

情報システムと Web 技術 # 4: DB 連携 Web アプリケーション

久野 靖*

2015.12.8

はじめに

前回、データベース設計の話をしました。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。

後半ではいよいよ、PHP プログラムと DBMS を連携させ、データベースのデータを読み書きする方法について取り上げて行きます。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 再掲: ネット書店のデータフロー図

ネット書店の DFD を図 1 に再掲しておきます。宿題は、この中に出て来るデータベースの項目を考えてくること、でしたね。

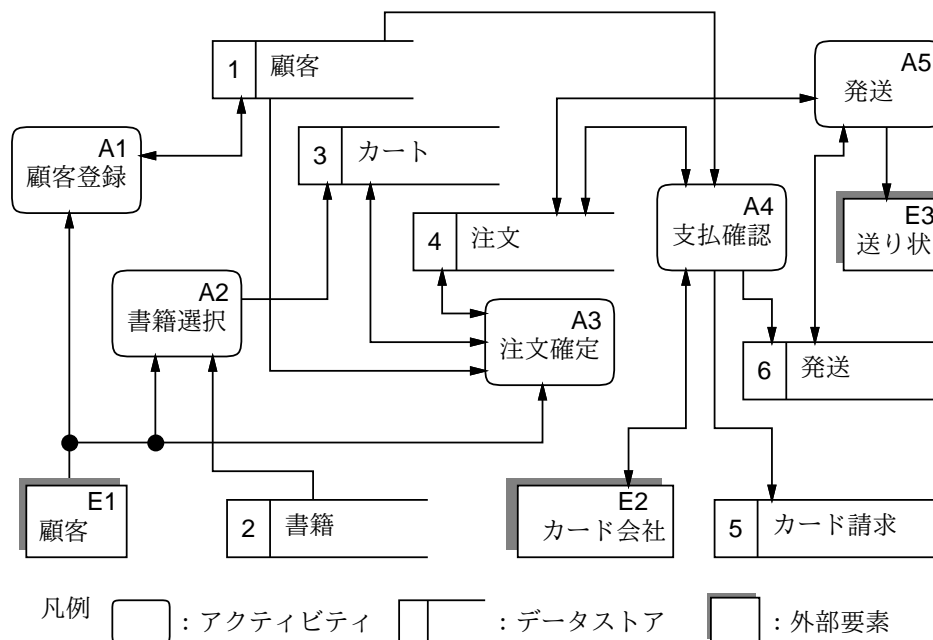


図 1: ネット書店の DFD 記述

1.2 エンティティの抽出

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考えられます。

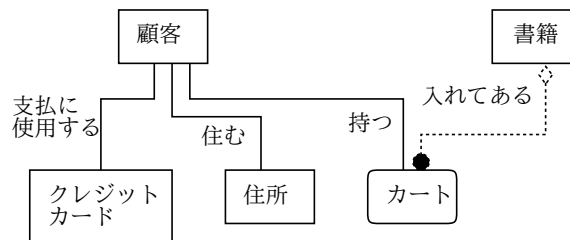


図 2: ネット書店のリソース系エンティティ

- 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
- 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2)。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あるので◇つきの点線としました。あとは1対1です。

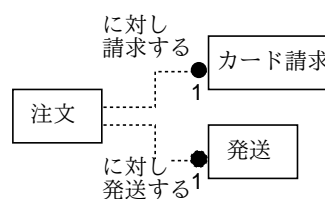


図 3: ネット書店のイベント系エンティティ

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3)。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。

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、この程度のものでかなりごちゃごちゃになりました(図4)。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。

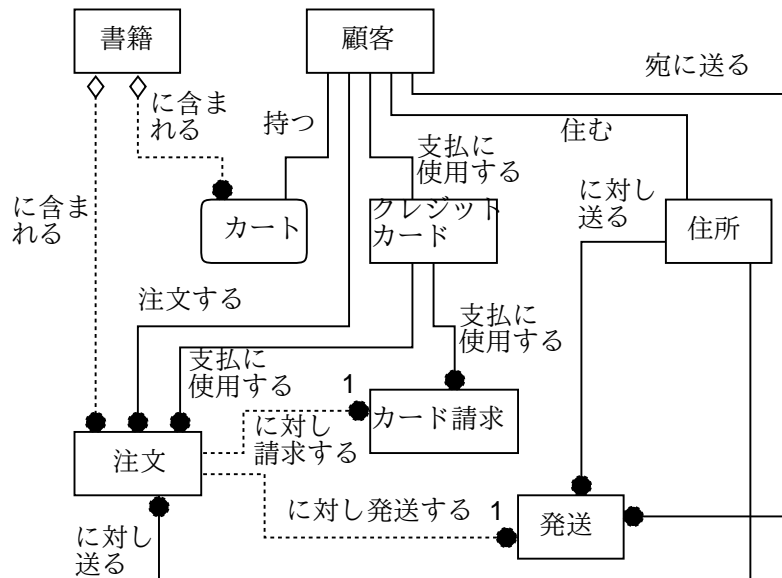


図 4: 全エンティティを含む関連図

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。

1.3 キーと主要属性の定義

ここま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

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4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つては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)。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5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。

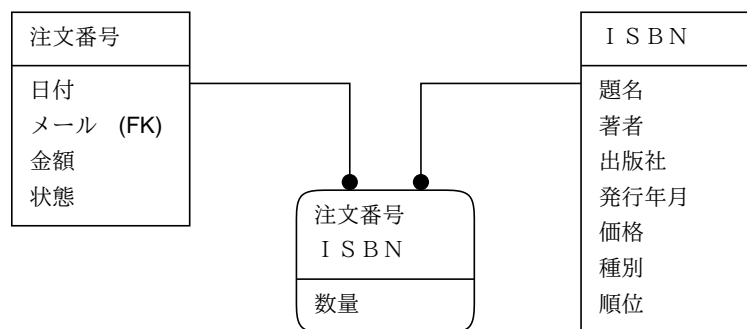


図 5: 注文明細と注文

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。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。

1.4 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 と DBMS の連携

2.1 PHP とデータベースの連携

では次は PHP の続きです。今回はフォームからのデータ入力までを扱いましたが、ここでいよいよ、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、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。

2.2 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みましょう(図 7)。



図 6: 名簿データの登録

```
<?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) }
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_escape_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。

2.3 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 8)。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>

```

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」 と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。

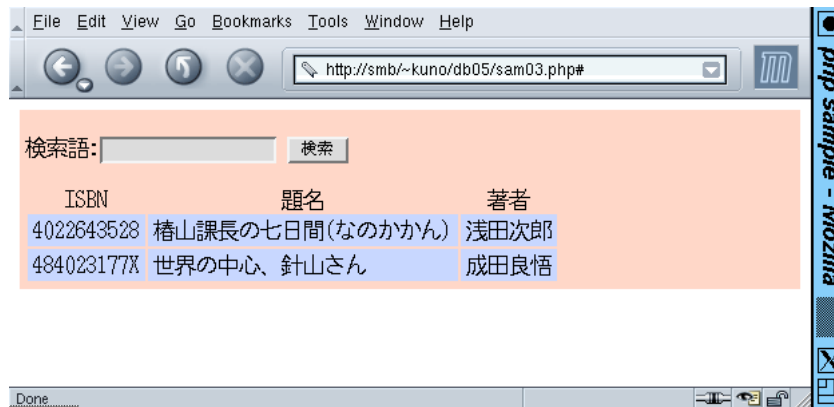


図 7: 簡単な書籍検索

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 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` するなどよい)。