポンスの交信が行われる.

各コントロールの属性は, Windows のプロファイルと類似した記法で記述する. KasidasiList の属性を以下に示す.

```
[KasidasiList]
Font=18 4 "MS ゴシック"
Class=listbox
Window=15,145,600,305
Format=%-9s %-40s %-5s %-9s
```

Font のパラメータは順に, ポイント, 太さ, フォント名である. Window のパラメータは順に, このリストボックスの X 座標, Y 座標, 幅, 高さである. また, Format は C 言語にならったもので, 各フィールドの表示幅を与える.

プログラム記述方式としたため, RDB ビュアは, 適用可能な業務が広く, コンパクトな (HTTP サポートを含め, 700 行弱の C 言語プログラム) ものとなった. また, 図書貸出管理システムの記述プログラムは約 400 行である. なお, 今回開発した RDB ビュアは Windows 95 および Windows NT Workstation を搭載したパソコン上で稼動する.

MS-Access SQL は, 基本的には ANSI-89 レベル1の仕様に準拠しているが, 若干の差異がある. MS-Access SQL では, 強力な式がサポートされている. Access Basic 関数やユーザー定義関数はすべて MS-Access SQL ステートメントで使うことができる.

図7の例では, Step 7, Step 8 における Now () は現在に日時を返す Access Basic の関数である. また, Step 3 の MagNo(tid), Step 8 の Syozoku(uid) はユーザー定義関数である. 前者は, 図書番号から雑誌コードを, 後者は, 所員番号から所属を求める関数である.

5. システム評価

5.1 WWW-RDB 連携機能の評価

提案システムでは, 広く普及している WWW ブラウザから, RDB 管理システムにアクセスして, 抽出条件を満たすレコードを表示したり, 新規レコードを追加したり, あるいは, 既存レコードの内容を更新できる機能を実現した.

連携システムでの SQL コマンドの与え方は, SQL コマンドを HTML ファイルに埋め込む方法[5),8),9)]と, 別ファイルで定義する方法に大別される[14)~16)].

Alexandria Digital Library[5)]では, クエリ/表示 (レスポンス) をフォームのデータとして, ユーザーに開放している. このような積極的な目的がある場合は別として, 一般には, 裏でどのような SQL コマンドが動いているか WWW ブラウザのユーザーに見せる必要はないので, 提案システムでは別ファイル方式を採用した.

ユーザー入力は HTML のフォーム機能で記述するため, アプリケーションの開発に HTML 以外の特別

表5 記述関数
Table 5 Programming Function.

関数名	引数	機能概要
SQL	FieldName: 列名表示先 Record: レコード表示先 SQL: SQL コマンド	WWW サーバーに SQL コマンドを送り，応答結果を指定先に表示
OpenForm	FormName: オープンするフォーム名	フォームをオープンする．
JumpOn Null	Control: 判定データ JumpNo: ジャンプ先 Message: 表示メッセージ	指定コントロールが空のとき，指定先へジャンプ
JumpOn NotNull	Control: 判定データ JumpNo: ジャンプ先 Message: 表示メッセージ	指定コントロールが空でなければ，指定先へジャンプ
Jump	JumpNo: ジャンプ先	無条件ジャンプ
Return	なし	実行終了
Select	ComboBoxName: コンボボックス名 SelectNo: 選択番号	リストボックスから選んで，入力ボックスに設定
SetText	SourceControl: コピー元 DestControl: コピー先	データ (文字列) をコピー
Disable	Edit: キー入力を禁止する入力ボックス名	バーコード入力のみ受け付ける．
Enable	Edit: キー入力を許可する入力ボックス名	キー入力を許可する．
GoTo	Control: コントロール名	フォーカスを移動

```
1 SQL(null, Work, "Select 書籍番号 From 書籍台帳 Where 書籍番号='TID'")
2 JumpOnNotNull(Work, 5, "")
3 SQL(null, Work, "Select 雑誌番号 From 雑誌名 Where 雑誌番号=MagNo('TID')")
4 JumpOnNull(Work, 10, "この図書番号<TID>は登録されていません")
5 SQL(null, Work, "Select 図書番号 From Q_貸出中 Where 図書番号='TID'")
6 JumpOnNull(Work, 8, "")
7 SQL(null, null, "Update Q_貸出中 Set 返却日時=Now() Where 図書番号='TID'")
8 SQL(null, null, "Insert into 図書貸出記録(図書番号,所員番号,貸出日時,所属,貸出IPA)
  Values('TID','UID',Now(),Syozoku('UID'),'_IPA')
9 SQL(FldName, KasidasiList, "Select 図書番号,書名,貸出,所員番号 From Q_貸出図書
  Where 所員番号='UID' And 返却日時 Is Null Order By 貸出日時 Desc")
10 Goto(TID)
```

図7 プログラム例
Fig. 7 An example of program.

な知識を必要としない．クエリを定義する sql ファイルの設計は，RDB 管理システムの SQL に従う．

今回開発したシステムでは，エンドユーザーコンピューティングを狙いとしたため，RDB 管理システムとしては，オフィス業務分野で広く使用されている MS-Access を使用した．図書貸出管理システムの開発を通じて，例外処理にも柔軟に対応できることが明らかとなった．

MS-Access は元々サーバー用に作られたものではないため，注意を要する点もある．たとえば，エラー発生時にダイアログボックスを表示して，指示待ちで停止してしまうケースがある．アクションクエリを実行する前に，選択クエリで重複がないかどうかチェックするなど，実行時エラーが発生しないように対策した．

また，WWW ブラウザでは実現できない，よりユーザーフレンドリーなインタフェースに対処するために，RDB ビュアを開発した．

これまで，MS-Access のフォーム機能を使用したスタンドアロンの図書貸出管理システムを使用していたが，RDB ビュアでこれと同等の GUI を実現することができた．

5.2 性能実測結果

システムの性能は，ハードウェア構成および OS に依存するところが大きい．性能実測に用いたサーバーおよびクライアントシステムの主な仕様は次のとおりである．

[サーバー]
プロセッサ: Pentium☆90 MHz，主メモリ: 48 MB，OS: Windows NT Server 3.5，ファイルシステム: NTFS, 4 GB RAID

[クライアント]
プロセッサ: Pentium 100 MHz，主メモリ: 24 MB，OS: Windows NT Workstation 3.5，ファイルシステム: FAT, 540 MB，WWW ブラウザ: Netscape☆☆1.1

WWW-RDB 連携に限らず，システム全体の性能を測定した．当然，ネットワークの負荷が高いときには，レスポンスは悪くなる．また，同一サイトのクライアントでなく，デジタル回線で LAN 間接続した遠隔地クライアントからのアクセスの方がそれ相応に時間がかかる．

同一サイトで，ネットワークがそれほど込んでいない状況での 3〜5 回の応答時間の平均値を表 6 に示す．

☆ Pentium は米国 Intel Corp. の商標である．
☆☆ Netscape は米国 Netscape Communications Corporation の商標である．

表6 WWW ブラウザ応答時間
Table 6 Response time for WWW browser.

No	種類	サイズ	応答時間	実効伝送速度
1	テキスト	11.2 KB	0.24 秒*	46.7 KB/秒
2	HTML	23.5 KB	1.25 秒	18.8 KB/秒
3	CGI/HTML	21.3 KB	0.96 秒	22.2 KB/秒
4	GIF	5.4 KB	1.11 秒	4.9 KB/秒
5	SQL/HTML	28.3 KB	2.74 秒	10.3 KB/秒
6	Cache/HTML	28.3 KB	1.14 秒	24.8 KB/秒

* RDB ビュアでは 0.22 秒

ここで，応答時間は，WWW サーバーソフトが着信を検出してから，クライアントへのレスポンス送信を終え，コネクションをクローズするまでの時間であり，Windows NT システムのタイマー関数（単位：ミリ秒）を用いて計測した．

当然，テキストファイルに対する応答がもっとも速い．サーバーでは各種のログをとっており，これらのオーバヘッドも平均応答時間 0.24 秒に含まれている．

RDB ビュアに対する応答の方が若干速い．0.14 秒とか 0.17 秒というケースもあった．

HTML ファイルの場合，WWW ブラウザで表示データを作成するのに時間がかかるため，応答時間はテキストファイルのおよそ 2 倍となっている．

No.3 は CGI 経由で文書ファイルのフルテキストサーチを行い，該当 200 件のファイルのタイトルを HTML ファイル形式でメニューとして戻したときの応答時間である．サーバーとしては No.2 の単に HTML ファイルを戻すより負荷がかかるが，HTML ファイルは No.2 よりもシンプルなため，WWW ブラウザの処理時間が短くなり，総合的には No.3 の方が応答が速くなったものと見られる．

CGI の部分にかかるオーバヘッドを独立して計測していないが，No.2 と No.3 の比較から，CGI 自体のオーバヘッドはきわめて小さいことが推定される．

GIF の場合，テキストの 10 倍，HTML の 4,5 倍の応答時間になっている．これは，WWW ブラウザ側で圧縮ファイルの復元に時間がかかるためである．

WWW-RDB 連携に直接関わるのは No.5 と No.6 である．No.5 は簡単な選択クエリを実行した抽出されたレコードを HTML の表形式に変換して，WWW ブラウザに戻したものである．No.6 は No.5 の抽出結果をキャッシュして，このキャッシュ結果を WWW ブラウザに戻したものである．両者の時間差 1.6 秒がキャッシュの効果である．表形式のため，WWW ブラウザの負担は No.3 より重いと思われるが，No.6 の方がサイズあたりの応答時間が短いのは，CGI のオーバヘッドもなく，また，データは主記憶にキャッシュされており，データの転送にファイル I/O がないためである．

表7 RDB ビュア応答時間
Table 7 Response time for RDB viewer.

No	SQL コマンド	応答時間	図7との対応
1	所属，氏名の取出し	1.16 秒	-
2	書籍台帳チェック	1.17 秒	Step 1
3	雑誌名チェック	1.17 秒	Step 3
4	貸出中かチェック	1.22 秒	Step 5
5	貸出レコード登録	1.51 秒	Step 8
6	貸出リスト取出し	1.34 秒	Step 9

表7 に RDB ビュア応答時間の実測結果を示す．この応答時間は，同一サイトのクライアントと遠隔地のクライアントとの間で差異は見られなかった．これは，転送データが小さく，応答時間の内訳のほとんどが SQL コマンドの実行時間で占められるためである．

6. おわりに

中規模データベースシステムを対象に，WWW-RDB 連携システムを提案し，評価を行った．

WWW ブラウザから，レコードの追加，削除，更新など，データベースの操作が行えることを確認した．

また，性能実測を行い，実用上十分な性能が得られることが明らかとなった．

WWW ブラウザからのアクセスは，フォームにデータをセットして，実行ボタンをクリックすると，設定データが WWW サーバーに送られて，SQL コマンドが実行されて，WWW ブラウザに表示する次の画面を WWW サーバーから WWW ブラウザに戻すというシーケンスを繰り返す．もっと滑らかな GUI を実現するために，RDB ビュアを併せて開発した．

WWW ブラウザは急速な進歩を続けている．Java のようなプログラマブル WWW ブラウザ用ソフトの開発環境が充実して，広く使われるようになる日も近いと思われる[12),13)]．今回は，定型業務用に，よりユーザーフレンドリーなインタフェースを実現するために，独自の RDB ビュアを開発した．今後，このようなユーザーフレンドリー・インタフェースをプログラマブル WWW ブラウザで実現したい．

参考文献

1) WWW と DBMS の連携が開く世界，WWW—データベース連携システム構築法，pp.2-7, 日経 BP 社 (1996.3).
2) 益岡竜介，木庭袋圭祐：解説 World-Wide Web (WWW)，情報処理，Vol.36, No.12, pp.1155-1165 (1995).
3) WWW を使ってオンラインで航空券を予約する

(ユーザー事例：米 Travel-R-Us 社)，WWW—データベース連携システム構築法，pp.20–25，日経 BP 社 (1996.3).
4) 部門間の情報交換を実現する社内情報検索システム (ユーザー事例：三菱商事)，WWW—データベース連携システム構築法，pp.32–35，日経 BP 社 (1996.3).
5) Andresen, D., et al.: The WWW Prototype of the Alexandria Digital Library, *Proc. International Symposium on Digital Libraries 1995*, pp.17–21 (1995).
6) Oracle WebServer，WWW—データベース連携システム構築法，pp.192–196，日経 BP 社 (1996.3).
7) 佐々木雅志：米 Sybase 社 web.sql，WWW—データベース連携システム構築法，pp.197–199，日経 BP 社 (1996.3).
8) 飯島秀幸：Illustra Web DataBlade，WWW—データベース連携システム構築法，pp.183–187，日経 BP 社 (1996.3).
9) 島田尚一：仏 O_2 Technology 社 O_2 Web，WWW—データベース連携システム構築法，pp.188–191，日経 BP 社 (1996.3).
10) 元田敏浩，徳丸浩二：WWW とデータベースサービスとの連携方式の検証，信学技報，KBSE95-7 (1995-05).
11) 遠山，軽部，指田ほか：レイアウト式 TFE の拡張，信学技報，DE95-36 (1995).
12) 山本雅史：Javaってどんなもの，日経パソコン，3 月 11 日号，pp.216–220 (1996).
13) 足立　太：最新開発環境レポート Borland C++ Ver. 5.0，*C MAGAZINE*，pp.138–139 (1996.4).
14) 杉山仁志：富士通 InfoProvider，WWW—データベース連携システム構築法，pp.234–237，日経 BP 社 (1996.3).
15) 竹内　宏：WebCARNIVAL，WWW—データベース連携システム構築法，pp.206–209，日経 BP 社 (1996.3).
16) WWW の覇権を狙う Microsoft のインターネット戦略，WWW—データベース連携システム構築法，pp.166–171，日経 BP 社 (1996.3).

(平成 8 年 6 月 24 日受付)
(平成 8 年 11 月 7 日採録)

畑田　稔（正会員）

1942 年生．1967 年姫路工業大学電気工学科卒業．1972 年京都大学大学院工学研究科博士課程修了．同年日立製作所入社，現在，システム開発研究所に勤務．制御系の安定問題，大規模システムの解析と評価などの研究を経て，技術情報サービスシステムの研究開発に従事．1971 年電気学会論文賞受賞．工学博士．システム制御情報学会会員．

遠藤　裕英（正会員）

1941 年生．1965 年京都大学工学部電子工学科卒業．同年日立製作所入社．中央研究所主任研究員，マイクロエレクトロニクス機器開発研究所部長を経て，現在システム開発研究所情報センタ長．この間，制御用計算機，パターン認識装置，ワープロ，技術情報サービス等の研究開発に従事．