データベースS

第11回　PHPによるWebインターフェース開発演習(2)

システム創成情報工学科　尾下 真樹

2018年度 Q2

今日の内容

- 前回の復習

- PHPによるインターフェース開発（2）
- レポート課題

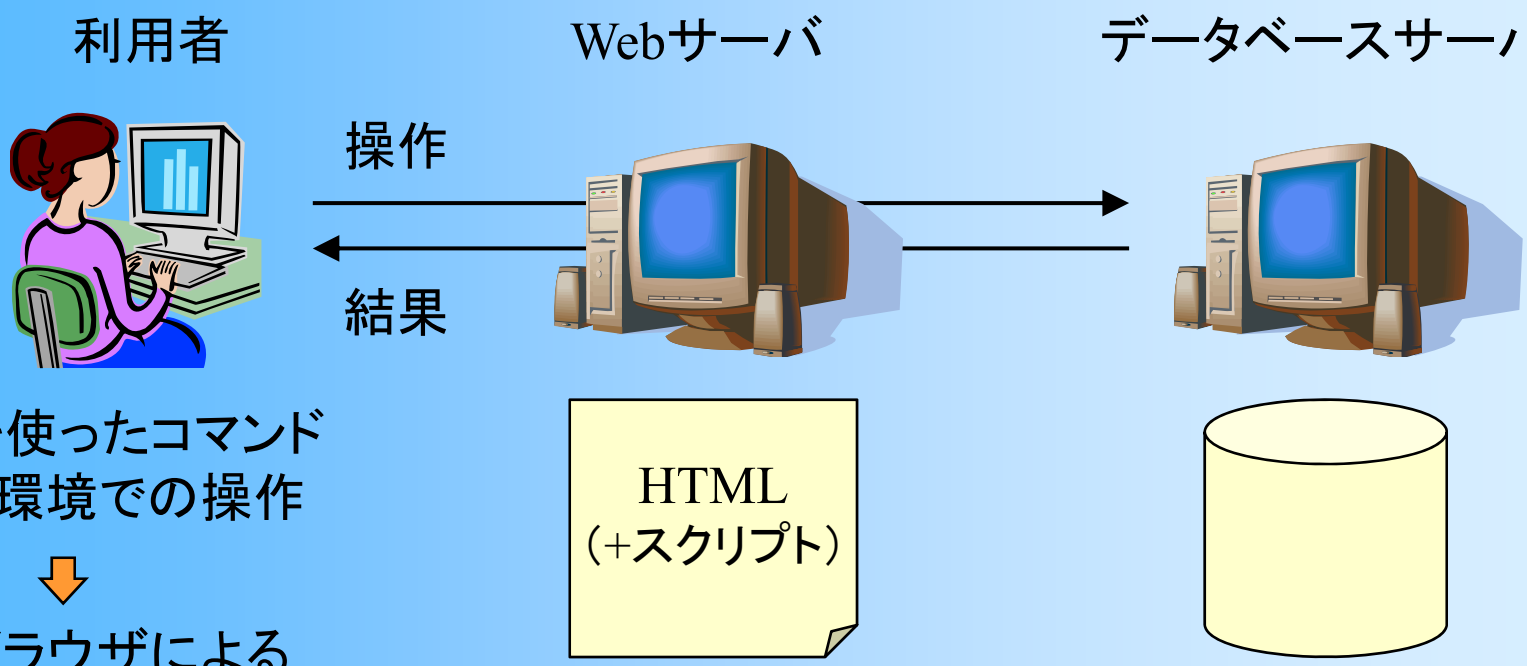
参考書

- 「PHP5 徹底攻略」
  堀田 倫英、桑村 潤 著
  ソフトバンクパブリッシング（3,800円）
  - PHP（本日説明）＋ PostgreSQL についての詳しい参考書

前回の復習

Webインターフェース

- Webページを経由してデータベースを操作



SQLを使ったコマンドライン環境での操作

⇩

Webブラウザによる
GUI環境での操作
(データベースを意識しなくても使える)

HTML中にスクリプトを記述することで、データベースにアクセス

データを管理
コマンドラインインターフェース

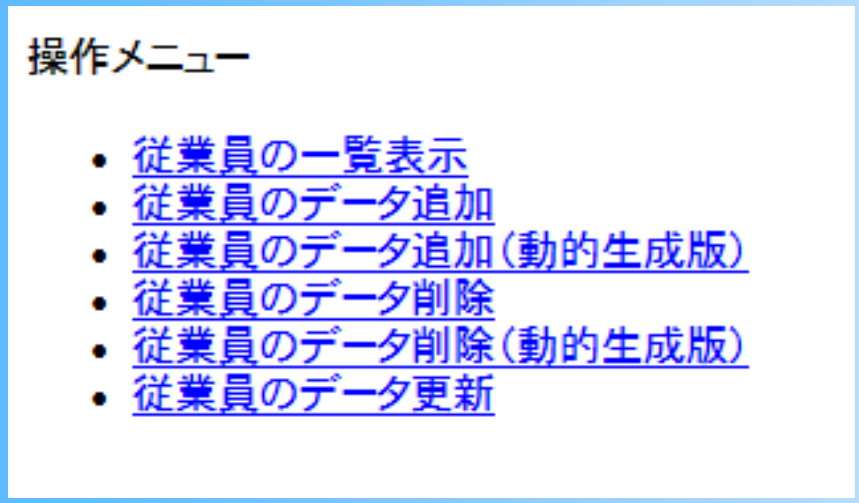
HTMLファイルの例

- メニュー（menu.html）

```
<HTML>
<HEAD>
  <TITLE>データ操作メニュー</TITLE>
</HEAD>
<BODY>
操作メニュー<BR>
<UL>
    <LI><A HREF="employee_list.php">従業員の一覧表示</A>
    <LI><A HREF="employee_add_form.html">従業員のデータ追加</A>
    <LI><A HREF="employee_add_form.php">従業員のデータ追加（動的生成版）</A>
    <LI><A HREF="employee_delete_form.html">従業員のデータ削除</A>
    <LI><A HREF="employee_delete_form.php">従業員のデータ削除（動的生成版）</A>
    <LI><A HREF="employee_update_form1.html">従業員のデータ更新</A>
</UL>
</BODY>
</HTML>
```

HTMLファイルの表示結果の例

- ウェブブラウザでの表示結果
  - フォントの種類や大きさ等は、ブラウザの設定により異なる



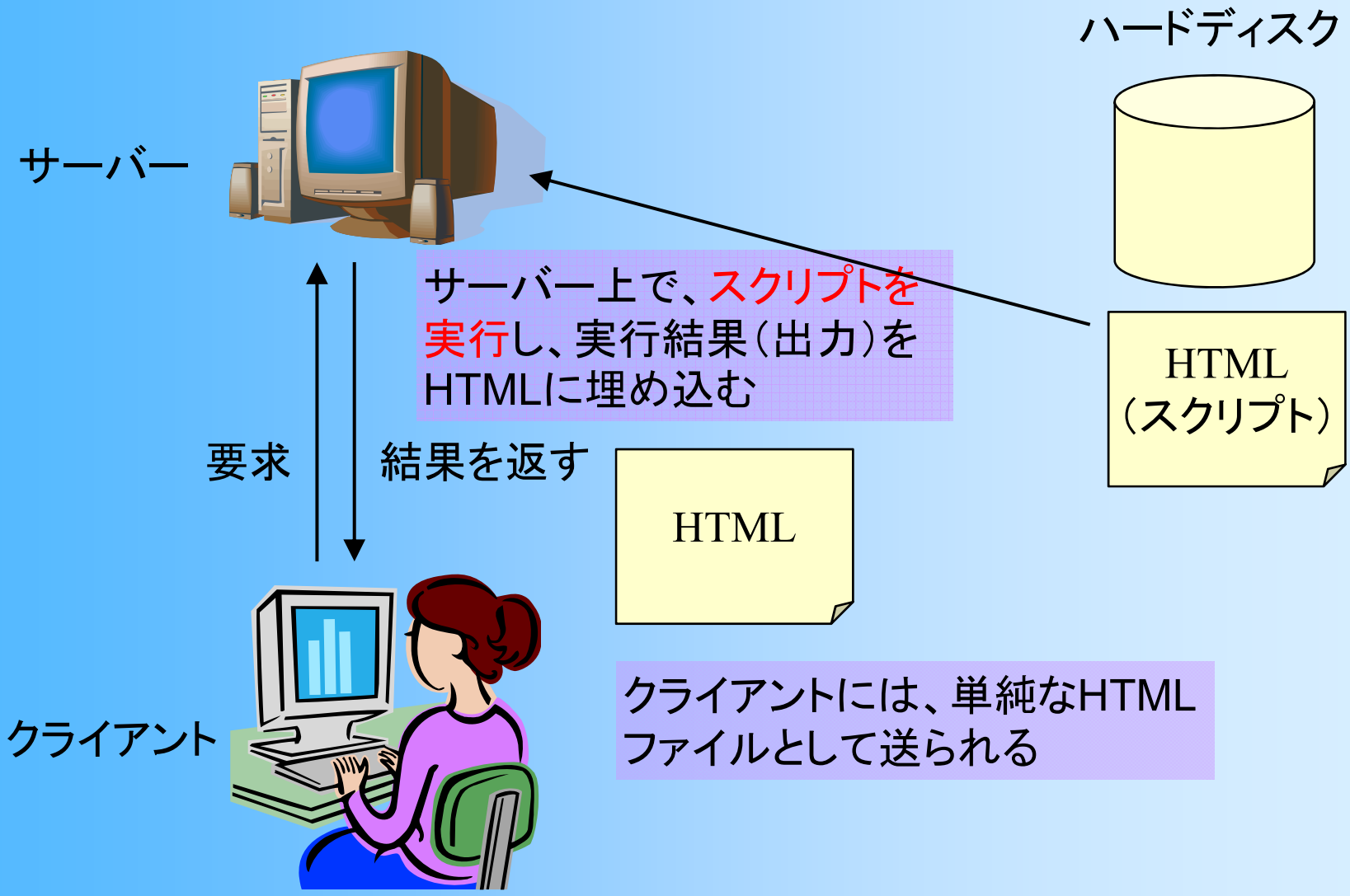
PHPの利用

- PHP
  - サーバーサイド・スクリプトとしての利用に適したプログラミング言語
    - JavaやC++と同様のオブジェクト指向言語
  - Webシステムの開発に便利な標準関数を備える
    - 多くのWebシステムで使われている（WordPress等）
- 本演習では、PHPを使って、Webインターフェースを作成する

PHPの利用形態

- サーバーサイド・スクリプトとしての利用
  - HTML と PHPスクリプト が混在したソースファイルを記述
  - ウェブサーバ側で PHPスクリプトを実行
    - PHPスクリプトから出力されたテキストが、元のHTMLに埋め込まれる
  - クライアント（ウェブブラウザ）には、最終的に生成された HTML が送られる

サーバサイド・スクリプト



PHPの記述(1)

- HTML内へのPHPスクリプトの記述
  - <?php ～ ?>
- PHPスクリプト
  - if や while などの制御構文は、Java や C と同じ
- 変数
  - $で始まる文字列を変数とみなす
  - 宣言せずに使って良い
  - 型は指定しなくても良い(値により自動的に決まる)
  - Javaや C とは、上記の点が大きく異なるので注意

PHPからPostgreSQLの操作

- 専用の関数が用意されている
  - pg_ で始まる関数
  - pg_connect(string option);
    - データベースに接続
  - pg_query(query);
    - クエリーを実行
  - pg_num_rows(result);
    - クエリーの結果の行数を取得
  - pg_fetch_result(result, i, j);
    - クエリーの結果のテーブルから i行j列の値を取得
      - i,j は 0 から始まることに注意（例：2行3列目→ i=1, j=2 ）
  - pg_close();

従業員番号	部門番号	氏名	年齢
0001	01	織田 信長	48
0002	02	豊臣 秀吉	45
0003	03	徳川 家康	39
0004	01	柴田 勝家	60
0005	01	伊達 政宗	15
0006	02	上杉 景勝	26
0007	03	島津 家久	35

SQL文の作成

- SQL文は文字列として扱える
  - $sql = "select * from employee where id='001'";
    - 注意：" （ダブルクォート）はPHPの文字列の区切り、
      ' （シングルクォート）はSQLの文字列の区切り
- 文字列を埋め込むことで動的にSQL文を作成できる（以下の3つは、どれも同じ結果になる）

```
$sql = "select * from employee where id='" . $id . "'";
```

文字列の連結

```
$sql = "select * from employee where id='$id'";
```

$id の値が文字列に埋め込まれる

```
$sql = sprintf( "select *  from employee where id='%s'", $id );
```

文字列中の %s の箇所が、$id の値で置き換えられる

ウェブページの準備

- ウェブサーバ
  - http://popuradb.ces.kyutech.ac.jp
  - 今回はデータベースサーバと同じコンピュータ

    ※ 学科外からはアクセスできないので注意
- 以下のディレクトリにファイルを置く
  - ホームディレクトリの public_html ディレクトリ
- 以下のURLでアクセスできる
  - http://popuradb.ces.kyutech.ac.jp/~ユーザ名/

演習手順

- データベースの準備
  - テーブルの作成、データの追加（前回終了）
  - テーブルの利用権限の設定
- html(php) ファイルの作成
  1. 講義のページからダウンロードした menu.html を適切な場所に置き、表示されることを確認
  2. 同じく employee_list.php を置き（一部修正が必要）、従業員一覧が表示されることを確認
  3. 他のファイル（追加、更新、削除）についても、動作を確認（次回行う）

演習課題

- 前回までの演習は完了しているものとする
- メニュー・一覧表示（menu.html, employee_list.php）のファイルをアップロード（＋一部変更）して、動作確認をする
- 一覧表示を行なうPHPプログラムを一部変更して、従業員の一覧が部門ごとに表示されるようにする（exmployee_list.phpを変更）

PHPによるインターフェース開発（1）

（前回の復習）

インターフェースの作成

- 作成する機能
  - 従業員データの一覧表示
  - 従業員データの追加
  - 従業員データの追加(動的生成)
  - 従業員データの削除
  - 従業員データの削除(動的生成)
  - 従業員データの更新
  - 従業員データの検索

サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

メニュー

- メニュー（menu.html）
  - <HTML> <HEAD> <TITILE> <BYDY>
  - <UL>～</UL> <LI> によるリスト
  - 各機能のページへのリンク
    <A HREF=“…”> ～ </A>

メニュー

- メニュー（menu.html）

```
<HTML>
<HEAD>
  <TITLE>データ操作メニュー</TITLE>
</HEAD>
<BODY>
操作メニュー<BR>
<UL>
    <LI><A HREF="employee_list.php">従業員の一覧表示</A>
    <LI><A HREF="employee_add_form.html">従業員のデータ追加</A>
    <LI><A HREF="employee_add_form.php">従業員のデータ追加（動的生成版）</A>
    <LI><A HREF="employee_delete_form.html">従業員のデータ削除</A>
    <LI><A HREF="employee_delete_form.php">従業員のデータ削除（動的生成版）</A>
    <LI><A HREF="employee_update_form1.html">従業員のデータ更新</A>
</UL>
</BODY>
</HTML>
```

ヘッダ情報

本文

改行

箇条書き

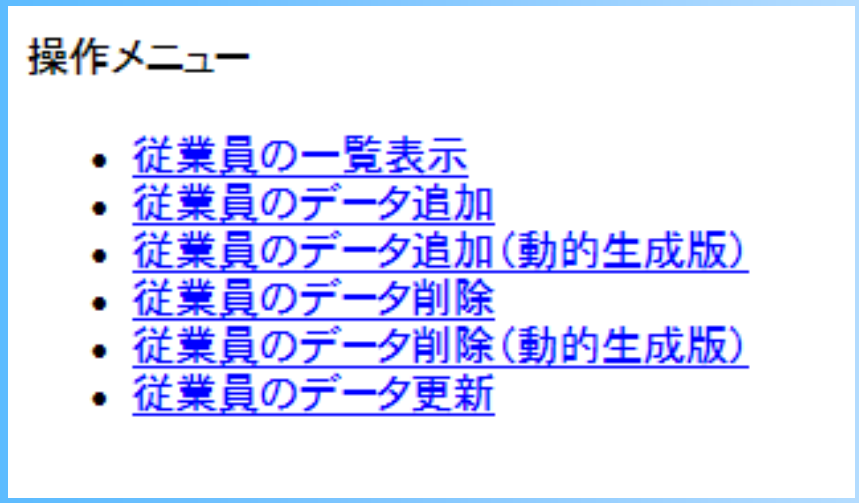
箇条書きの項目

リンク（リンク先のURL）

リンク閉じ

表示結果

- ウェブブラウザでの表示結果
  - フォントの種類や大きさ等は、ブラウザの設定により異なる



サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

一覧表示(1)

- 一覧表示(exmployee_list.php)
  - PHPプログラムの開始（12行目）
  - データベースへの接続（16行目）
    - データベース名を、各自の名前に変更する必要がある（前回の資料の通りに作業していれば、自分のアカウント名でデータベースを作成しているはず）
    - 接続情報を $conn に記録

一覧表示(2)

```
<HTML>
<HEAD>
  <TITLE>従業員リスト</TITLE>
</HEAD>
<BODY>
<CENTER>
検索結果を表示します。<BR><BR>

<!-- ここからPHPのスクリプト始まり -->
<?php
// データベースに接続
// ※ your_db_name のところは自分のデータベース名に書き換える
$conn = pg_connect( "dbname=your_db_name" );
// 接続が成功したかどうか確認
if ( $conn == null )
{
        print( "データベース接続処理でエラーが発生しました。<BR>" );
        exit;
}
```

PHPスクリプトの開始

PostgreSQLデータベースへの接続を行う、PHPの関数

データベース名を指定（自分のデータベース名に書き換える）

接続情報が返される（失敗時はnull）

一覧表示(3)

- 一覧表示(exmployee_list.php)
  - SQL文を実行 (26, 29行目)
    - 全従業員のデータを取得するSQL文(変数 $sql)
    - 検索結果のテーブルが $result に格納される

```
// SQLを作成
$sql = "select id, department.name, employee.name, age from employee,
department where employee.dept_no = department.dept_no order by id";

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );
if ( $result == null )
{
    print( "クエリー実行処理でエラーが発生しました。<BR>" );
    exit;
}
```

一覧表示(4)

- 一覧表示(exmployee_list.php)
  - 検索結果の行数・列数を取得(37, 38行目)
    - SQL文で4つの出力属性を指定しているため、列数は必ず4になる(今回は、わざわざ列数を取得しなくても分かっているが、例のために、取得している)

```
// 検索結果の行数・列数を取得
$rows = pg_num_rows( $result );
$cols = pg_num_fields( $result );
```

引数には、さきほどのSQLの実行結果を格納した変数を指定

SQLの実行結果から、行数(データ数)と列数(属性数)を取得するPHPの関数

一覧表示(5)

- 一覧表示(exmployee_list.php)
  - テーブルを使って結果を表示(42～69行目)
    - <TABLE> <TR> <TD>
  - 各データ(検索結果の各行)の情報を順番に表示(53～65行目)
    - for 文を使って、各行・列ごとに繰り返し
  - 検索結果から属性値を取得して表示(59行目)
    - pg_fetch_result(結果, 行番号, 列番号)

一覧表示(6)

```
// 検索結果をテーブルとして表示
print( "<TABLE BORDER=1>¥n" );

// 各列の名前を表示
print( "<TR>" );
print( "<TH>従業員番号</TH>" );
print( "<TH>部門</TH>" );
print( "<TH>氏名</TH>" );
print( "<TH>年齢</TH>" );
print( "</TR>¥n" );

.........
```

テーブルの開始

1行目の見出しの表示

検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	尾下 真樹	27
0002	営業	下戸 彩	17
0003	総務	本村 拓哉	30
0004	開発	宇田 ヒカル	20
0005	開発	織口 裕二	35
0006	営業	松浦 亜矢	17
0007	総務	山田 一郎	30

以上、7 件のデータを表示しました。

操作メニューに戻る

表示されるテーブル

一覧表示(7)

```
// 各行のデータを表示
for ( $j=0; $j<$rows; $j++ )
{
        print( "<TR>" );
        for ( $i=0; $i<$cols; $i++ )
        {
                // j行i列のデータを取得
                $data = pg_fetch_result( $result, $j, $i );

                // テーブルのj行i列に属性値を表示
                print( "<TD> $data </TD>" );
        }
        print( "</TR>¥n" );
}

// ここまででテーブル終了
print( "</TABLE>" );
print( "<BR>¥n" );
```

テーブルの各行ごとに繰り返し

各行全体を <TR> タグで囲む

各列ごとに繰り返し

SQLの実行結果からj行i列の属性値を取得するPHPの関数

各セルを <TD> タグで囲む

変数 $data の値を表示

一覧表示(8)

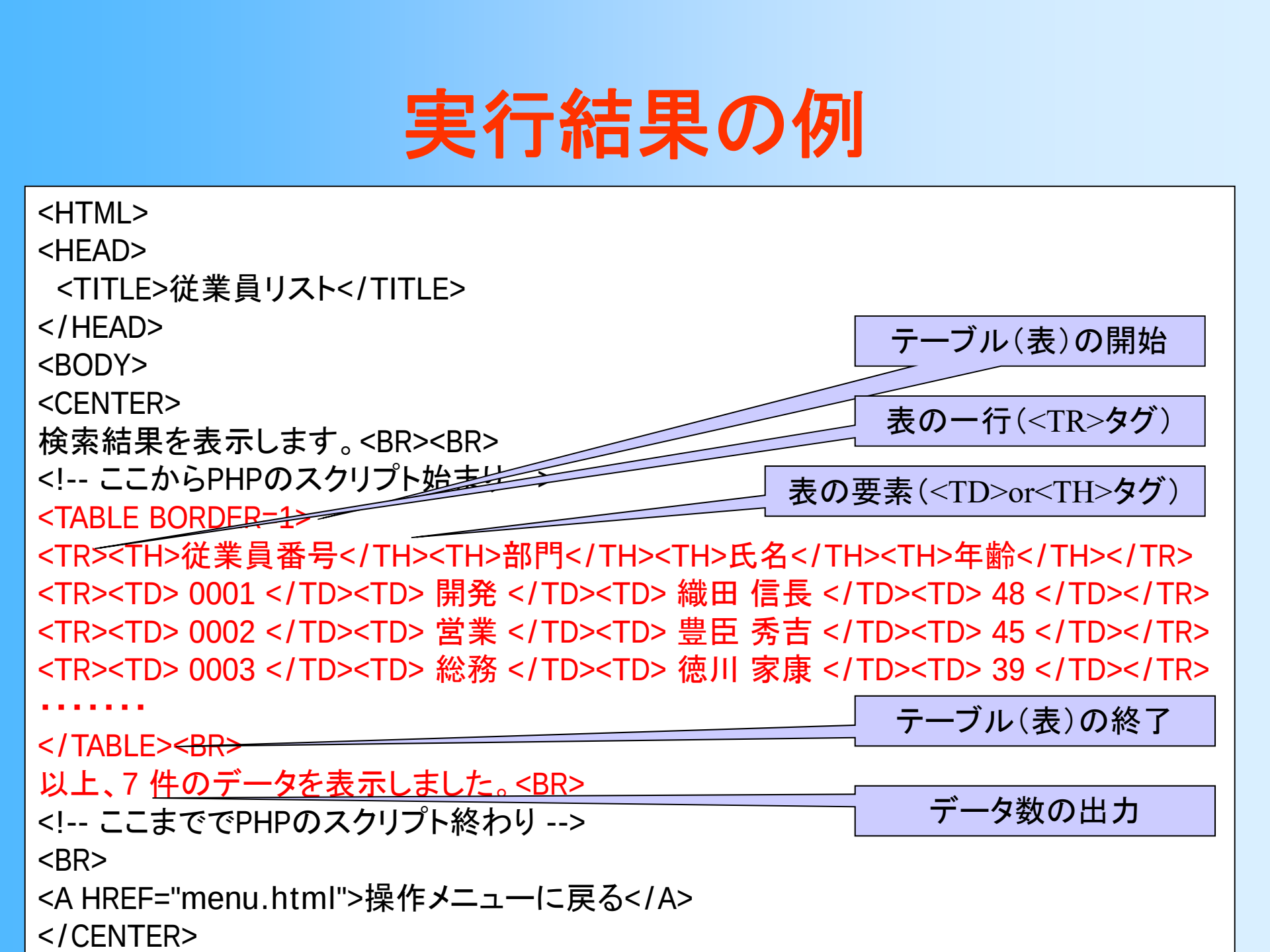
- 一覧表示(exmployee_list.php)
  - データ数を表示(74行目)
  - 終了処理(78, 81行目)
    - 検索結果の開放
    - データベースへの接続を解除

```
// 検索件数を表示
print( "以上、$rows 件のデータを表示しました。<BR>¥n" );

// 検索結果の開放
pg_free_result( $result );

// データベースへの接続を解除
pg_close( $conn );
```

文字列の中に、変数 $rows の値が埋め込まれて出力される

実行結果の例

```
<HTML>
<HEAD>
  <TITLE>従業員リスト</TITLE>
</HEAD>
<BODY>
<CENTER>
検索結果を表示します。<BR><BR>
<!-- ここからPHPのスクリプト始まり -->
<TABLE BORDER=1>
<TR><TH>従業員番号</TH><TH>部門</TH><TH>氏名</TH><TH>年齢</TH></TR>
<TR><TD> 0001 </TD><TD> 開発 </TD><TD> 織田 信長 </TD><TD> 48 </TD></TR>
<TR><TD> 0002 </TD><TD> 営業 </TD><TD> 豊臣 秀吉 </TD><TD> 45 </TD></TR>
<TR><TD> 0003 </TD><TD> 総務 </TD><TD> 徳川 家康 </TD><TD> 39 </TD></TR>
・・・・・・・
</TABLE><BR>
以上、7 件のデータを表示しました。<BR>
<!-- ここまででPHPのスクリプト終わり -->
<BR>
<A HREF="menu.html">操作メニューに戻る</A>
</CENTER>
```



表示結果の例

- ウェブブラウザでの表示結果

検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15
0006	総務	上杉 景勝	26
0007	開発	島津 家久	35

以上、7 件のデータを表示しました。

操作メニューに戻る

テーブル（表）として表示される

PHPによるインターフェース開発（2）

サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

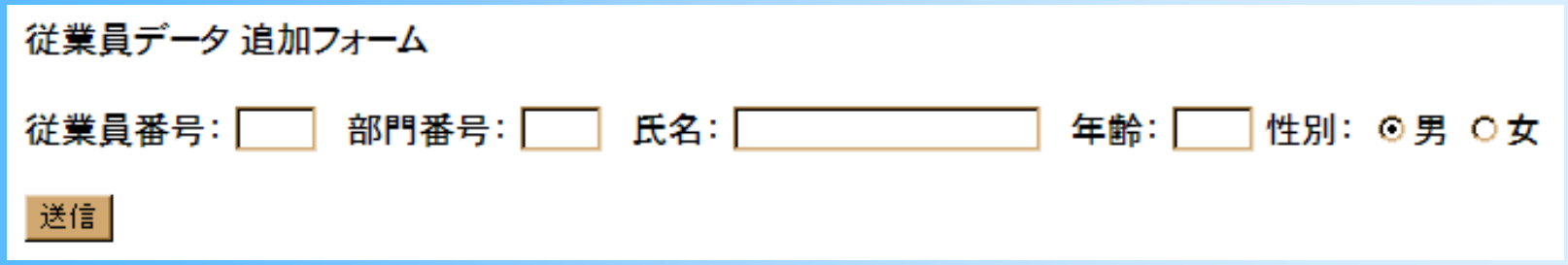
追加

- 2つのページにより実現される

1. 追加フォーム（exmployee_add.html）
   - HTMLのフォームを使ってデータを入力できるようにする
   - 各データの変数名を指定（次のページでデータを受け取るために必要）
2. 追加処理（exmployee_add.php）
   - 前のページで入力されたデータをもとに、データ追加のためのSQL文を作成し、実行

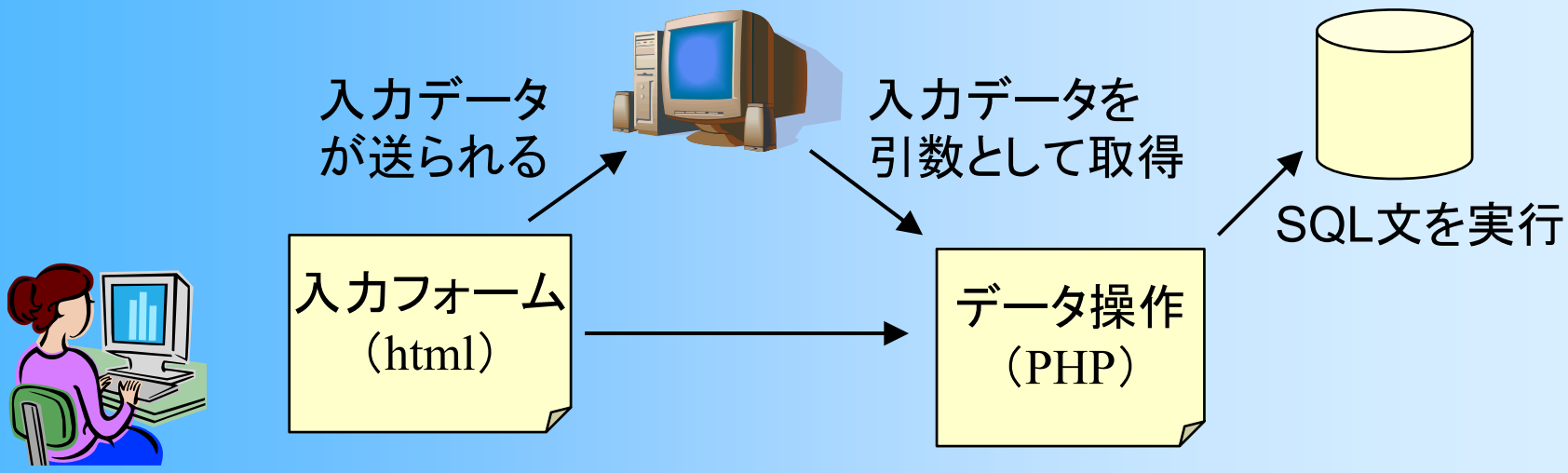
フォーム

- ウェブページに入力できる仕組み
  - <FORM> ～ </FORM>
  - 送信ボタンを押すと、指定したURLを呼び出し
  - フォーム記入したデータをURLで指定したプログラムに引数として送信できる
    - データの受け渡し方に、GET と POST の２種類がある



PHPでの引数の受け取り

- スーパーグローバル変数経由で取得
  - フォーム側が GET で出力した場合
    - $_GET[引数名]
  - フォーム側が POST で出力した場合
    - $_POST[引数名]

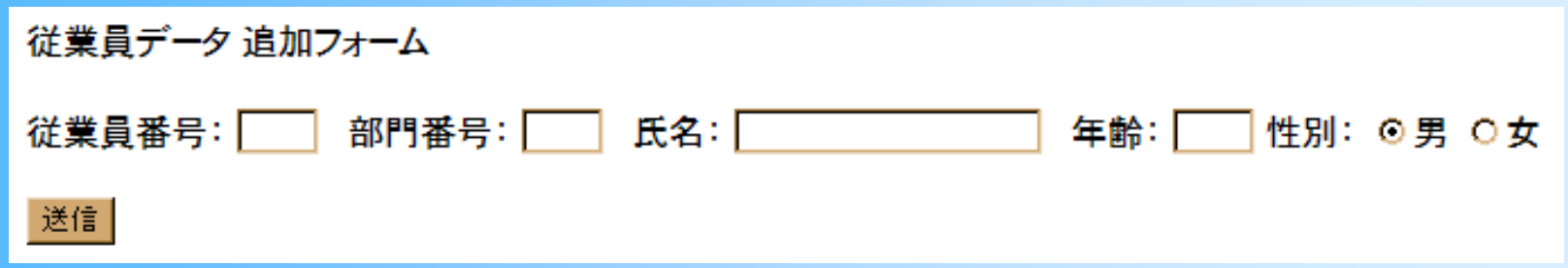


引数の受け渡し方の違い

- GET
  - URLに付与する形で受け渡し（利用者に引数が見える）
  - サーバー側のプログラムは、環境変数という仕組みを使って受け取る
- POST
  - ウェブブラウザとサーバーが通信をして受け渡し（利用者には引数が見えない）
- 両者の使い分け
  - 一般に、POSTの方が処理が面倒な代わりに、高機能
  - PHPで使う場合は、PHPが細かい処理をやってくれるので、どちらも簡単に使える
  - 本演習では、引数が分かりやすいように、GETを使用

追加フォーム

- 追加フォーム（exmployee_add.html）
  - フォームの開始（9行目）
    - <FORM ACTION=“～” METHOD=“～” >
  - 各入力フィールド（12～28行目）
    - <INPUT TYPE =“～” NAME=“変数名” >



追加フォーム

フォームの開始

データ送信後に実行されるページ

データの受け渡し方法に GET を使用

```
.......
従業員データ 追加フォーム<BR><BR>
<FORM ACTION="employee_add.php" METHOD="GET">
従業員番号:
<INPUT TYPE="text" SIZE="4" NAME="id">
部門番号:
<INPUT TYPE="text" SIZE="4" NAME="dept_no">
氏名:
<INPUT TYPE="text" SIZE="24" NAME="name">
年齢:
<INPUT TYPE="text" SIZE="4"
性別:
<INPUT TYPE="radio" NAME="sex" VALUE="MALE" CHECKED>男</INPUT>
<INPUT TYPE="radio" NAME="sex" VALUE="FIMALE">女</INPUT>
<BR><BR>
<INPUT TYPE="submit" VALUE="送信"><BR>
</FORM>
.......
```

テキスト入力エリアを表示
（4文字分のスペースを用意）

入力されたデータは、id という名前で、実行ページに渡される

選択項目（ラジオボタン）。この項目が選択されると、id という名前のデータに、文字列 MALE が入る。

送信ボタンを表示

追加処理(1)

- 追加処理(exmployee_add.php)
  - データベースへの接続などは、一覧表示と同じ(説明は省略)
  - フォームから渡された引数を取得(11～14行目)
    - $GET[変数名]
    - 取得データを変数に格納 $id, $dept_no, $name, $age

```
// フォームから渡された引数を取得
$id = $_GET[ id ];
$dept_no = $_GET[ dept_no ];
$name = $_GET[ name ];
$age = $_GET[ age ];
```

$_GET は、PHPが持つグローバル変数(配列)
前のページのフォームから GET を使って渡された値を受け取ることができる

追加処理(2)

- 追加処理(exmployee_add.php)
  - データ追加のためのSQL文を作成(28行目)
    - ここでは、sprintf を使う方法を使用 (前回の講義で説明したように、別の方法を使っても構わない)
  - SQLの実行(31～33行目)

```
// データ挿入のSQLを作成
$sql = sprintf( "insert into employee( id, dept_no, name, age ) values( '%s', '%s', '%s', '%s' );", $id, $dept_no, $name, $age );

// 確認用のメッセージ表示
print( "クエリー「" );  print( $sql );  print( "」を実行します。<BR>" );

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );
```

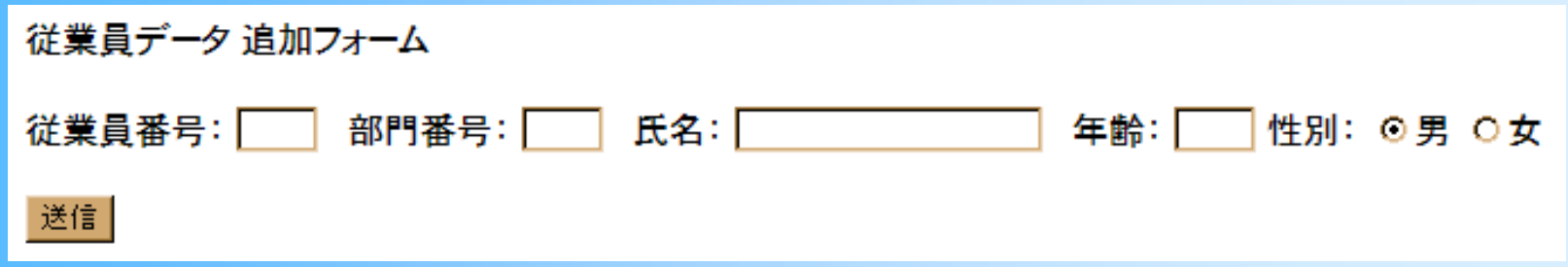
4つの %s は、次の4つの引数の値に置き換えられる

サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

追加フォームの動的生成

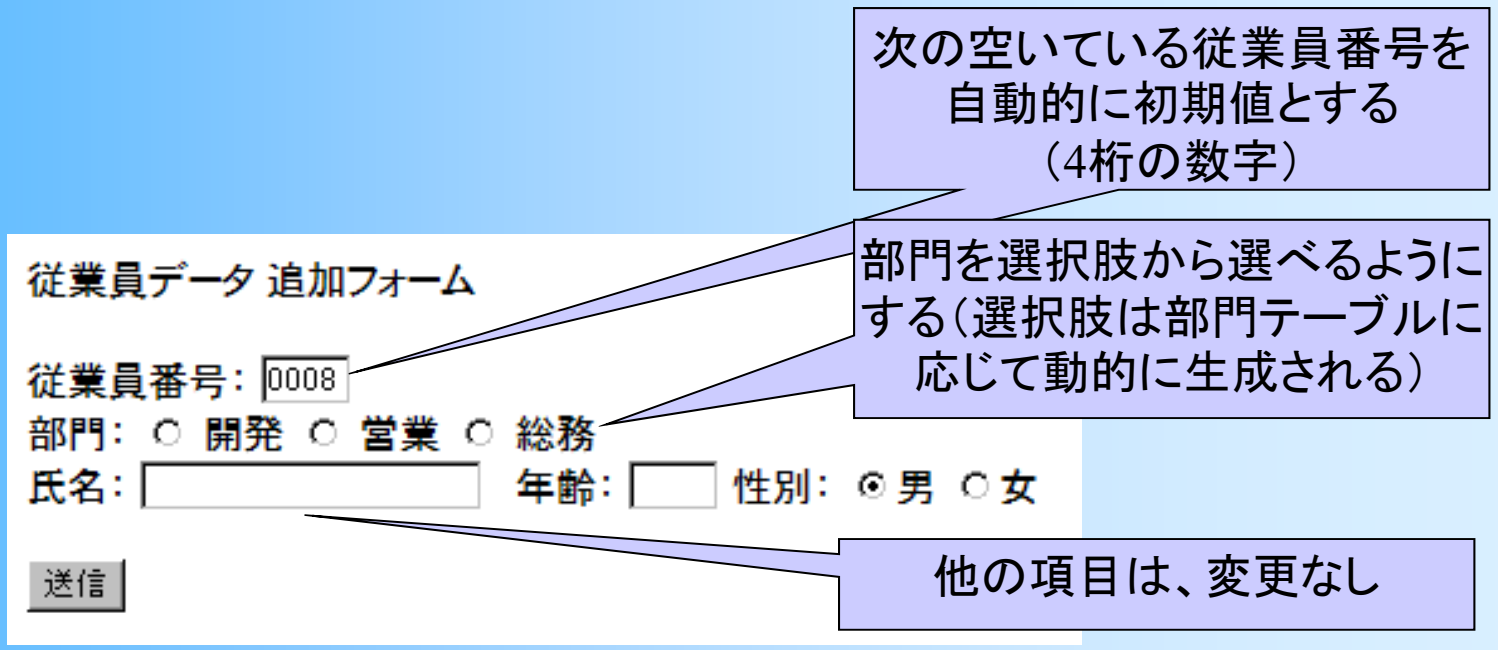
- 全てのデータを入力するのは大変、また、一部のデータは入力可能なデータが限られる
  - 例えば、部門番号には、外部参照整合性制約があるので、存在しない部門番号は入力不可能



- 適切な初期値や選択肢を表示することで、入力を簡便化したり、不適切なデータが入力されることを防止したりできる

表示結果の例

- ウェブブラウザでの表示結果



追加フォームの動的生成(1)

- 追加フォーム2(exmployee_add_from.php)
  - html ではなく php である点に注目
    - phpスクリプトを使って動的にフォームを生成する
  - 従業員番号の初期値を取得(26～48行目)
    - 最大の従業員番号 +1 を変数 $max_id に格納

```
// 最も大きな従業員番号を取り出すSQLの作成
$sql = "select max(id) from employee";

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );

// 最大の従業員番号を取得
if ( pg_num_rows( $result ) > 0 )
        $max_id = pg_fetch_result( $result, 0, 0 );
$max_id ++;
```

SQLの出力の属性値を取得
(出力は必ず1行1列のテーブルになる)

+1 加算

追加フォームの動的生成(2)

- 追加フォーム2(exmployee_add_from.php)
  - 従業員番号の初期値を設定する(44行目)
    (フォームを表示したときに、自動的に初期値が表示される)

```
// 従業員番号の初期値を指定して入力エリアを作成
print( "従業員番号:¥n" );
printf( "<INPUT TYPE=text SIZE=4 NAME=id VALUE=%04s>", $max_id );
print( "<BR>¥n" );
```

%04s が、次の引数の $max_id で置き換えられる
(%04s の 04 は、変数の値が3桁以下であれば、左に0 を加えて 4 桁の数字として表示することを表す)

追加フォームの動的生成(3)

- 追加フォーム2(exmployee_add_from.php)
  - 部門の一覧を取得し、選択肢として表示(62～75行目)
  - 残りの項目には変更はないので、php スクリプトが終わった後に html として記述(85～96行目)

追加フォームの動的生成(4)

```
// 部門一覧を取得するSQLの作成
$sql = "select dept_no, name from department";

// Queryを実行して検索結果をresultに格納
$result = pg_query( $conn, $sql );

// 検索結果の行数を取得
$rows = pg_num_rows( $result );

// 部門の数だけ選択肢を出力
print( "部門:¥n" );
for ( $i=0; $i<$rows; $i++ )
{
        $dept_no = pg_fetch_result( $result, $i, 0 );
        $dept_name = pg_fetch_result( $result, $i, 1 );
        printf( "<INPUT TYPE=¥"radio¥" NAME=¥"dept_no¥"
                VALUE=¥"%s¥"> %s </INPUT>¥n", $dept_no, $dept_name );
}
```

部門番号($dept_no)、部門名($dept_name)を取得

"(ダブルクォート)を出力するためには、前に ¥ をつける必要がある

選択肢が選択されたときに、部門番号($dept_no)を次のページに渡す値とする

選択肢として部門名($dept_name)を表示

実行結果の例

```
・・・・・・・・
従業員データ 追加フォーム<BR><BR>

<FORM ACTION="employee_add.php" METHOD="GET">

<!-- ここからPHPのスクリプト始まり -->
従業員番号:
<INPUT TYPE=text SIZE=4 NAME=id VALUE=0008><BR>
部門:
<INPUT TYPE="radio" NAME="dept_no" VALUE="01"> 開発 </INPUT>
<INPUT TYPE="radio" NAME="dept_no" VALUE="02"> 営業 </INPUT>
<INPUT TYPE="radio" NAME="dept_no" VALUE="03"> 総務 </INPUT>
<!-- ここまででPHPのスクリプト終わり -->

氏名:
<INPUT TYPE="text" SIZE="24" NAME="name">

年齢:
<INPUT TYPE="text" SIZE="4" NAME="age">
・・・・・・・・
```

従業員番号の初期値を設定

部門の選択肢を表示

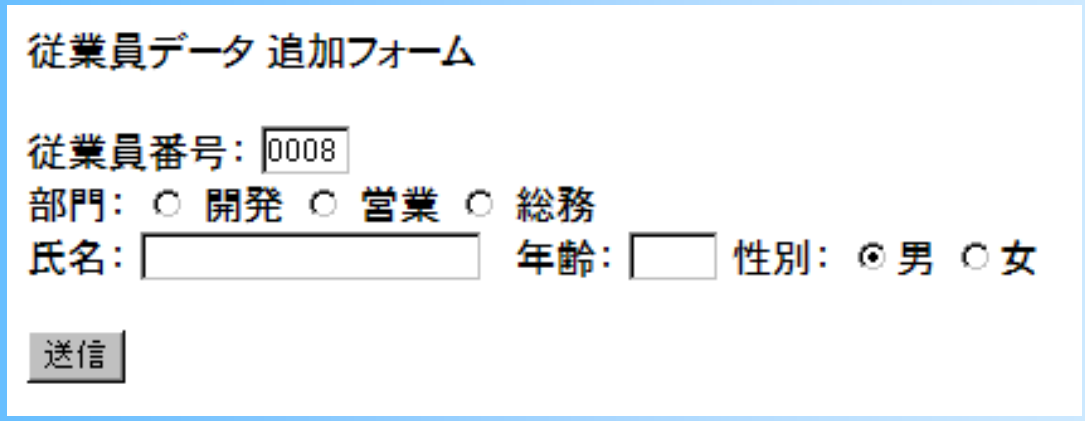
表示される部門名

以降は、html をそのまま使用

部門を選択したときに、dept_no の値として次に渡される部門番号

表示結果の例

- ウェブブラウザでの表示結果



サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

削除

- 削除フォーム（exmployee_delete_form.html）
  - 削除する従業員の従業員番号を入力
- 削除処理（exmployee_delete.php）
  - 指定された従業員番号のデータを削除
    （DELETE構文 → 各自記述）

- プログラムの中身は、これまでと同じなので、説明は省略

サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

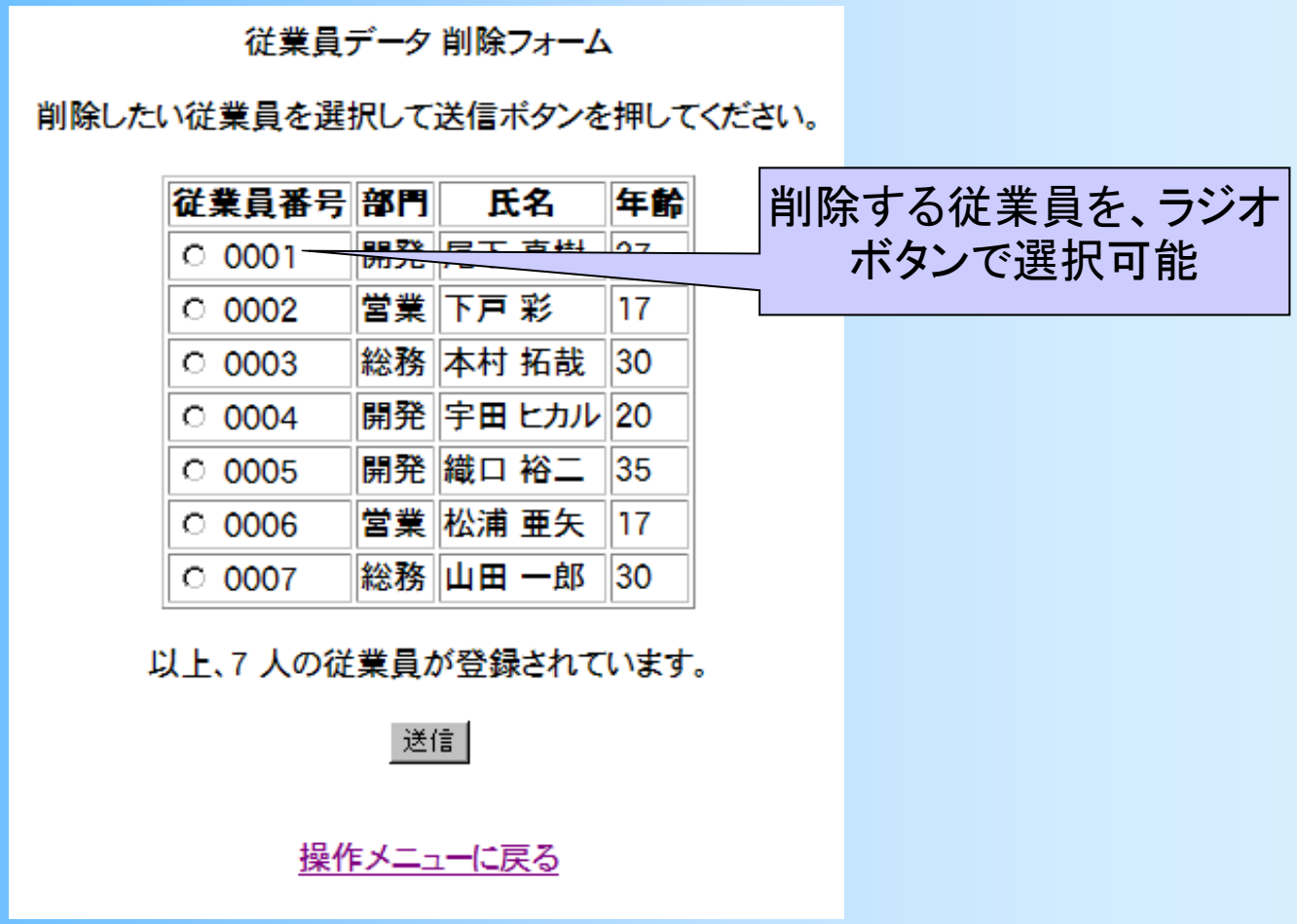
削除フォームの動的生成

- 削除指定フォーム（動的生成版）（exmployee_delete.php）
  - 従業員の一覧表示 ＋ 削除する従業員の選択ボタンの表示（64行目）

- 一覧から選べるようにすることで、非常に使いやすくなる
- プログラムの実現方法は、これまでの方法の組み合わせなので、説明は省略

表示結果の例

- ウェブブラウザでの表示結果



サンプルページの構成

- メニュー（menu.html）
  - → 一覧表示（employee_list.php）
  - → 追加フォーム（exmployee_add.html）
    - → 追加処理（employee_add.php）
  - → 追加フォーム（動的生成版）（exmployee_add_form.php）
    - → 追加処理（employee_add.php）
  - → 削除フォーム（employee_delete.html）
    - → 削除処理（employee_delete.php）
  - → 削除フォーム（動的生成版）（employee_delete_form.php）
    - → 削除処理（employee_delete.php）
  - → 更新フォーム（employee_update_form1.html）
    - → 更新フォーム（employee_update_form2.php）
      - → 更新処理（employee_update.php）
  - → 検索フォーム（動的生成）（employee_search_form.php）
    - → 削除処理（employee_search.php）

更新

- 3段階の処理になる

1. 更新指定（exmployee_update_form1.html）
   - どの従業員のデータを更新するかを指定
2. 更新フォーム（exmployee_update_form2.php）
   - 指定された従業員の現在の属性値を表示し、修正するためのフォームを表示
3. 更新処理（exmployee_update.php）
   - データを受け取って更新処理
     （UPDATE構文 → 各自記述）

検索

- 2段階の処理になる

1. 検索フォーム（employee_search_form.php）
   - どの部門（部門名）の従業員を検索するかを指定
   - 部門名の選択肢を動的に生成
2. 検索結果表示（employee_search.php）
   - 選択された部門番号を受け取って、検索結果の従業員の一覧を表示
   - 全従業員の一覧表示のプログラムを拡張することで実現

演習課題

1. 更新機能の拡張
   - 対象の従業員を選択するフォームの動的生成
   - 削除処理での従業員選択を参考に作成
2. 検索機能の追加
   - 選択した部門に所属する従業員の一覧を表示（「全ての部門」を選択すると、全従業員を表示）

- 実現方法はこれまでに学習したものと同様
- 演習資料を参考に各自作成すること

注意点

- サニタイズ
- 文字コード

サニタイズ

- 引数として受け取った文字列をSQL文に埋め込むときは、本来はサニタイズが必要
  - 悪意のある利用者が $id にSQL文を記述して実行すると、意図しない操作が実行されてしまう
    - 例：・・・ where $id= " 001; delete from employee ";（全データが削除される）
    - SQLインジェクションと呼ばれる、セキュリティホールになりうる
  - 本来は、入力をそのまま用いず、数値以外は取り除くなどの無害化処理（サニタイズ）が必要
    - 本講義の演習では、ここまでは扱わない

文字コード(1)

- 日本語の文字コードは、Shift-JIS、EUC、UTF(ユニコード) など複数ある
  - 状況に応じて適切な文字コードの使用が必要
    - 場合によっては、文字コードの変換も必要
  - ソフトウェアによっては、複数の文字コードを用いることが可能
    - 自動的に判別・変換、もしくは、手動で切り替え可能
      - 例えば、ほとんどのウェブブラウザは、文字コードを自動判別して正しく表示
    - ただし、ひとつのファイルの中に、複数の文字コードを混在させてしまうと、確実に問題(文字化け)が発生する

文字コード(2)

- 本演習で使用する PostgreSQLサーバ、paql クライアントの文字コードは UTF
  - 演習で作成する html や php ファイルも、UTF で記述する必要がある
    - データベースから取得するデータの文字コードは UTF なので、UTF で統一することが必要
      - データベースから取得したデータの文字コードを、PHPの関数を使って変換する方法もある(本講義では扱わない)
- 改行コードの違いにも注意
  - CR+LF(Windows)、CR(Mac)、LF(Unix)

演習課題(1)

前回までの演習は完了しているものとする

1. 削除処理を行なうPHPプログラムに削除処理のためのSQLを追加し、削除が正しく動作するようにせよ(exmployee_delete.php)
2. 更新処理を行なうPHPプログラムに更新処理のためのSQLを追加し、更新が正しく動作するようにせよ(exmployee_update.php)

演習課題(2)

3.更新機能で、更新する従業員をリストから選択できるように拡張したものを作成し、正しく動作するようにせよ
（exmployee_update_form1.php）

4. 検索機能として、選択された部門の従業員の一覧を表示するSQLを追加し、検索が正しく動作するようにせよ
（exmployee_search_form.php, exmployee_search.php ）

演習課題の提出

- 演習課題のテキストファイルに回答を記述して、Moodleから提出
- 提出締め切り 7月30日（月）18:00

レポート課題

レポート課題

- データベースの作成
  - 自分で決めた何らかのテーマを題材にして、データベースとWebインターフェースを作成
- Moodleから提出
  - レポート、作成したプログラム一式を提出
  - ウェブインターフェースも作成
- レポートの締め切りは後日連絡
  - 8月下旬（期末試験後）の締め切りを予定

レポート課題

- 課題内容
  - 自分で決めた何らかのテーマを題材にして、データベースとWebインターフェースを作成

  1. スキーマの設計
     - データベースに格納するデータを決めて、思いつく属性を挙げる → 正規形を満たすように正規化
  2. テーブルの作成、データの追加
  3. Webインターフェースの作成
     - 一覧表示・追加・削除・修正
     - なるべく実用的に使えるような検索機能などを追加

1. スキーマの設計

- 思いつく全ての属性を挙げて1つのリレーションとし、全ての関数従属性を列挙
  - 従業員(従業員番号、氏名、年齢、部門番号、部門名、部門代表、担当顧客番号1、担当顧客番号2、・・・、住所、電話番号)
    - ヒント: 属性値が複数になる場合は、上の例のように「・・・」などとしておき、最初に、第1正規形を満たすように、複数のリレーションに分解する
  - 候補キー
  - 関数従属性
    - 部門番号 → 部門名、・・・ → ・・・、・・・

1. スキーマの設計（続き）

- 各正規形を満たすかどうか順番に検証して、分解
  - ・・・より、第？正規形を満たす　or
    ・・・より、第？正規形を満たさないので、分解
- 最終的に得られたスキーマを示す

- 必ず最初は1つのスキーマとして、段階的に分解していくこと
  - 正規化の練習なので、最初から正規化済みの複数のスキーマを挙げているものは減点とする

2. データベースの作成

- テーブルの作成
  - 設計したリレーションスキーマをもとに、複数のテーブルを作成する
  - テーブル名、属性名は、適切な英単語（アルファベット）に変更する
- データの追加
  - インターフェースのテストに必要な最低限のデータを追加する（最低20個程度）
- レポートには、テーブルの作成に使用したsql と、データの一覧を示す

2. データベースの作成（続き）

- データベースは、これまでの演習で作成したものを用いること
  - 自分のアカウント名のデータベース
  - 勝手に新しいデータベースを作成しないこと

3. Webインターフェースの作成

- Webインターフェースを作成する
  - 一覧表示
  - 追加（フォームの動的生成に対応すること）
  - 検索（なるべく実用的な検索機能をつけること）
  - 削除、更新
- レポート
  - 全体のページの構成、各ファイルの説明（フォームから渡す引数、フォームの動的生成の方法、検索処理で使用しているSQL文、など）を必ず書くこと
  - どのようにしてインターフェースを実現しているかが分るようなレポートを作成する（インターフェースができていても、説明が不十分であれば、大幅に減点となる）

レポート課題に関する注意(1)

- くれぐれも十分早くから準備を始めること
  - 締め切りの直前になって始めて、間に合わない人がいる
  - 学期末には、他の科目のレポートも重なるので、まとめてやろうとしても間に合わない
  - 1ヶ月程度の期間を与えて課題を出しているので、きちんと計画的に課題に取り組むこと
  - 少なくとも、締め切りの１週間前には課題の内容は終らせて、残りの時間はレポート作成に使った方が良い
  - 何か質問や相談等があれば、早めに申し出ること(締め切り直前になって来ても間に合わない)

レポート課題に関する注意(2)

- できるだけ工夫をすること
  - 選択肢の動的生成、高度な検索インターフェース、など
  - 実用的なデータベースであれば、高評価
  - こういった自由度の高い課題で、どれだけ工夫できるかが、非常に重要（応用能力や自己PR）

レポート課題に関する注意(3)

- レポートをきちんと書くこと
  - どのようにして処理を実現しているのかが、きちんと分かるように書く(きちんと文章で説明できることも重要)
  - 見出しや段落分け、適切な余白・行間、引用は枠で囲むなど、レポートの見やすさも重要

レポート課題に関する注意(4)

- 不正行為は絶対にしないこと
  - たとえ一部でも、他人のプログラム・レポートを丸写しした場合は、不正行為となり、厳重に処罰・減点される

よくある間違い(1)

- 課題1 .スキーマの設計
  - キー属性・関数従属性がきちんと書かれていない、もしくは、間違っている
  - 正規化の間違い(正規形の判定・分解)
- 課題3. Webインターフェースの作成
  - 選択肢の動的生成に対応していない
  - 検索機能がない、もしくは、ほとんど意味のない検索機能しかない
  - サンプルプログラムの一部がそのまま残っている(タイトルなど)

よくある間違い(2)

- レポートの書き方
  - 説明不足のレポートが多い
    - 処理の実現方法の説明、使用した変数やSQLの説明、ファイル構成の図、など
    - 作成結果だけではなく、何故そのようなプログラムを作成したのか、理由の説明が必要
  - レポートの見易さ・読みやすさ、レポートの体裁
    - 適切な余白・行間、章分け・段落分け、インデント、本文と引用プログラムの区別、誤字脱字、など
- 詳細は、レポート課題の説明を参照すること

演習環境（CL以外での演習）

- 基本的にCLの端末（Windows環境）で演習を行う
- 他の端末室や自宅での演習も可能
  - Moodleの演習補助資料を参照
    - 情報科学センタ端末（Linux環境）やマルチメディア教室（Windows環境）での演習
  - データベース操作（psql）やファイルアップロードの方法が環境によって異なるため注意
  - 文字コード・改行コード等の問題にも注意

まとめ

- 前回の復習

- PHPによるインターフェース開発（2）
- レポート課題

次回予告

- データベースシステム内部で用いられる技術
  - 物理的データ格納方式
  - 同時実行制御
  - 問い合わせ処理の最適化
  - 障害回復

- 物理的データ格納方式
  - データ格納方法
  - データアクセス方法