UNIX実験 II
データベース

提出日：2006/01/09 消印有効

中村そば　ずみ！

店員：　035737F：中村匡
035709A：宇座瞬
035727J：平良勇志
035740F：根保光秀
035743A：比嘉雅樹
035756B：宮國渡
035758J：村山正嗣

1　実験の目的

　この実験では、サーバー上で動作するデータベース閲覧システムを作成し、データベース の構築方法やデータベースにアクセスするためのサーバーの設定方法 サーバー上で動作するプログラミング手法に付いて学ぶこと目的とする。

2　実験の背景

　データベースには、MySQL や PostgresSQL などフリーで使えるものや Oracle のような商用データベースなどがあり、Web サービスや業務などに使用されている。 本実験では商用として実際に多く使われている Oracle を使用し、データベースを構築しデータベース閲覧システムを開発する。

Oracle データベースの最新バージョンである Oracle10g が発表された。上述のように Oracle は商用データベースなのでトライアル版をインストールして使用することにした。

サーバー上で動作するプログミング言語は php や perl などがある、 本実験では html やデータベースとの相性が良い php を使用して閲覧システムを開発する。

3　Level1：データベース設計のための情報収集

　インタビュー先：ヤマダ電機 テックランド豊見城店
インタビュー内容
質問１：ヤマダ電機ではどのようなデータベース（以下 DB）があるのでしょうか。
解答　：商品管理のための在庫の DB、売り上げを立てた後の管理のための DB、お金の管理の DB です。

質問２：在庫の DB についてと、利点や欠点などを教えてください。
解答　：在庫の DB は閉店後から開店前に更新されます。その日に入荷予定の商品は翌日にプラスされます。この DB は商品の型番をうつと全国のお店の在庫が調べられるようになっています。
利点としては全国の在庫がある程度わかるので、足りない分は他のお店から移してもらったり、○○店なら在庫または展示品がありますと案内できる点です。

欠点としては、物流センター（メーカーから来た商品をそれぞれの店舗に振り分ける場所）を通さず、直接商品をお店に持ってくる時（在庫のないものを急いで配送してもらいたいといった時などがあるそうです）はそのデータが自動的には反映されない点、朝のデータのままなので数が少ないものは DB をみただけでは有無の判断ができないので、直接在庫の確認に行かないといけない点です。また、DB にある数自体が信用できないときもあります。（色違いの販売ミス等。）

質問３：売り上げ管理の DB についてと、利点や欠点を教えてください。
解答　：この DB では、実際のレシートと同じ情報が売り上げを立てた順番に蓄積されます。イメージとしてはレシートが何枚も縦につながっていると考えてもらえればいいです。レシートにはレシートナンバー、日時、誰が売ったのか、ポイント会員ナンバー、商品の型番と金額、会計とポイント数、そのレシートを呼び出すための JAM コードが書かれています。
利点としては、レシートがあれば返品や保証の変更（当日のみ）がすぐにできる点、レシートをなくしてもポイントカードといつ頃何を買ったのかを覚えていればそのデータを呼び出せる点です。また、ポイントカードも携帯電話でのインターネットへの登録であれば、電話番号で会員ナンバーを呼び出せるので一応電話番号さえ覚えていればデータの呼び出しは可能な点です。
欠点としては誰が買っていったのかがわからないので、会員ナンバーがわからない時、例えばカードをなくしてしまったりやカードを使用または発行せずに会計をしたときには探すのに時間がかかる点（ほぼ不可能）です。

質問４：お金管理の DB についてと利点や欠点を教えてください。
解答　：この DB では、どこのレジにいくら入っているかわかるようになっています。
利点はお金の管理がしっかりできる点です。
欠点としては過不足が出たときは、そのレジの売り上げの DB を見ながら怪しいところを１つ１つ調べるのでなかなか大変なところです。

4　Level2：収集した情報の記述

使用したツール：jude
まず、以下にユースケース図を示す。

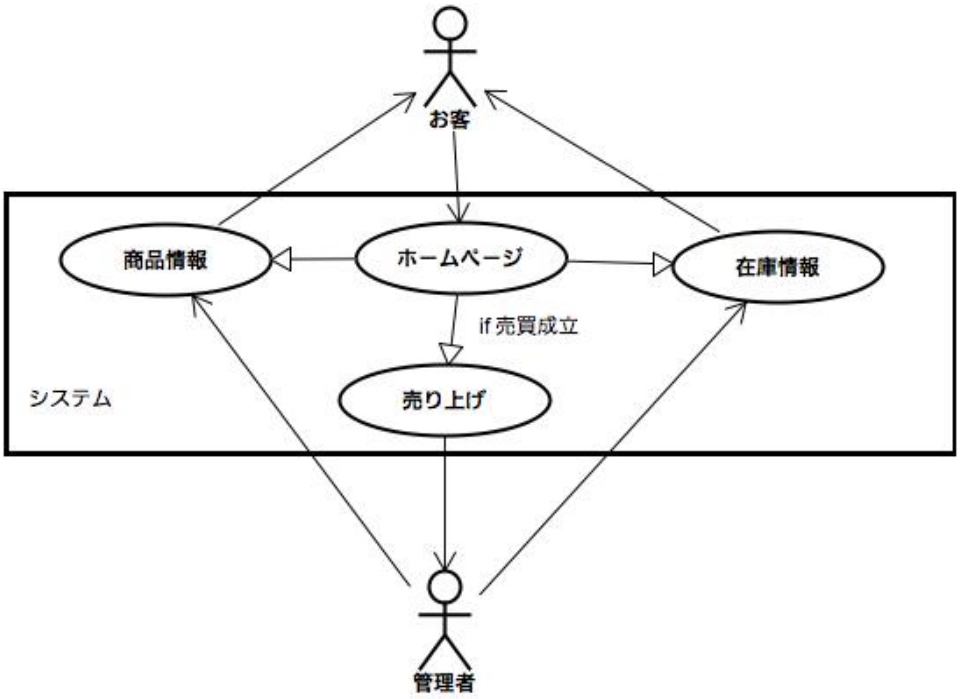


図 1: ユースケース図

ユーザーはホームページにアクセスし欲しい商品を検索する。ホームページは商 品情報と在庫情報のデータベースにアクセスし、取り出したデータをホームペー ジ内で表示する。このようにユーザーは商品の情報を得ることができる。もし売 買が成立した場合は在庫情報にアクセスし売り上げをけいさんして管理者に渡す。 管理者は商品情報や在庫情報を追加、変更する。次にこのシステムのシーケンス 図を示す。

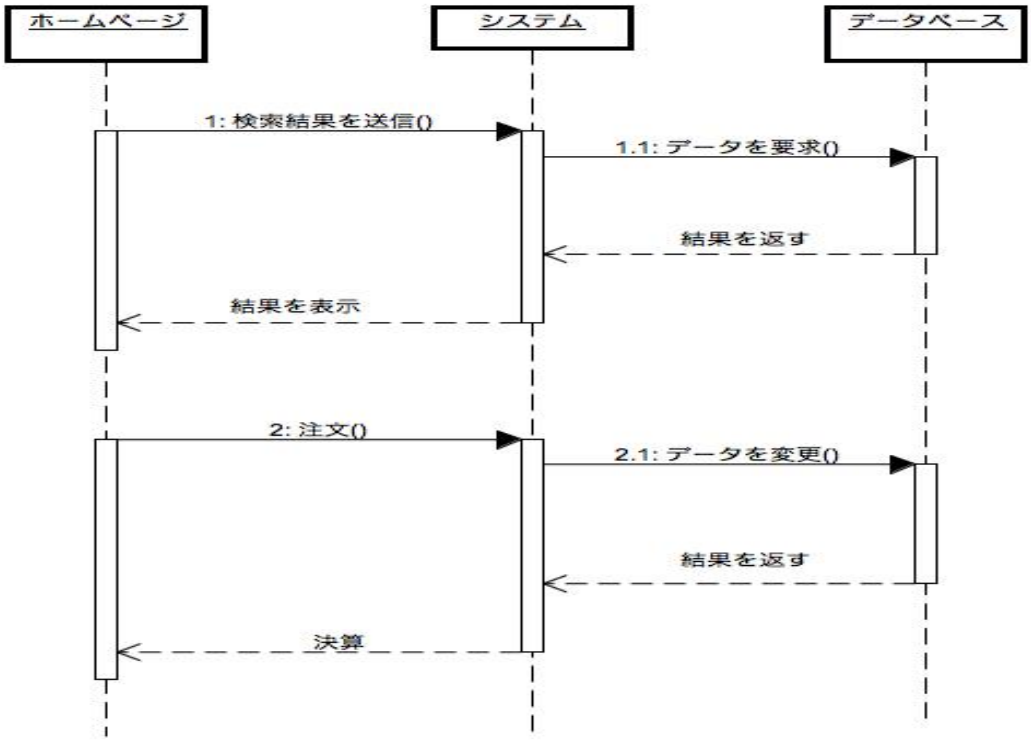


図 2: シーケンス図

ホームページからの情報をシステムが受け取り、データベースにアクセスする。 データベースには商品情報や在庫情報が格納されているので要求されたデータを システムに返す。システムは得られたデータを元にホームページを更新する。ユー ザーはそのホームページを見て注文の処理を行った場合、システムがその要求を 受け取りデータベースにアクセスし、必要な処理を行う。次にデータベースのク ラス図を示す。

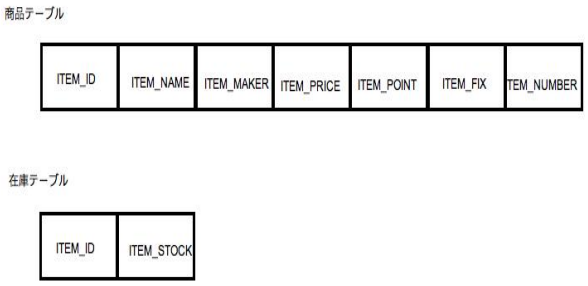


図 3: クラス図

商品テーブルには商品の通し番号、商品名、製造者、値段、ポイント、定

価、型 番がそれぞれ記録されている。山田電機ではカテゴリ (家電やパソコン,CD/DVD) ごとに分かれてデータベースを作っているとのことなので実際にはカテゴリごと に同じテーブルをいくつか作る。

在庫テーブルには通し番号と在庫の数が記録されている。この他にもユーザーテー ブルなどを作る必要があるが、それは後のレベルで詳しく説明する。

5 Level3 データベースの構築

概念モデルに従ってテーブルを作成した。テーブルの作り方は

表 1: テーブルを作る SQL 文

```
create table item_deji(
ITEM_ID NUMBER(10),
ITEM_NAME VARCHAR2(100),
ITEM_MAKER VARCHAR2(40),
ITEM_PRICE NUMBER(14),
ITEM_POINT NUMBER(7),
ITEM_FIX VARCHAR2(40),
ITEM_NUMBER VARCHAR2(40));
–以下カテゴリごとに同じテーブルの追加–
–省略–
create table stock(
ITEM_ID NUMBER(10),
ITEM_STOCK NUMBER(5));
```

テーブル名は

表 2: テーブル名

カテゴリ	テーブル名
AV・デジカメ館	item_deji
家電・健康館	item_kaden
パソコン館	item_pc
文具・オフィス館	item_ofice
CD・DVD 館	item_cdvd
在庫	stock

次に、テーブルの要素を下に示す。

表 3: 商品テーブル

説明	カラム名	大きさ (byte)
商品番号	ITEM_ID	数字 10 桁
商品名	ITEM_NAME	文字 50 字 (100byte)
製造者（歌手）	ITEM_MAKER	文字 20 字 (40byte)
値段	ITEM_PRICE	数字 14 桁
中村ポイント	ITEM_POINT	数字 7 桁
定価	ITEM_FIX	文字 20 字 (40byte)
型番	ITEM_NUMBER	文字 20 字 (40byte)

表 4: 在庫テーブル

説明	カラム名	大きさ (byte)
商品番号	ITEM_ID	数字 10 桁
在庫数	ITEM_STOCK	数字 5 桁

6 Level4：テスト・データベースの構築

ヤマダ電機の通信販売ページから、データを抽出する事にした。入力データは、ヤマダ電機の商品 ID、商品名、会社名、型番（CD は JAM コード）、値段、ポイント、定価を抽出し、全てのデータが揃っている商品のみ、データベースに登録する。データの抽出は、AV デジカメ館、家電健康館、パソコン館、文具オフィス館、CDDVD 館から抽出する。それ以外のカテゴリは、条件に合ったデータを得る期待値が低いため、除外した。CDDVD 館においても、DVD に関しては、期待値が低かったので同じく除外した。

データの抽出においては、文字列の扱いに優れている java を用いて行った。おおまかな流れとしては、あらかじめ作成した index ファイルを参照し、それぞれのカテゴリへ wget で読み込み、それを解析し、商品リストを wget で読み込み、解析する。無効なデータの抽出も多く予想されるので、目標取得件数を 120% の 12 万件と定めた。実際に取得したデータは 138,540 件で、うち有効なデータ 107,049 件の入力を試みた。しかし、商品名が長すぎるといった、我々の定義したテーブルに格納不可能なデータも含まれているので、実際はこれより少なくなる事が予想される。

6.1 データの取得ルーチン

目標とするデータを取得するために、いくつかのステップに分けて説明する。使用したプログラムは、web_get.java、Function.java、yamada_data.java である。
まず、YAMADA 電機のインデックスページから、使用するカテゴリのみを

ファイルに書き込む。これについては手動で作成した。ファイルには以下のようにして書き込んだ。

```
AV・デジカメ館
http://www.yamada-denkiweb.com/contents.php/category/10/
家電・健康館
http://www.yamada-denkiweb.com/contents.php/category/11/
パソコン館
http://www.yamada-denkiweb.com/contents.php/category/12/—
```

次に、先に作成したファイルを参照しながらリンクをたどる。そしてカテゴリーの解析を、html ファイル中に`<!-- カテゴリーリスト Start -->`といったフレーズが出ると開始する。プログラム中では、web_get.filter1(String root,String uri,String to); で行っている。root はダウンロードしたページの URL で、uri は、ダウンロードしたファイルの保存先、to は書き出し先を指している。また、基本的に、uri は temp ファイルとして保存し、to は、作業ディレクトリ/カテゴリ名.list、として保存される。データの保存形式を以下に記す。

```
##roothttp://www.yamada-denkiweb.com/contents.php/category/10/
##category:デジタルカメラ
５００万画素デジカメ
/item/list.php/category/10/315/16/
６００万画素デジカメ
/item/list.php/category/10/315/17/
７００万画素デジカメ
/item/list.php/category/10/315/18/
８００万画素デジカメ
/item/list.php/category/10/315/19/
##category:衛星放送機器・アンテナ
ＢＳアンテナ
/item/list.php/category/10/81/10/
ＢＳチューナー
/item/list.php/category/10/81/11/
ＢＳデジタルチュー
/item/list.php/category/10/81/12/
```

次に、作業ディレクトリ/カテゴリ名.list を参照しさらに下ると、商品リストのページに出る。商品リストのページには３種類のページが存在するが、そ

れぞれを自動で判別する機能は装備していない。よって、手動での切り替えでプログラムを実行した。web_get.filter2(String base,String category,String from,String to,boolean bool); を再帰的に実行してリンクを辿っていった。base はベースアドレスで、ホームページの URL が書かれている。category にはカテゴリ名、from はファイル読み込み先、to は書き出し先、bool はリンクを辿るかどうかである。一般的に、to は作業ディレクトリ／カテゴリ名／サブカテゴリ名 xx.list　として保存される。このページでは、<!-- 商品情報 START -->がキーワードとなっている。データの保存形式例を以下に記す。

```
##category:デジタルカメラ
会社名　：キャノン
商品名　：Canon500 万画素デジタルカメラ IXYDIGITAL60
型番　　：IXYDIGITAL60
商品番号：3150692019
特価　　：38,800 円
ポイント：9,700
定価　　：オープンプライス
会社名　：カシオ
商品名　：CASIOEXILIMZOOM500 万画素 EX-Z500 シルバー
型番　　：EXZ500S
商品番号：3150676019
特価　　：35,200 円
ポイント：7,744
定価　　：オープンプライス
##category:キャンプ・スキー用品
会社名　：
商品名　：スキー
型番　　：RD メンズタートル
商品番号：9972426016
特価　　：990 円
ポイント：99
定価　　：オープンプライス
```

また、取得したデータを list_analysis.java を用いて、sql へ入力用に変換し、入力した。上のデータを INSERT INTO item_deji VALUES(1351078014,'MS-PD','SONY',19800,2376,' オープンプライス','MSXM2GS'); といった形に変換した。

在庫テーブルのデータを追加する際、在庫テーブルの ITEM_ID は上で追加したデー タに対応していなければならないため、商品テーブルから ITEM_ID を読み込む必 要がある。在庫の数はとりあえず全て 100 個にした。データを

追加する PHP のソースは、下記の場所参照。

http://www.ie.u-ryukyu.ac.jp/~j03040/nakamurasoba2/level4/zaiko.php.txt
また、データを追加した後下記の SQL 文を実行してデータがちゃんと格納さ
れているか 確認する。

```
SQL> select count(ITEM_ID) from stock;

COUNT(ITEM_ID)
--------------
104964
SQL>
```

上記のようにちゃんと追加されていることが確認できた。 データは全部で 10 万 4 千件以上追加されていて 1 レコード約 220byte なので 実験条件の”最低でも 1 万件以上、 大きさにして、10MB 以上 1GB 以下の大きさのテスト・データベースを生成する必 要がある。”をクリアしている。

7 Level5：データベース閲覧システムの作成

7.1 アルゴリズム

プログラムは大きく分けて、検索条件を入力する入力部と 入力された文字列を元に SQL 文を生成する生成部、生成した SQL を実行し 検索結果を表示する出力部の 3 つに分けられる。

QBE(Query By Example) を使った検索システムは次項 Level6 で実現する、php でフォームを実現するには html と組み合わせる必要があるため。 本レベルでは入力された検索条件から SQL 文を生成する生成部を作成する。 入出力は標準入力と標準出力を使用する。

7.2 プログラムと実行結果

以下に、作成したプログラムを示す。Oracle データベースへのアクセスは Oracle 関数 oci8 を利用して実現している。

```
--入力部は省略--
/*SQL 文を生成*/
$sql="select * from ".$table[$data[0]];

for($i=1;$i<count($data);$i++){
 if($data[$i]!=NULL){
  if(!$fr){
   $sql=$sql." where ";
    $fr=TRUE;
  }else{
  $sql=$sql." AND ";
 }
 if($i!=count($data)-1){
  if($i==1||$i==2){
   $data[$i]=ereg_replace('[[:space:]]',"%' OR $colm[$i] LIKE '%",$data[$i]);
   $sql=$sql.$colm[$i]." LIKE '%".$data[$i]."%'";
   }else{
   $sql=$sql.$colm[$i]." = '".$data[$i]."' ";
   }
   }else{
   $sql=$sql.$colm[$i]." <= ".$data[$i];
   }
 }
/*SQL を実行*/
$con=oci_connect("SYSTEM","toybox","");
$stmt=oci_parse($con,$sql);
if(!oci_execute($stmt,OCI_DEFAULT)){
 echo "execute erorr!?n";
 oci_close($con);
 exit;
}
//select 結果を受け取る
$ros=oci_fetch_all($stmt,$res);
--出力部は省略--
```

このプログラムの実行結果を以下に示す。

```
[oracle@pw008 soba]$ php serchdata.php
Database Serch System
カテゴリの番号を選択
1.AV・デジタル 2. 家電 3. パソコン 4. オフィス 5.CD・DVD
>5
商品名を入力
>
キーワードを入力
>サップ
型番を入力
>
値段を入力（以下）
>
select * from item_cdvd where ITEM_MAKER LIKE '%サップ%'
         1 件ヒットしました。

番号:2641625017
メーカー:ボブ・サップ
商品名:SAPPTime!
型番:4988018313977
特価 : 1050 円
158 ポイント
定価 : 1050
[oracle@pw008 soba]$ php serchdata.php
Database Serch System
カテゴリの番号を選択
 1.AV・デジタル 2. 家電 3. パソコン 4. オフィス 5.CD・DVD
>2
商品名を入力
>DVD
キーワードを入力
>SONY 東芝
型番を入力
>
値段を入力（以下）
>1000
select * from item_kaden where ITEM_NAME LIKE '%DVD%' AND ITEM_MAKER
LIKE '%SONY%' OR ITEM_MAKER LIKE '%東芝%' AND ITEM_PRICE <= 1000
         59 件ヒットしました。
番号:701284013
メーカー:東芝
商品名:アルカリ単 4 形 6 本入エコパッケージ
型番:LR03AG6EC
特価 : 525 円
58 ポイント
定価 : オープンプライス
--------------------------------------------------
番号:712549019
メーカー:東芝
商品名:ミニクリ X
型番:KR100V54WXWLA
特価 : 380 円
38 ポイント
定価 : 473
--------------------------------------------------
--略--
```

上記のようにちゃんと動作していることが分かる。問題点は検索結果が多過ぎる と結果の表示が大変なことになってしまう。また、カテゴリを指定し

ないで検索 ができないので次レベル以降改良して行く。

8 Level6：WWW インターフェースを持つデータベース閲覧システムの作成

Level5 で作成したプログラムを変更し html から値を受け取る QBE 方式のシステム を作成する。フォームから値を受け取り SQL 文を生成し、html で出力する。

8.1 プログラム

まず、メインページのプログラムを以下に示す。メインページは QBE 方式を FORM を使用して実現している。FORM に入力された値を SQL 文を生成する関数に渡し、生成された SQL を実行しその結果を受け取って整形 して表示する。

```
require('sql.php');

–form 部は省略–

//FORM から値を受け取って SQL を生成する関数へ渡す処理
if(isset($_POST['st_w'])){
$cate_id = $_POST["cate"];
$item_word = htmlspecialchars(trim($_POST["item_word"]));
$maker_word = htmlspecialchars(trim($_POST["maker_word"]));
$serial_word = htmlspecialchars(trim($_POST["serial_word"]));
$price = htmlspecialchars(trim($_POST["price"]));
$highlows = $_POST["highlows"];

//print "$cate_id,$item_word,$maker_word,$serial_word,$key_word";
$sql=make_sql($cate_id,$highlows,$item_word,$maker_word,$serial_word,$price);
//$sql="select * from item";
echo $sql;//debug
/*
検索結果を受け取って html で表示する処理
*/
list($ros,$res)=select_data($sql);
print "<center>";
print "検索結果 : $ros 件ヒットしました。
";
print "</center>";
print "<center><table cellpadding=15 cellspacing=0
bgcolor#ffffaa width=80%>";
for($i=0;$i<$ros;$i++){
reset($res);
$a=0;
while($val=each($res)){
$data=$val['value'];
$tmp[$a++]=$data[$i];
}
–省略–
}
```

次に関数のプログラムを以下に示す。

```
//SQL 文を生成する関数
function make_sql($cate,$highlows,$item,$maker,$serial,$price){
–省略–
    $sql="select * from ".$table;
   if($item != NULL){
   $fr=TRUE;
   $item=str_replace('  ',' ',$item); //全角スペースを半角スペースに変換
   //スペースで区切られた入力を
   $buf=ereg_replace('[[:space:]]',"%' OR ITEM_NAME LIKE '%",$item);
   $sql = $sql." where "."ITEM_NAME LIKE '%".$buf."%' ";
   }
–省略–
    /*
カテゴリを指定しない場合は全てのテーブルから検索する。
    */
    if($table_all){
    for($i=1;$i<count($category);$i++){
        $tmp=$tmp.str_replace('TB',$category[$i],$sql);
     if($i%(count($category)-1)!=0){
     $tmp=$tmp." UNION  ";
         }
 }
 $sql=$tmp;
    }
    $fr=FALSE;
    return $sql;
}
//SQL を実行する関数
function select_data($sql){
    $con=oci_connect(USER,PASS,DB); //接続
    $stmt=oci_parse($con,$sql);
    if(!oci_execute($stmt,OCI_DEFAULT)){
    echo "oi! select?n";
    return;
    }
    //oci_commit($con);

    $ros=oci_fetch_all($stmt,$res);
    return array($ros,$res);
    oci_close($con); //切断
}
```

Level5 での問題点だった”カテゴリを指定しない”で検索を実装した。カテゴリを 指定しない場合は全てのテーブルにアクセスしその結果を UNION でくっつけて表 示することにした。この方法だと、検索結果が膨大になってしまうという問題が ある。検索結果が多い場合の処理は実現できなかった。以降の Level で改良して いく。なお、プログラムの実行は
`http://pw008.st.ie.u-ryukyu.ac.jp/~oracle/oratest.php`
にてにて行える。

9 Level7：トランザクションの実装

今回はデータベースに追加をするプログラムと、在庫を更新するプログラムを作成した。

9.1 データベースに追加するプログラム

```
define('USER','SYSTEM');
define('PASS','toybox');
define('DB','');

$con=oci_connect(USER,PASS,DB);          //データベースと通信をする。

print "choice tablename 1.AV・デジカメ館 2. 家電・健康館
3. パソコン館 4. 文具・オフィス館
5.CD・DVD 館 =>";     //テーブルを選択
$table = chop(fgets(STDIN));
if($table == 1){
$tablename = item_deji;
}if($table == 2){
$tablename = item_kaden;
}if($table ==3 ){
$tablename = item_pc;
}if($table == 4){
$tablename = item_ofice;
}if($table == 5){
$tablename = item_cdvd;
}else{
exit;
}

print "ID =>";
$ID = chop(fgets(STDIN));    //登録する ID を入力

print "name =>";      //商品名を入力
$name = chop(fgets(STDIN));

print "maker =>";     //メーカーを入力
$maker = chop(fgets(STDIN));

print "price =>";      //値段を入力
$price = chop(fgets(STDIN));

print "point =>";      //ポイントを入力
$point = chop(fgets(STDIN));

print "fix =>";
$fix = chop(fgets(STDIN));

print"number=>";
$number = chop(fgets(STDIN));

$sql="lock table ".$tablename." in exclusive mode nowait";
//ロックを実行
$stmt=oci_parse($con,$sql);

    $sql="insert into ".$tablename."
values('$ID','$name','$maker','$price','$point',
'$fix','$number')";     //テーブルにデータを追加
    $stmt=oci_parse($con,$sql);
    if(!oci_execute($stmt,OCI_DEFAULT)){
        oci_rollback($con);   //エラーがでたらロールバックを実行
    }else{
        oci_commit($con);    //成功したらコミットを発行
    }

oci_close($con);
```

追加したい商品をテーブル別に更新するようにした。

9.2　在庫数を更新するプログラム

```
define('USER','SYSTEM');
define('PASS','toybox');
define('DB','');

$con=oci_connect(USER,PASS,DB);

print "ID =>";
$ID = chop(fgets(STDIN));          //更新したい商品の ID を入力する

$sql="lock table ".$tablename." in exclusive mode nowait";          //ロッ
クを実行
$stmt=oci_parse($con,$sql);

    $sql="update stock set ITEM_STOCK = ITEM_STOCK -1  where ITEM_ID =
".$ID."";
//アップデートを実行
    $stmt=oci_parse($con,$sql);
    if(!oci_execute($stmt,OCI_DEFAULT)){
        oci_rollback($con);
    }else{
        oci_commit($con);
    }

oci_close($con);
```

今回は商品が購入されると仮定して在庫を 1 つ減らすように更新をした。

9.3　ロックについて

データベースにおいて更新をする際ロックは必要である。なぜならば別のユーザが同時 に更新をしようとした場合、User1 による書き込みが User2 によって上書きされてしまい 正しく更新されないという問題が発生してしまうからである。

しかしロックをかけるといっても完全ロックを行ってしまうと、その間は select なども 実行されなくなってしまうため不便である。そのためできるだけ小さい範囲でロックを かける必要がある。なので今回のプログラムで使用したロックは読み込み以外のアクセ スにのみロックをかけるものを使用した。

10　Level8：速度の測定

Level7 で作成したプログラムを変更し、処理速度を測定するプログラムを書き加える。処理前と処理後の時間を比較して速度を測る。

10.1 プログラム

```
/*時刻を取得する関数*/
function getmicrotime(){
    list($msec, $sec) = explode(" ", microtime());
    return ((float)$sec + (float)$msec);
}

$con=oci_connect(USER,PASS,DB);

//print "ID =>";
//$ID = chop(fgets(STDIN));

/*SQL より ID を取得*/
$sql="select ITEM_ID from stock";
$ID=oci_parse($con,$sql);
oci_execute($ID,OCI_DEFAULT);

/*取得した ID を配列 id_data に格納*/
$i=0;
while(oci_fetch($ID)){
    $id_data[$i++]=oci_result($ID,"ITEM_ID");
}

$MAX = 1000;    //測定の回数

for($i=0;$i<$MAX;$i++){

$Stime = getmicrotime();   //処理前の時刻

$sql="lock table ".$tablename." in exclusive mode nowait";
$stmt=oci_parse($con,$sql);

    $sql="update stock set ITEM_STOCK = ITEM_STOCK -1  where ITEM_ID = ".$id_data[$i]."";
    $stmt=oci_parse($con,$sql);
    if(!oci_execute($stmt,OCI_DEFAULT)){
        oci_rollback($con);
    }else{
        oci_commit($con);
    }

$Etime = getmicrotime();  //処理後の時刻

$tm[$i] = $Etime - $Stime;  //処理時間
print"$tm[$i] s\n";
$stm = $stm + $tm[$i];  //測定した時間の合計
}
$atm = $stm/$MAX;  //測定した時間の平均
print"Sum = $stm s \n";
print"Ave = $atm s \n";
oci_close($con);
```

時刻は microtime() を用いることで小数点以下の時間も取得した。

10.2 実行結果

```
0.10905408859253 s
0.13218903541565 s
0.076097011566162 s
0.12702822685242 s
0.14478015899658 s
0.11919498443604 s
0.10693597793579 s
--省略--
0.13985991477966 s
0.11988806724548 s
0.069957971572876 s
0.15980696678162 s
0.069906949996948 s
0.11997413635254 s
Sum = 114.36270356178 s
Ave = 0.11436270356178 s
```

1000 回出力した結果、データベースの書き換えの処理速度は 1 つのデータ当たり、約 0.1 秒程度だと分かる。

11 Level9：システムの改良

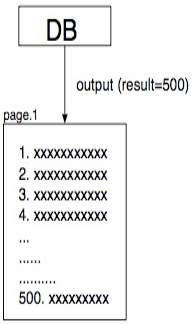
11.1 改良したシステムおよびインターフェース

中村電機オンラインショップ (改) ”http://pw008.st.ie.u-ryukyu.ac.jp/ oracle/j03056/nakaden.php”
(Oracle の使用期間の都合上、閲覧は 2006/01/15 頃まで)

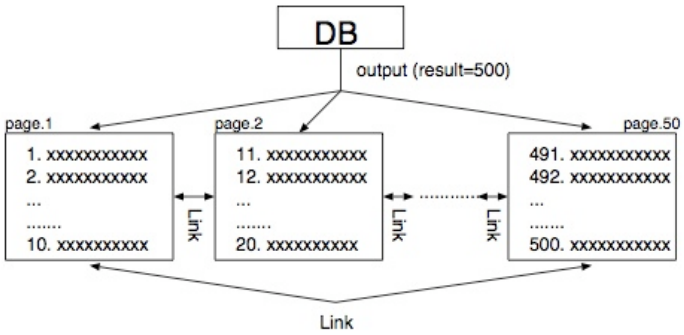
11.2 検索結果の分散表示

11.2.1 ページング

Level6 で作成した閲覧システムでは、検索結果を同ページに全て出力する。そのため、該当件数が膨大な数になってしまうと全てを表示するのに時間が掛かり、計算機への負荷も大きくなる。



そこで、検索結果を指定した件数で分割し、1 ページにおける検索結果の表示数を減らす手法 (ページング) をとることにした。



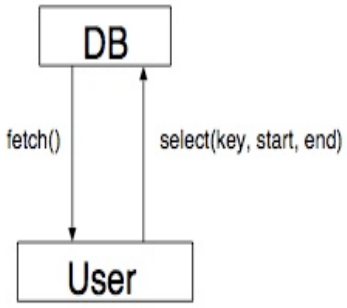
ページングは大手検索サイトである「Yahoo!」や「Google」でも使用されている。各ページはそれぞれに相互リンクを持ち、ユーザが自由に行き来できる。

11.2.2 仮想テーブルの利用

検索結果を分割・表示する処理では

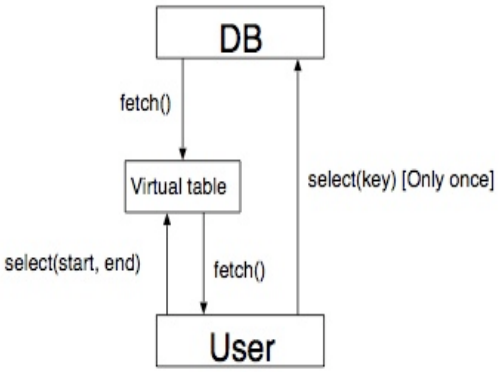
1. 入力したキーワードに該当する行を指定したテーブルから抽出する。
2. 抽出された複数 (または単数) 行から指定範囲行を出力する。(ex. 出力件数が 10 件、3 ページ目の検索結果を見る場合) [(3-1)*10+1, 3*10] = [21, 30] の範囲の行を出力する。

を繰り返し行うことになる。(下図参照)



11.2.1 で示したページングにより生成される、相互リンクでのページ移動の場合、「商品名」や「値段」などの条件は同じである。したがって、ページ移動の時まで DB にキーワードを送信するのは不要である。そこで、今回はビューと言われる機能を使用する。

ビューとは、既に定義されている実テーブルから作成される仮想的なテーブルのことである。ビューの中にはデータは存在せず、実テーブルに対するアクセスの形を定義する。



まず、入力されたキーワードに該当する行を全て抽出する。抽出された行にしたがってビューを生成するため、キーワードによる select は今後新しいキーワードを入力しない限り不要となる。その後、作成されたビューから指定範囲内の行を抽出する。これにより、繰り返し行うデータ抽出の際の SQL 文の短縮に繋がる。

11.3 テーブルの結合

前述の Level で、複数のテーブルにおける select 文の結果の結合を「UNION」によって行っていた。UNION 演算子は、2 つ以上のクエリの出力を和結合、すなわち、クエリの結果を結合し、そこから重複する行を削除する。

今回扱うデータは通信販売での商品情報であり、複数のテーブル (ジャンル) において商品名やメーカー、特に商品 ID などが重複することは考えられない。そこで、UNION 演算子の代わりに「UNION ALL」を使用する。「UNION ALL」は UNION と違って重複する行の削除を、つまり重複チェックを行わない。

そこで、実際に「UNION」と「UNION ALL」での違いを計測する。条件は

- ジャンルを指定しない (全てのテーブルから)
- その他条件をつけない (つまり "select * from A UNION [ALL] select * from B UNION [ALL] ...");
- 11.2.1、11.2.2 を組み込んである (プログラムの説明は 11.4 で)

union.php

```
[oracle@pw008 j03056]$ php union.php // UNION
01:  2.76349 秒
02:  2.70194 秒
03:  2.89512 秒
04:  3.32322 秒
05:  2.85082 秒
06:  2.85565 秒
07:  2.95929 秒
08:  2.78938 秒
09:  2.86383 秒
10:  2.70990 秒
sum: 28.71264 秒
```

```
[oracle@pw008 j03056]$ php union.php // UNION ALL
01:  0.22327 秒
02:  0.21694 秒
03:  0.21755 秒
04:  0.21746 秒
05:  0.22283 秒
06:  0.22368 秒
07:  0.21811 秒
08:  0.21747 秒
09:  0.21932 秒
10:  0.22271 秒
sum: 2.19934 秒
```

結果からわかるように、かなりの時間短縮になった。検索条件が詳細に設定されていればそれほど差は見られないが、データの重複がほとんどない場合に限り、「全検索」といった大規模な検索では非常に有効である。

11.4 プログラム

11.4.1 メインプログラム

nakaden.php

```
//--------------------
// [$flag]
// 1: 検索ページ初期アクセス状態
// 2: 新しいキーワードによる検索結果
// 3: リンクによる検索結果のページ移動
// 4: 商品を購入した
//--------------------
$flag = 1;


〜 html(head) 部省略〜

<center><h2>中村電機</h2></center>

<center>

<?
if ($flag == 4) {
     zaiko_updata($_POST['buy_id']);
     echo $_POST['buy_item']."をお買い上げ真にあざーす !! &;t;br>\n";
}
?>


〜 form 部省略 〜


// DB への接続
$conn = start_connect();

if ($flag == 1) {
     exit_script("");
} else if ($flag == 2) {
     // 入力したキーワードの該当行を抽出する SQL 文の生成
     $sql = make_sql($_POST['cate'],     $_POST['highlows'],
     $_POST['item_word'],   $_POST['maker_word'],
     $_POST['serial_word'], $_POST['price']);
     if ($debug) {
  echo $sql."<br>\n";
     }
     // 仮想テーブルの生成
     $all_ros = create_view($conn, $sql, $view_tbl);
     if($all_ros == NULL) {
  exit_script("fail create_view()<br>\n");
     }
     $_SESSION['all_ros'] = $all_ros;
     $st = 1;
} else {
     $st = ($page-1) * $max_ros + 1;
}

〜省略〜

//---------------------------------
// SQL 文の実行
// $_SESSION['all_ros'] : 仮想テーブル全体の行数
// $ros     : 返されたテーブルの行数 ($max_rows と同値)
// $res     : テーブルの連想配列
//---------------------------------
list($ros,$res) = select_range_data($conn, $view_tbl, $st, $st+$max_ros-1);

if ($ros == NULL) {
     print_r($ros);
     exit_script("fail select_range_data()");
}

$start = ($page-1)*$max_ros+1;
$end   = $start + $max_ros-1;

for ($i = 0; $i < $ros; $i++) {


〜商品表示前半省略〜


  // 在庫チェック
  $zaiko = get_zaiko($conn, $tmp[1]);
  if ($zaiko == NULL) {
       exit_script("fail get_zaiko\n");
  } else if ($zaiko > 1) {
       $msg = "在庫有り";
       if (1) {
    $msg .= " $zaiko";
       }
  } else {
       $msg = "在庫なし";
  }
}


// 同キーワードによる検索結果の他ページへのリンク
for ($i=1; $i*$max_ros < $max_ros+$_SESSION['all_ros']; $i++) {
  if ($i == $page) {
    $next .= " $i ";
  } else {
    $next .= " <a href=\"$PHP_SELF?page=$i\">$i</a> ";
  }
}
echo "$next<br>";
end_connect($conn);
?>
</div>
</body>
</html>
```

ページングやビューの作成の他に

- PHP のセッション変数による、ページ移動の際の変数保持
- 在庫テーブル stock から在庫数をチェックするといった処理を追加した。

SQL プログラム

sql.php

```
function make_sql($cate,$highlows,$item,$maker,$serial,$price) {

～省略～

     if ($table_all) {
  for ($i=1;$i<count($category);$i++) {
       $tmp = $tmp.str_replace('TB',$category[$i],$sql);
       if($i%(count($category)-1)!=0){
            // UNION ALL に変更
    $tmp = $tmp." UNION ALL ";
       }
  }
  $sql = $tmp;
     }
     return $sql;
}

//------------------------------
// $sql によって抽出された行を
// 仮想テーブルとして生成する
//
// arg
//   $con      : 接続 id
//   $sql      : SQL 文
//   $view_tbl : 仮想テーブル名
//
// return
//   $view_tbl の行数
//------------------------------
function create_view($con, $sql, $view_tbl) {
  $sql  = "create or replace view $view_tbl as ".$sql;
  $stmt = oci_parse($con,$sql);

  if (!oci_execute($stmt,OCI_DEFAULT)) {
       echo "error create_view\n";
       oci_free_statement($stmt);
       return NULL;
  }
  oci_free_statement($stmt);

  /* $view_tbl の行数を計算する */
  $stmt = oci_parse($con, "select count(*) from $view_tbl");
  if(!oci_execute($stmt,OCI_DEFAULT)){
       echo "error create_view_2\n";
       oci_free_statement($stmt);
       return NULL;
  }
  if (!($all_ros = oci_fetch_row($stmt))) {
       echo "error create_view3\n";
       oci_free_statement($stmt);
       return NULL;
  }
  oci_free_statement($stmt);
  return $all_ros[0];
}

/*
 * 指定したテーブルから指定した件数のデータを返す
 */
function select_range_data($con, $tbl, $start, $end)
{
     $sql  = "select * from (";
     $sql .= "select rownum as seq_no,$tbl.* from $tbl) ";
     $sql .= "where seq_no between $start and $end";

     $stmt = oci_parse($con,$sql);
     if(!oci_execute($stmt,OCI_DEFAULT)){
  echo "error select_range_data\n";
  oci_free_statement($stmt);
  return array(NULL, NULL);
     }

     $ros = oci_fetch_all($stmt,$res);

     oci_free_statement($stmt);
     return array($ros,$res);
}


/*
 * 商品 ID が $id の在庫を取得する。
 */
function get_zaiko($con, $id)
{
     $sql = "select item_stock from stock where item_id = '".$id."'";
     $stmt = oci_parse($con,$sql);

     if (!oci_execute($stmt,OCI_DEFAULT)) {
  echo "error get_zaiko\n";
  oci_free_statement($stmt);
  return NULL;
     }

     if (!($zaiko = oci_fetch_row($stmt))) {
  echo "error get_zaiko2\n";
  oci_free_statement($stmt);
  return NULL;
     }

     oci_free_statement($stmt);
     return $zaiko[0];
}
```

```
/*
 * 在庫の更新（在庫数-1）
 */
function zaiko_updata($ID)
{
     $sql  = "lock table stock in exclusive mode nowait";
     $con  = start_connect();
     $stmt = oci_parse($con,$sql);

     $sql  = "update stock set ITEM_STOCK = ITEM_STOCK -1 where ITEM_ID = $ID";

     $stmt=oci_parse($con,$sql);
     if (!oci_execute($stmt,OCI_DEFAULT)) {
  oci_rollback($con);
  return NULL;
     } else {
  oci_commit($con);
     }

     oci_free_statement($stmt);
     oci_close($con);
}
```

ビューの作成、及びテーブルの範囲指定の SQL 構文は以下の通りである。

ビューの作成

SQL > create or replace view ビュー名 as select ....;

「or replace」により、同じ名前の view があれば上書きする。

```
SQL> select * from (
   > select rownum as seq_no,テーブル名.* from テーブル名 where ...)
   > where seq_no between START and END;
```

まずは副問い合わせで、条件に合う行に seq_no という擬似列を追加する。seq_no には昇順で整数が入る。その後、where 句の between によって START から END までの行を抽出する。なお、プログラムの実行は
`http://pw008.st.ie.u-ryukyu.ac.jp/~oracle/oratest.php`
にて行える。

11.5 実行結果

図 4: 初期アクセス状況

図 5: 検索結果

図 6: 他ページへのリンク

図 7: 該当件数多過時の処理

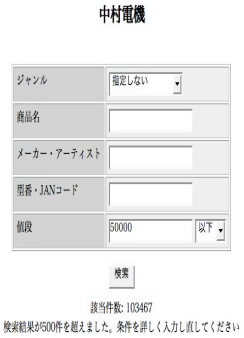
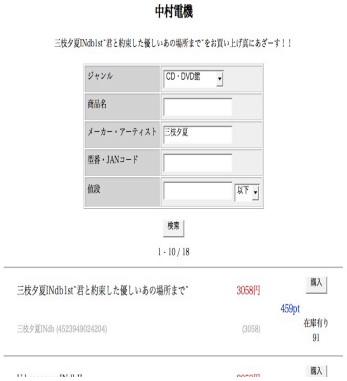


図 8: 購入ボタン押下時



12 LevelX

12.1 サーブレットの設定

今回、我々は tomcat を使用する事にした。オラクル 9i の実行環境と同じ環境で動くものとして、バージョンは 4.1.31 を選択した。tomcat.apache.org から、バイナリをダウンロードし、ホーム上で展開して使用した。tomcat は、デフォルトでポート 8080 が指定されている。pw008 上では、他のプロセスとポート番号がぶつかってしまったので、8081 にポート番号を変更する事にした。tomcat の設定ファイルは、tomcat/conf 内の server.xml に記述されている。server.xml 上で、ポート番号の変更と、新しいディレクトリの定義を行った。これは、我々が使用するディレクトリを、ネット上で公開するために必要な設定である。tomcat は、server.xml 上で指定したディレクトリのみのアクセスを許可する。webapps 以下に、server.xml で設定した nakaden と、nakaden/WEB-INF、nakaden/WEB-INF/classes、nakaden/WEB-INF/lib を手動で作成す

る。classes は、ウェブ上で実際に動作するサーブレットクラスを格納する。しかし、これも class を置くだけではなく、nakaden/web.xml を作成して、明示的に記述しなければ、アクセスできない。もう一つの方法として、conf/web.xml にて、サーブレットのマッピングをどのように行うか、設定できる。我々は、後者を選択し、実験を行った。記述した内容は以下の通りである。

```
<servlet-mapping>
<servlet-name>default</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
```

12.2 nakaden 検索システム

プログラム構成は、MVC プログラミングに基づくように設計した。モデル部分が、Oracle_model.java、Config.java、ビューア部分が html_View.java、コントロール部分を、System_Controller.java、list.java が担当している。

図 9: GET が要求されたとき

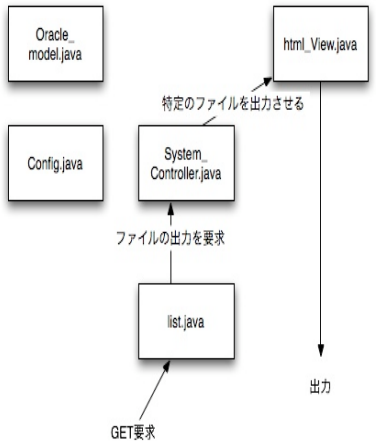
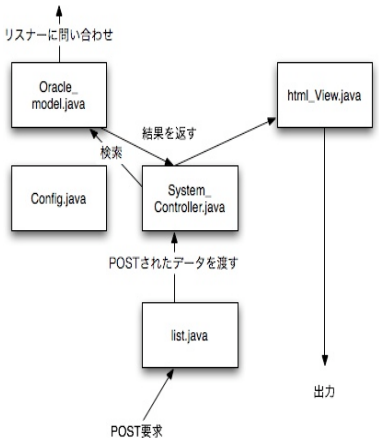


図 10: POST が要求されたとき



データベースへのコネクションは、Oracle で起動しているリスナーを経由して、Oracle JDBC OracleDriver を経由して行う。一般的に、OracleDriver は java のライブラリに存在しないものである。使用するにあたって、/u01/app/oracle/product/10.2/jdbc/lib/classes12.jar へのパスを通す。それにより、実行可能になる。

Class.forName(”oracle.jdbc.driver.OracleDriver”); これは、見てもわかる通り、実行時にドライバを検索し、ロードする。実行はサーブレット上で行うので、環境変数の CLASSPATH は読み込んでくれない。この問題を解決するためには、サーブレット上のディレクトリ、WEB-INF/lib/に jar ファイルを置く事により、実行時にロードする事が可能になる。しかし、このままコン

パイルして実行しても文字化けが発生する。javac -encoding shift_jis といった具合にエンコーディングを指定しないと文字化けする。一般的に、コマンドライン上での実行においては、各々の java ソースにおいて、文字コードが異なっていても文字化けする事は無い。原因として考えられるのは、servlet 上のバーチャルマシンは、j2se のバーチャルマシンの動作が微妙に異なると思われる。明示的にエンコーディングを指定しなければ、化けてしまう。

12.3 比較する

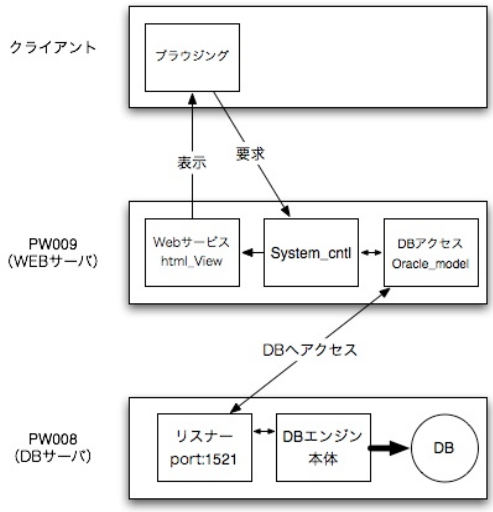
中村電機PHP バージョン (”http://pw008.st.ie.u-ryukyu.ac.jp/ oracle/oratest.php”)
中村電機サーブレットバージョン (”http://pw008.st.ie.u-ryukyu.ac.jp:8081/nakaden/servlet/list”)
System_Controller に System.currentTimeMillis(); を用いて、開始時と終了時を獲得して、検索結果の一番下に出力するようにした。

検索内容	検索件数	時間
商品名=S9000、AV デジカメ館	1 件	229ms
キーワード=北原愛子、CD・DVD 館	12 件	341msec
キーワード=JAMProject、CD・DVD 館	36 件	523msec
商品名=VAIO、CD・DVD 館	2 件	212msec
商品名=KENWOOD、AV デジカメ館	2 件	146msec
商品名=鉛筆、文具オフィス館	20 件	412msec
キーワード=ツインバード、家電	11 件	1304msec
商品名=ベーカリー、家電	5 件	307msec
商品名=眼鏡、家電	0 件	1231msec
型番=FXZ1B,AV デジカメ館	4 件	319msec

php バージョンは平均、114ms、サーブレット 504ms。かなりの差が見られる。原因として、リスナーを経由している事が考えられる。

12.4 改善

少しでも早くするために、三層クライアント・サーバ型 データベース・システムを実装する。こうしようと思った理由は、サーブレットを立ち上げるにあたって、多くの java プロセスが立ち上がる。Oracle の起動だけでもメモリが足りてないので、WEB サーバを別にすることにより、ページングの量や、CPU の負荷軽減に繋がると考えた。



同じキーワードで検索した。

検索内容	検索件数	時間
商品名=S9000、AV デジカメ館	1 件	21295ms
キーワード=北原愛子、CD・DVD 館	12 件	271msec
キーワード=JAMProject、CD・DVD 館	36 件	806msec
商品名=VAIO、CD・DVD 館	2 件	167msec
商品名=KENWOOD、AV デジカメ館	2 件	89msec
商品名=鉛筆、文具オフィス館	20 件	189msec
キーワード=ツインバード、家電	11 件	78msec
商品名=ベーカリー、家電	5 件	133msec
商品名=眼鏡、家電	0 件	76msec
型番=FXZ1B,AV デジカメ館	4 件	68msec

速度にむらが出てきた。一回目以外のデータに関しては、速度が速くなっている。これはリスナーとの接続に時間がかかってしまうためだと思われる。２回目以降からの実行は高速に行える。 以上のようにサーブレットを用いてシステムの改良を実現した。