データベース 2009 #3

久野　靖*

2009.10.22

はじめに

前回、データベース設計の話をしましたが、全部一気にやると多いのと、多少演習して頂いてからの方が分かると思ったので、途中までにしてありました (そうは書いてなかったけど)。今回その残りをやります。

その後が今回の本題で、課題としてデータベース連携 Web アプリケーションを作るので、Web 上のスクリプト言語 PHP を用いて簡単なデータ操作を行う練習から始めてみます。その後は実際の書籍データを用いてデータベースからの検索部分だけを作ってみるので、その先の工夫を考えておいてください。

1　データベース設計の概要 (つづき)

1.1　エンティティの抽出

QUIZ: 前回の宿題として IDEF1X を作成してみて、どういうことが問題でしたか?
QUIZ: IDEF1X で何と何をエンティティとして描くかはどうやって決めましたか?

トップダウンアプローチでは特に、エンティティとして何と何を選ぶか、それらをどう関連づけるかが設計のノウハウとして大切です。具体的にはどうすればいいでしょう。

通常、第 1 段階として、リソース (資源) 系のエンティティをまず特定します。リソースとは具体的なもの (書籍とか顧客とか) に対応するものなので、イメージしやすいからです。

やり方としては、自然言語による記述、業務モデル、データフロー図などからリソースに対応すると思われるものを候補としてリストアップします。前回課題としたネット書店では、たとえば次のものが**リソース系エンティティ**の候補になると思われます。

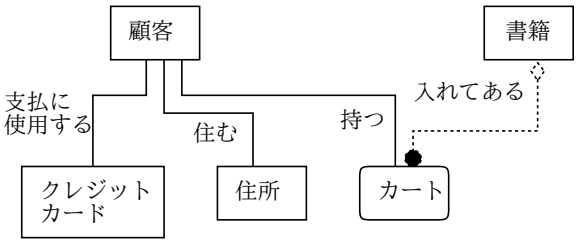

図 1: ネット書店のリソース系エンティティ

- 顧客 — 作成する Web サイト上で書籍を購入する個人
- 書籍 — 販売対象となる本

*経営システム科学専攻

- 住所 — 本の配送先
- クレジットカード — 代金を請求するのに使用するカード
- カート — 選んだ本を入れておく買物かご

第 2 段階として、リソース系エンティティが抽出できたらそれらの間の関係を考え、依存エンティティかどうかも含めて IDEF1X で記述してみます。ここではまだ詳しい分析はしていないので、エンティティ名のみを記します。カートは顧客がいなければ意味がないので依存エンティティと考えてみました (図 1)。書籍とカートの関連は必須でなく、どのカートにも入っていない書籍もがあるので◇つきの点線としました。あとは 1 対 1 です。

第 3 段階として**イベント系のエンティティ**をこれに加えます。イベント系のエンティティとは、注文とか出荷など、「もの」ではなく「起こること」に対応するエンティティです。ここでは次のものがあると考えます。

- 注文 — 顧客による注文
- カード請求 — カード会社への代金請求
- 発送 — 商品の発送

第 4 段階として、イベント系エンティティの間の関連も IDEF1X で記述してみます。ここでは注文、カード請求、発送とも独立エンティティとしました (図 2)。これらの関係は、すべて 1 対 1 ではありますが、ただし注文が先にあって、それに基づいて後からカード請求、発送が起こるため、点線で結び「1 つだけ対応」の数字を記入しています。

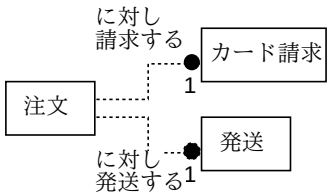

図 2: ネット書店のイベント系エンティティ

第 5 段階として、リソース系とイベント系のエンティティの関連を検討し、2 枚の図を 1 枚に統合します。このとき、前に描いた図の不都合に気がつくこともありますから、その場合は遡って修正していきます。ここでは手戻りはなかったのですが、この程度のものでもかなりごちゃごちゃになりました (図 3)。本ものの業務でははるかに大変なので、図を描いて整合性をチェックしたり、最後は SQL のスキーマ定義 (CREATE TABLE のことです) を自動生成してくれたりするツールを使うのが普通でしょう。

この後第 6 段階として、業務データから有益な分析を行うためのサマリーを保存するためのエンティティを追加することがありますが、それについてはデータウェアハウスと関連していますので後日取り上げる予定です。

1.2 キーと主要属性の定義

ここまででではエンティティの中身は考えずやってきましたが、次の段階としてそれぞれのエンティティの内容を検討します。具体的には、エンティティごとに次のものを検討していきます。

- どのような属性があるか
- そのうち何を主キーとするか
- 他のエンティティを参照する外部キーはどれか

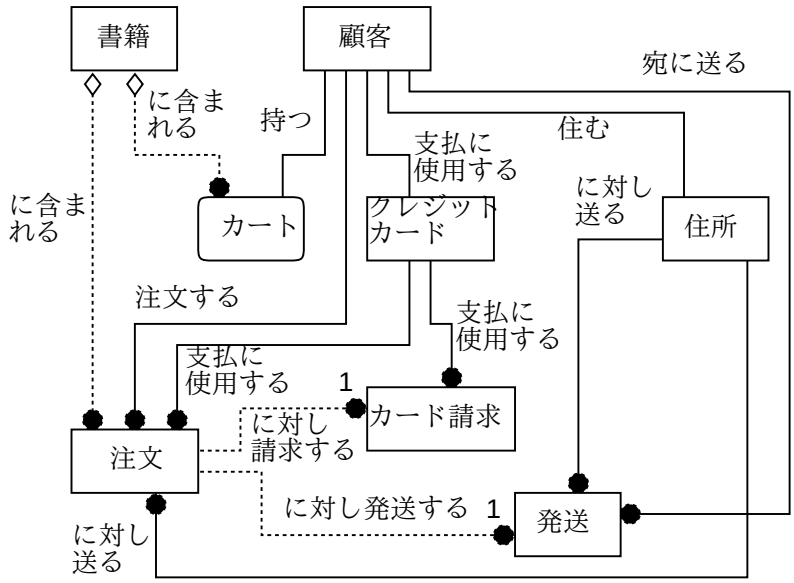

図 3: 全エンティティを含む関連図

とりあえず、書籍と顧客関係のエンティティは次のようにしました。

書籍 — これは問題で規定されている通りでよい。ISBN が主キー、属性は題名、著者、出版者、発行年月、価格、種別 (文庫、新書、単行本)、順位。

顧客 — ここでは「メールアドレスをもらって、それを ID として使う」ことにした。メールアドレスは一意だからこれを主キーとする。残る属性として、パスワード、氏名、かな氏名を入れる。

住所 — 顧客を参照するためにメールアドレスを入れ、これが一意なので主キーともする。あとの属性は電話番号、郵便番号、都道府県、市町村区、町名番地とする。メールアドレスで顧客を参照する。

クレジットカード — これも顧客を参照するためにメールアドレスを入れ、これが一意なので主キーともする。あとの属性はカード会社、カード番号、カード氏名、更新年月。メールアドレスで顧客を参照する。

カートはどうしようかと考えたのですが、簡単のため「1 つのカート項目に持ち主と 1 種類の本のデータを保持し、ある顧客の (1 つだけある) カートとは、自分が持ち主であるカート項目の集まりである」と考えました。(とすると、図 3 等には修正が必要ですが、どこだか分かりますか?)

カート — メールと ISBN の組が主キーで、これらはそれぞれ顧客と書籍を参照する。あとの属性は数量のみ。

次は注文ですが、これは少し面倒です。とりあえず属性を書いて見ましょう。

注文' 注文番号という一連番号を生成しこれを主キーとする。属性は日付、メールアドレス、金額、状態 (確認待ち、発送待ち、発済み)、そして「ISBN と数量の組み」が N 個。

しかし、このように 1 つの組の中に繰り返しがあっては RDB に載せられません (最近の RDB ではこのようなものをサポートすることもありますが…)。そこで、繰り返しを分解するため「注文」と「注文明細」を分けます。「注文明細」は 1 項目について特定注文の 1 つの書籍の行に対応します。

注文 注文番号という一連番号を生成しこれを主キーとする。属性は日付、メール、金額、状態 (確認待ち、発送待ち、発済み)。メールは顧客を参照。

注文明細 注文番号と ISBN の組を主キーとする。加えて数量を持つ。注文番号は注文を参照し、ISBN は書籍を参照する。

この部分の構造を図 4 に示します。皆さんも自分の設計を「すべてのエンティティについて」図示してみてください。

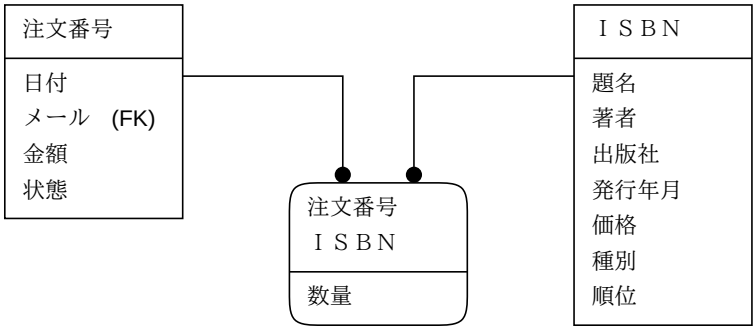

図 4: 注文明細と注文

あとはカード請求と発送ですが、これらは簡単です。

カード請求 注文番号を主キーとする。属性は日付とメールで、注文番号は注文、メールはクレジットカードを参照する。

発送 注文番号を主キーとする。属性は日付とメールで、注文番号は注文、メールはクレジットカードおよび顧客を参照する。

発送では顧客まで参照としているのは、名前はエンティティ顧客に入っていてこれを必要とするからです。

なお、計算機科学基礎 II で説明しましたが、スキーマの中の繰り返し部分を分解したりしてテーブルを正規形に変換することを**正規化**と呼ぶのでしたね。次の 2 つは覚えておきましょう。

第 1 正規形 — 表のどの属性も、複数のデータが組合わさったり並んだりしたものではないこと。

第 3 正規形 — 表のどの属性も、主キーのみに依存して決まること (つまり、主キー以外の属性に依存していたり、主キーの一部分だけに依存して決まってはいけない)。

上では第 1 正規化を行いましたが、チェックしたところこの状態で第 3 正規形となっていることが分かりました。なお、これも計算機科学基礎で説明しましたが、正規化は絶対条件ではなく、実務上のデータベース設計では性能上や利便上の都合から第 3 正規化されていないスキーマを使用することもあります。

1.3 データ設計/データベース設計のまとめ

前回と今回でデータ設計/データベース設計という作業をごく大まかに見て来ましたが、これらが「何を」意味しているかは一応お分かり頂けたと思います。しかし、実際に大きなシステムのデータ設計やデータベース設計を行うのは込み入った問題であり、そのための細かいプロセスやサポートツールも色々あります。ここで紹介したのは大まかな概要だということを頭に置いて、必要ならさらに勉強してみてください。

2 PHP による Web アプリケーション

2.1 復習: PHP の概要

PHP については「計算機ソフトウェア」で取り上げましたが、既に忘れていると思うので、簡単に復習しておきましょう。まず、PHP プログラムは普通のテキストファイル (通常は HTML ファイル) の中に次のような形 (**処理命令**と呼ばれるものの一種) として埋め込むのでした。

```
<?php …PHP プログラム… ?>
```

PHP プログラムの部分は何行に渡っても構いません。PHP プログラムの中では C や C++や Java と同じコメント (/* ... */」で囲むコメント、または「//」(「#」でもよい) から行末までのコメントが使えます。

もう 1 つ重要なことですが、PHP プログラムの中で HTTP ヘッダを出力する機能 (header() 関数) を使うときは、一切の通常出力を行う前に行う必要があることです。たとえば、PHP で日本語が正しく表示できるように文字エンコーディングを指定するときにも header() が必要です。なので、PHP プログラムをサーバ上に置く時は次の形のものを使うとよいでしょう (このファイルを/u1/kuno/work/base.php に置いておくので、適宜コピーして利用してください)。

```
<?php
header("Content-type: text/html; charset=euc-jp");
/* その他ヘッダ出力はここで */
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>title...</title>
<style type="text/css">
/* CSS 記述はここに */
</style><head><body>
<?php
/* ページ本体出力を含む動作 */
?>
</body></html>
```

ファイル中に日本語を含める場合は、ファイルの文字コードは EUC にしてください。Emacs であれば「Ctrl-X [RET] f euc-jp [RET]」を実行することで編集中のファイルの文字コードを EUC にできます。ステータス表示の最後が「…E」となっていれば EUC ですので、そうなっていない時だけ上記を実行すればよいでしょう。

前に紹介したと思いますが、簡単なサンプルを再掲します (図 5)。

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { margin: 2px 20px; background: rgb(200,215,255) }
</style></head><body>
<?php
  for($i = 0; $i < 10; ++$i) {
    echo "<div class='test'>番号{$i}です。</div>";
  }
?>
</body></html>
```

資料では見やすいように下げてありますが、実際のファイルでは最初の「<?php」より前に 1 文字の空白もあってはいけません (あるとページ内容が出力されてしまい、header() が失敗します)。

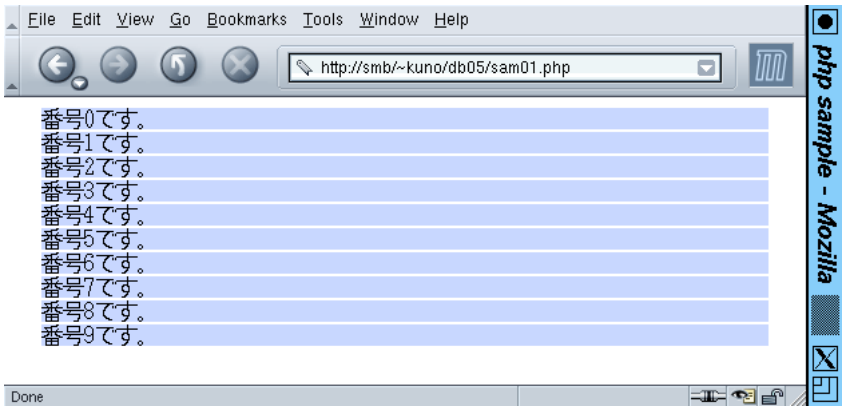

図 5: PHP のループでページ出力

2.2 変数とデータ型

PHP では (Perl などと同様)、変数は先頭に「$」をつけた名前で表します。PHP の変数には型がなく、任意のデータを格納できます。データの種類としては次のものがあります。

- 論理値 (boolean) — 真偽値であり TRUE と FALSE という定数も用意されている。
- 整数 (integer) — 整数値。
- 浮動小数点数 (float, double) — 小数点以下の部分を含む値を表現できる形式の数。
- 文字列 (string) — 文字の並び。
- 配列 (array) — 複数の値が並んだもの
- オブジェクト (object) — PHP5 で加わったオブジェクト指向機能によって作成されたオブジェクト
- リソース (resource) — ファイルやデータベース検索結果など外部とのやりとりに使うデータ
- NULL — 「何もない」ことを表す特別な値。NULL という定数もある。

なお、PHP の中核部分では変数や定数の大文字小文字は区別されません。

文字列リテラルは「'...'」でも「"..."」でも囲めますが、ダブルクォートでは変数を書いておくとその値が埋め込まれます。なお、先の例題では埋め込みでもよかったのですが、「.」(文字列連結演算子) の方を使いました (埋め込みを使う場合は、「…{$i}…」のように「{}」で囲んでどこまでが変数の範囲なのか指定する必要がある場合が多いので、例では常にそう書くようにしています。

配列は関数 array() を使って「array(値, 値, ...)」で作り出すことができますし、上記で値を 1 個も指定しないで空の配列を作り、直接添字を指定して「$a[添字] = 値」により値を入れることもできます (もちろん添字を指定して特定の要素を読み出すこともできます)。要素の個数に制限はなく、また添字には文字列などを使うこともできます (連想配列)。

2.3 演算子

PHP の演算子は C などと共通のものが多いです。主要なものを強さの (かっこ無しで書いた時に結び付きの強い) 順に挙げておきます。

- !、=~、++、--、(型名)、@ — 論理否定、ビット反転、増加、減少、キャスト、エラー無視。[1]
- *、/、% — 乗算、除算、剰余

[1] @は関数呼び出しなどの前につけることでエラーメッセージを抑制させられる。

- +、-、. — 加算、減算、文字列連結
- <、<=、>、>= — 大小比較
- ==、!=、===、!== — 等しい/等しくない。後の 2 つは型変換しないで比較する。たとえば「1 == TRUE」だが「1 === TRUE」ではない
- && — 「A かつ B」(論理積)
- || — 「A または B」(論理和)
- =、+=、-=、*=、/=、%= — 代入および代入演算 (足し込む等)

2.4 制御構造

PHP の制御構造はほぼ C と同じなので、C にない foreach と echo 以外は説明は略します。

```
if(条件) 文 [ else 文 ] --- if 文
while(条件) 文 --- while 文
for(式; 式; 式) 文 --- for 文
foreach(配列 as 変数) 文 --- foreach 文その 1
foreach(配列 as 変数 => 変数) 文 --- foreach 文その 2
switch(式) 文 --- switch 文
{ 文… } --- ブロック
break; --- ループから抜け出す
continue; --- ループの次の周回へ
echo 文字列; --- 文字列の出力。式; --- 代入や関数呼び出し
```

foreach ですが、これは配列の全要素を順番に処理するのに使います。その 1 では値を順番に指定した変数に入れながらループします。その 2 は添字と値をそれぞれ 1 番目と 2 番目に指定した変数に入れながらループします。

echo については、普通の関数呼び出しでも済みそうに思えますが、PHP では特別扱いの命令になっています。というのは、PHP では文字列を出力するのが非常に重要な機能だからです (<?php ... ?>の外側にある HTML なども実はすべて echo によって出力されていることに注意)。

なお、制御構造より外側のプログラム構造ですが、PHP では先の例題にもあるように、C などと違い、一番外側レベルに書かれた文を順番に実行するのが main に相当します。関数やクラスは次の形により定義します。

```
function 関数名 ( 引数名,... ) { 文… }
class クラス名 { クラス本体 }
```

関数定義の箇所を通ると関数が定義されるので、通る前に関数を呼ぶことはできません。if 文などを使って枝別れの中で関数を定義することもできます。クラスとオブジェクト指向機能については話が長くなるのでここでは説明しません。

2.5 フォームを用いた対話的ページ

WWW における**対話的ページ**とは、ユーザからの入力を受け取ってそれに応答するようなもの全般を指します。対話的ページが動くためにはそれを動かすプログラムが必要ですが、それらはプログラムが動く場所により次の 2 通りに分類されます。[2]

[2]最近では両者を複合させた **Ajax**(ブラウザ上の JavaScript プログラムがサーバ上のプログラムと通信して必要なデータを取り寄せ画面に反映する技術全般をいう) なども現れています。

- クライアント側プログラミング — ブラウザ上で動くプログラムによる対話的ページ。応答が速い、細かい画面操作やブラウザ操作ができるなどの特徴がある。JavaScirpt、Flash、Java Applet など。
- サーバ側プログラミング — サーバ上で動くプログラムによる対話的ページ。サーバ上のリソース (典型的にはデータベースやファイル) にアクセスできるという利点がある。CGI(言語は何でもよいが Perl が多い)、ASP、Java Servlet、JSP、PHP などが代表的。

以下ではもちろん、PHP を用いたサーバ側プログラミングを前提に説明します。サーバ側プログラミングの場合、ブラウザ画面からサーバにデータを送信するのには HTML の**フォーム機能**を使うので、まずフォームを作るための HTML 要素について説明しましょう。まずは全体を囲む form 要素から。

- `<form method="post" action="#"> ... </form>` — フォーム全体は form 要素で囲まれる必要がある。form タグでは、データの送信方法や送信先 URI(データを処理するサーバ側プログラムのありか) を指定する。ここでは PHP を使うので、フォームを生成する URI とデータを処理する URI は同一 (同じプログラム) になるので「`action="#"`」を指定しておけばよい。送信手法はデータが URI にくっついて見える「`"get"`」と別途送られる「`"post"`」があるが、通常はデータが見えない方がいいので後者を選ぶ。

form 要素の内側には、さまざまな入力部品 (HTML 用語ではコントロールと言います) を入れます。入力部品は通常のテキストや HTML 要素と混ぜて入れられるので、通常の HTML の機能を使って説明文やラベルをつけたり、見やすく配置することができます。

入力部品は以下で説明するようにさまざまな種別がありますが、すべてに共通するのは `name` 属性 (名前) と `value` 属性 (値) です。これらは、サーバ側にフォームが送られる時に対になって送られます。たとえば「`name="age" value="30"`」という属性だったら、「age:30」という対が送信されるわけです。なお、値は部品によってはユーザが入力した文字列になります。

以下に主要な (以下の例題などで使用する) 入力部品とその属性指定を説明します (`name` と `value` は上記の通りなので特に注記することがない場合は説明していません)。

- `<button name="名前" value="値">...</button>` — 送信送信ボタン。要素の内側部分がボタンの表示内容になる。ボタンが押されるとフォームのデータがサーバに送信される。送信ボタンは複数あってよいが、送信のために押されたボタンの名前と値だけがサーバに送られる。
- `<input type="text" name="名前" [size="長さ"] [value="値"]>` — テキスト入力欄。送信時に欄に入っている文字列が値として送られる。`size` で欄の幅 (文字数単位) を指定できる。`value` で最初に入っている文字列を指定可能。
- `<input type="password" name="名前" [size="長さ"] [value="値"]>` — 上と同じだが、打ち込んだものが見られないように「*」で表示される。ただし、データそのものは暗号化されるわけではないので、本当に盗まれるとまずい情報は暗号化通信を使ったページから送信すること。
- `<textarea [rows="行数"] [cols="文字数"]> ... </textarea>` — 複数行入力欄。機能は複数行入れられること以外は上と同じ。要素の内側部分が最初の表示内容になる。`rows` と `cols` で大きさを指定できる。
- `<input type="checkbox" name="名前">` — チェックボックス。画面上でチェックを ON/OFF でき、送信時にチェックされているものだけが「`"on"`」という文字列を送信します。
- `<input type="radio" name="名前" value="値" [checked]>` — ラジオボタン。同じ名前のものが複数あってよく、その中でどれか 1 つだけが ON になる。最初に ON であって欲しいものがあれば、それ 1 つだけに `checked` を指定する。送信時には ON になっているものの値が送信される。
- `<input type="hidden" name="名前" value="値">` — 特別な部品で、画面に表示されない (従ってユーザが値を変更することもない)。常に指定された名前と値を送信する。

- `<select name="名前">...</select>` — 選択メニュー。内側には次に示す option 要素を複数入れられ、これらを項目とするメニューができる。
- `<option [value="値"] [selected]> ... </option>` — 選択メニューの 1 項目。この項目が選択されている時は選択メニューの値としてこの項目の `value` が送られる。ただし `value` を省略しているときは要素の内側の内容 (項目として画面に表示される) が送られる。最初に選択された状態であってほしい項目には `selected` を指定する。

2.6 PHP とデータベースの連携

ではページからのデータ入力を復習し終わったところで、いよいよ PHP とデータベースを組み合わせる方法について説明します。大部分復習ですが、`pg_escape_string()` だけは新しい内容です。なお、PHP では複数の DBMS との接続をサポートしていますが、ここではもちろん PostgreSQL との接続について説明します (が、他の DBMS でも同様と思ってよいです)。

PHP から PostgreSQL の機能を呼び出すには、次の関数だけ知っておけば十分です。

- `pg_connect(`オプション指定文字列`)` — PostgreSQL サーバと接続し、接続オブジェクト (以下単に「接続」と記す) を返す。接続が失敗した場合は FALSE を返す。オプション指定文字列では色々な指定ができるが、とりあえず「`"dbname=`ユーザ名`"`」という指定のみで十分。なお、我々の環境では PHP は nobody というユーザ ID で動作しているので `pgsql` で「`grant all on` 表名 `to nobody;`」を実行して PHP が扱う表のアクセス許可を与えておくこと。
- `pg_query(`接続，実行文字列`)` — 任意の SQL 文を文字列として受け取って実行し、結果オブジェクト (以下単に「結果」と記す) を返す。実行が失敗した場合は FALSE を返す。
- `pg_num_rows(`結果`)` — 結果の中に何個の組が含まれているかを返す。
- `pg_fetch_row(`結果`)` — 結果から 1 つ組を取り出して配列として返す。繰り返し呼ぶと 1 つずつ組を取り出すことができ、終わりになったら FALSE を返す。
- `pg_escape_string(`文字列`)` — 文字列を受け取り、その中の特殊文字をエスケープする。なぜこれが必要かというと、入力文字列の中に「SQL コマンドをいったん終了させ、別のコマンドを実行する」ような仕掛けを組み込むことでデータベースの内容を任意に操作されることを防ぐため。つまり、フォームなどでユーザから入力されてきた情報は「すべて」この関数で変換してからでなければ `pg_query()` に渡してはならない。この関数はコマンドの区切りとなる「;」など「危険な」文字にエスケープ文字を付加して普通の文字として扱わせるように加工し、上記のような危険を抑止する。

なお、`pg_query()` はその名前にも関わらず、クエリだけでなく任意の SQL 文を実行できます。

2.7 例題: 名簿データの登録

では、ごく簡単な例として、insert 文を使ってデータベースに情報を登録するページを作って見ましょう (図 6)。

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { padding: 4px; background: rgb(180,215,250) }
```

図 6: 名簿データの登録

```
td, th { padding: 5px }
</style></head><body>
<h1>名簿入力フォーム</h1>
<form method="post" action="#"><div class="test">
<table>
<tr><th>氏名:</th><td><input type="text" name="shimei"></td><tr>
<tr><th>生年:</th><td><input type="text" name="seinen" size="4"></td><tr>
<tr><th>月:</th><td><input type="text" name="gatsu" size="2"></td><tr>
<tr><th>日:</th><td><input type="text" name="hi" size="2"></td><tr>
<tr><th>郵便番号:</th><td><input type="text" name="yuubin" size="8"></td><tr>
<tr><th>住所:</th><td><input type="text" name="juusho" size="40"></td><tr>
<tr><th></th><td><button type="submit">登録</button></td>
</table>
<?php
  if($_POST['shimei'] == '') {
    exit;
  } else if(!($conn = pg_connect("dbname=kuno"))) {
    echo "<p>データベース接続できません。管理者に連絡を。</p>";
  } else {
    $shimei = pg_escape_string($_POST['shimei']);
    $seinen = pg_escape_string($_POST['seinen']);
    $gatsu = pg_escape_string($_POST['gatsu']);
    $hi = pg_escape_string($_POST['hi']);
    $yuubin = pg_escape_string($_POST['yuubin']);
    $juusho = pg_escape_string($_POST['juusho']);
    $res = pg_query($conn, "insert into 名簿 values("
      ."'{$shimei}',{$seinen},{$gatsu},{$hi},'{$yuubin}','{$juusho}')");
  }
?>
</div></form></body></html>
```

このPHPプログラムは前半部分が入力するデータを受け取るための入力欄を集めたフォームであり、後半に入力処理を行うPHPのコードがくっついています。ただし、入力処理が行われるのは実際にデータがあるときだけで、データがない場合 (たとえば初回) などは処理は何もしません。データがある場合はその各データを pg_escale_string() でチェックしてから変数に取り出し、それらの値をまとめた insert 文でデータベースに挿入しています。なお、名簿テーブルの設計は次のようにしました。

```
  Column  |         Type          | Modifiers
----------+-----------------------+-----------
 氏名     | character(12)         |
 生年     | integer               |
 誕生月   | integer               |
 誕生日   | integer               |
 郵便番号 | character(8)          |
 住所     | character varying(20) |
```

演習 1 このPHPプログラムをコピーしてきて動かせ。

1. psql に入り「create table 名簿 (...);」でテーブルを作成する。
2. さらに「grant all on 名簿 to nobody;」でこのテーブルをPHPから扱えるようにする。
3. /u1/kuno/work/php02.php を自分のWWWディレクトリにコピーしてくる。ファイルの名前はつけ変えてもよい。
4. ファイル中の「dbname=kuno」の「kuno」を自分のユーザ名に変更して保存する。
5. 「chmod a+r ファイル名」でそのファイルをWWWサーバに読めるようにする。
6. ブラウザでそのPHPファイルを開く。

演習 2 次のことを行え。

a. 入力の形をもっとやりやすくしてみよ。たとえば生年/月/日を1行にするなど。
b. 月を選択メニュー (HTMLのselect) にしてみよ。
c. このままではデータベースに色々ヘンな値が入る可能性がある。実際に入れてみよ。また、ヘンな値が入らないようにするためには、どのようなチェックがあり得るか思い付く限り列挙せよ。
d. そのほか、入力をやりやすくするには、どのような工夫があり得るか検討してみよ。

2.8 例題: 書籍検索

次に、書籍データの中から特定の文字列をタイトルに含むものだけを表示させるという例を示しましょう (図7)。書籍データは課題用に作成したものを使用します。

```
<?php
header("Content-type: text/html; charset=euc-jp");
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>php sample</title>
<style type="text/css">
div.test { padding: 4px; background: rgb(255,215,200) }
td { margin: 2px 4px; padding: 4px; background: rgb(200,215,255) }
</style></head><body>
<form method="post" action="#"><div class="test">
```

```
<p>検索語:<input type='text' name='word'>
<button name="cmd" value="calc">検索</button>
<?php
  if($_POST['word'] == '') {
    echo "<p>検索キーワードを入れてください。</p>";
  } else if(!($conn = pg_connect("dbname=kuno"))) {
    echo "<p>データベース接続できません。管理者に連絡を。</p>";
  } else {
    $word = pg_escape_string($_POST['word']);
    $res = pg_query($conn, "select isbn, 題名, 著者 from 書籍"
      ." where 題名 like '%{$word}%'");
    if(pg_num_rows($res) > 0) {
      echo "<table><tbody><tr><th>ISBN</th><th>題名</th><th>著者</th></tr>\n";
      while($a = pg_fetch_row($res)) {
        echo "<tr><td>{$a[0]}</td><td>{$a[1]}</td><td>{$a[2]}</td></tr>\n";
      }
      echo "</tbody></table>";
    } else {
      echo "<p>検索結果が空でした。</p>";
    }
  }
?>
</div></form></body></html>
```

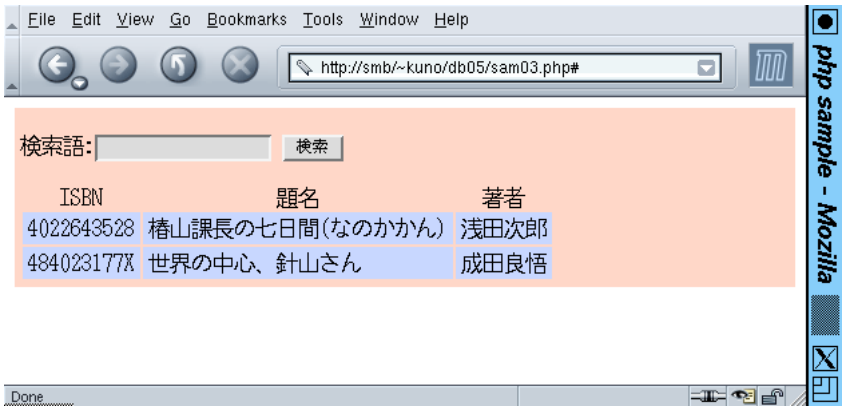

図 7: 簡単な書籍検索

このプログラムの概要は次の通りです。

- 冒頭部分はこれまでと変わらない。
- このページではフォーム部分は PHP の処理によって変化しないので、冒頭にフォームをまず書いてしまっている。
- 続いて、検索語が空かどうかをチェックし、もし空なら (これには最初にページを表示させた場合も含む)「キーワードを入れてください」いう内容だけのページとする。
- そうでなければ、次は PostgreSQL に接続するが、接続できなければエラー表示のみ行う。

- それ以外の場合は、検索語を安全に処理して word に入れ、その word を題名のどこかに含む本を検索する。
- 結果の組が 0 より多ければ、HTML の表を生成する。0 であれば「空でした」というメッセージを表示する。

このように、ごく簡単に DBMS が呼べるのも PHP の特徴です。多少込み入った検索でもだいたいは SQL 側の記述で処理できますから、PHP 側ではこのように結果を順次表示するだけで済むことが多いわけです。

演習 3 上の例題をコピーしてきて動かせ。

1. `grant all on 書籍 to nobody;`」でこのテーブルを PHP から扱えるようにする (前回の書籍データを入れた表の名前が「書籍」でないならそれにとりかえる)。
3. `/u1/kuno/work/php03.php` を自分の WWW ディレクトリにコピーしてくる。ファイルの名前はつけ変えてもよい。
4. ファイル中の「dbname=kuno」の「kuno」を自分のユーザ名に変更して保存する。
5. 「`chmod a+r` ファイル名」でそのファイルを WWW サーバに読めるようにする。
6. ブラウザでその PHP ファイルを開く。

演習 4 さらに、自分なりに検索ページを工夫してみよ。自分のネット書店ではどのようにしたいかきちんと考えること。

演習 5 上記の書籍検索結果について「その本を数量指定して注文できる」ようにするにはどうしたらいいか考えてみよ。もしできれば、注文をデータベースに記録できるようにしてみるとよい (まだカートとかを設計していないので、とりあえず ISBN と冊数のデータを入れる適当なテーブルを用意してそこに insert するなどでよい。