展開演習Bテキスト

椙山女学園大学現代マネジメント学部　三木　邦弘

2020 年 12 月 23 日版

1. はじめに

前期の「展開演習 A」は初めてかつオンラインと言うことで思うように進めることができませんでした。2年生の「Web デザイン」や「プログラミング基礎」で扱った範囲を全てやり直してさらに先に行きたかったのですが、「プログラミング基礎」の PHP の手前で終わりました。課題もつまらないところで引っかかるので、2 年生の課題と同レベルのものしかできませんでした。つまらないところで引っかかるのは初心者では避けられないところです。そして初心者にはつまらない間違いかどうかも分かりません。どうせ悩むならば本質的な問題で悩んでもらわないともったいないです。遠隔授業で一番困るのは学生側の状況が見えないことです。課題の回答からではどこに問題があったのか見出すのは困難です。

「展開演習 B」では、**データベース**と **Web でのデータ収集**をテーマにしたいと思います。その前に「プログラミング基礎」の PHP を片付けます。PHP は Web サーバーで動くプログラムを記述するもので、これでデータベースを扱ったり、データ収集をします。データ収集の対象は、データ入力用の web ページに入力されたもの、インターネット上にある Web ページと IoT (Internet of Things) と呼ばれるネットワークに接続された機器です。

この国ではコロナの感染者の把握するために、当初 Fax を利用していました。Fax で各地の保健所から送られてきた感染者の情報を手入力して集計していました。感染者情報を入力するための web ページを作成し、そこに入力された情報をデータベースに入れるだけのものがなぜすぐできなかったのか？大変残念な現実です。現在はそういうシステムを作ったようですが、感染者一人あたりについて入力する情報量が多くて保健所を困らせているようです。自分でデータ入力用の web ページを作成する場合は、よく考えないといけません。

インターネット上には大量の情報があります。それをうまく取り込めれば、少ない労力や費用でデータを集めることができます。ただ人が見るために作られた Web ページには余分なものが大量にあるので、その中から必要な部分を取り出さなければなりません。PHP では簡単に Web ページ全体を取り込むことができます。その中から必要な部分を取り出すところは自分でなんとかしなければなりません。ある手順に従えば必要な情報だけ提供してもらえるシステムもあります。たいてい有料になりますが、安定して情報を受け取ることが可能になります。

IoT は機器をインターネットに接続する費用が低減したことがきっかけになっています。Wi-Fi 経由で良いというのであれば、数百円のレベルの機器でインターネットに接続できます。どのような機器を接続するかによって集められる情報が変わります。また出力先として利用することも考えられます。例えば数千円レベルでインターネット経由で電源の on/off ができるコンセントなどが売られています。

このようなものを実際に作りながら学んで行けたらいいなぁ、と考えています。

2. PHP

前章では JavaScript を利用して、利用者の入力に応答する Web ページを作成する方法について学びました。しかしさらに利用者に対して高度な応答を行うためには、ブラウザ側だけでは無理があります。例えば Yahoo などのキーワード検索を行おうとしても、あらかじめ検索のデータベースの内容を全てブラウザに送ることは無理でしょう。また注文の受付のページでは入力された注文内容をサーバーの方で記録し、発注の処理に入らなければなりませんが、お客のブラウザからではそのようなことができません。

この章で扱う PHP は、1994 年に Rasmus Lerdorf 氏によって開発が始まったスクリプト言語[*1]です。もともとは「Personal Home Page」から名づけられたようですが、現在は個人の手を離れて「PHP: Hypertext Preprocessor」として多くの人々の手によって開発が続けられています。

PHP の特徴の一つは、サーバー側での処理を通常の HTML のファイルの中に混在させることができる点です。サーバーで処理した結果をブラウザで表示するためには、これまでどおり HTML による記述が必要です。そしてそのような部分はこれまでどおり記述し、サーバーからの応答の部分だけ PHP の記述するような形になります。慣れれば大変便利ですが、HTML、JavaScript、PHP がごちゃごちゃにならないようにご注意ください。

PHP の特徴として他に良く上げられるのは、様々なデータベースソフトと簡単に接続できる点です。さきほど例に挙げたキーワード検索や注文受付の処理では、通常データベースが利用されます。検索ならばあらかじめデータをデータベースに入れておき、その中から探します。また注文受付も、受け付けた注文の内容をデータベースに登録するのが普通です。このように大抵の処理ではデータベースの利用が必須となっていますが、PHP から簡単にデータベースを利用できるので、大抵の処理が簡単に記述できるということになります。

Web サーバーで利用できるプログラミング言語として以前は Perl などが良く使われて来ましたが、現在では多くのシステムが PHP で構築されています。

PHP についての参考サイトとしては本家のマニュアルのページが一番だと思います。ほとんどの部分が日本語になっていますし、使用例などもあります。

「PHP マニュアル」　https://www.php.net/manual/ja/index.php

2.1 どこに入れるのか

PHP による記述はこれまでの HTML の入っていたファイルの中に含めます。入れる場所は特に決まっていません。記述された内容は基本的にはファイルの最初の方に記述された方から実行されます。そして実行された結果がブラウザに送られます。ブラウザ側で「ソースの表示」を行っても PHP の記述は残っていません。

HTML の記述と PHP の記述が混ざらないように、PHP の記述は`<?php` と`?>`の間に入れます。つまり HTML 的には「?php」というタグの形をしています。終わりのタグではありませんから、`/?>`のように「/」を入れないでください。

```
<?php
    PHP の記述
?>
```

PHP を利用する際の重要な注意は、

- ファイルの拡張子は「.htm」ではなく「.php」にする。
- 動作を確認する際は必ずサーバーに送らなければならない。

[*1] 簡易プログラミング言語とも呼ばれるもの。通常実行のための変換を必要とせず、すぐにコンピュータで実行できる。

と言う点です。拡張子が従来どおりの「.htm」では、PHP の記述は単なる変なタグと言うことでブラウザで無視されます。またサーバーが PHP の実行を行うので、デスクトップに保存したアイコンをダブルクリックしたのではだめです。peditor で作成した場合は、できたファイルは mars 上にありますので問題はありません。パソコン上で作成した場合は必ず mars のデスクトップにある「www」フォルダーに入れて、ブラウザで動作の確認をしてください。

2.2 表示の命令

まずは JavaScript の document.write に相当するものです。これを利用して後述の変数の内容を表示するだけでなく、通常のメッセージなどを HTML のタグを含めた形で表示することもできます。簡単な例を次に示します。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>PHP の例</Title>
</Head>

<Body>
<?php
   echo "こんにちは<Br>";
   echo 'He said, "I love you.".<Br>';
   echo '<Font size="7">',3+4,"</Font>";
?>
</Body>
</HTML>
```

この例からわかるように何らかの表示を行いたいときは、PHP では「echo」と言う命令を用います。JavaScript の document.write は関数だったので () の間に表示したいものを入れましたが、こちらは命令なので () は不要です[*2]。また区切りの「;」は必ず必要なので忘れないようにしてください。

PHP で何か間違えるとエラーのメッセージが表示されます。例えば、先ほどの例で 9 行目にある二つ目の echo の行の最後の「;」を忘れると、「Parse error: parse error, unexpected T_ECHO, expecting ',' or ';' in ..../test.php on line 10」のようなメッセージが表示されます。10 行目で「,」か「;」を期待していたのに、お呼びでない「echo」があったぜ、と言うような意味です。このように行の最後で間違えるとエラーの場所が次の行として表示されることもあります[*3]。

JavaScript では"〜"と'〜' は全く同じ意味でしたが、PHP では少し違います。次に出てくる変数を「〜」の部分に含んだとき、前者では変数の部分がその中身と置き換えられますが、後者ではそのようなことがありません。

```
$x=123;
echo "x=$x<Br>";      // x=123 と表示される。
echo 'x=$x<Br>';      // x=$x  と表示される。
```

2.3 変数・計算式

PHP では変数を宣言せずに使用することができます。ただそれではどれが変数なのか見分けがつきにくくなるので、「$x」のように必ず先頭に「$」を付けます。また変数名には残念ながら漢字は使用できません。英

[*2] PHP にも関数があります。PHP の関数を用いる場合は、JavaScript と同様に () の中に入れることになります。
[*3] 「}」を忘れると、はるかかなたでエラーになることもあります。

数字[*4]にしてください。

変数が「=」の左側に出てきた場合は、変数に何か入れようとしていることを示しています。変数には数値や文字列などを入れることができます。その他の場合は変数に入っている内容を示しています。”123”と言うような文字列は計算式の中にあれば数値の 123 として扱われます。また必要があれば数値が文字列に自動的に変換されることもあります。

```
$x="123"+456;
echo "x=",$x;      // x=579 と表示される。
```

計算式は JavaScript と同様に記述することができます。ただ「+」は本当に加算の意味でしかありません。文字列と文字列をくっつけたいときは「.」を使用します。

```
$yen=1234;
$kekka="お値段は".$yen."円です。<Br>";
echo $kekka;                    // お値段は 1234 円です。 と表示される。
```

2.4 繰り返し (1)

コンピュータは昔「電子計算機」とも呼ばれたように高速に計算ができます。しかしいくら高速でも、計算すべき式が少ししかなければあまり意味がありません。また式がたくさんあってもそれをいちいち入力しなければならないのでは大変です。変数があるので計算結果をもう一度入力すると言うような事は避けられますが、これだけでは足りません。プログラミング言語では通常繰り返しと言うものが簡単に記述できるようになっており、数行のプログラムで何億回も計算させると言うような事が可能です。

たとえば PHP では for 文と呼ばれる次のような記述で「I love you.」を 100 回表示することができます。

```
for ($i=0;$i<100;$i++){
   echo $i," I love you.<Br>";
}
```

これは for 文と呼ばれるもので類似のものが通常のプログラム言語では必ず存在します[*5]。PHP では for 文は、for(A;B;C){D} のようなちょっと複雑な形をしており、次のような意味になります。

1. A を実行します。ここには通常代入文が入ります。
2. B の条件を調べます。もし条件不成立の場合は for 文は終了します。
3. D を実行します。ここには任意の複数の文を書くことができます。
4. C を実行します。ここには例のような式または代入文が入ります。
5. 2 番目に戻る。

B が条件であること、C と D が書いてある順番と逆に実行されることに注意してください。上記の実例の場合次のような感じで実行されます。

1.「$i=0」を実行するので変数 i の値がゼロになる。
2.「$i<100」の条件を調べると、変数 i の値はゼロなので条件は成立する。
3.「echo」を実行するので「0 I love you.」と表示する。
4.「$i++」を実行する。これは「$i=$i+1」と同じ意味なので、変数 i の値は 1 になる。

[*4] 英字または数字で、先頭は英字にしてください。
[*5] 既に扱った JavaScript でももちろん使用可能です。

5.「$i<100」の条件に戻る。変数 i の値は **1** なので条件は成立する。
6.「echo」を実行するので「**1** I love you.」と表示する。
7.「$i++」を実行する。変数 i の値は **2** になる。
8. 繰り返すたびに変数 i の値は増加していく。
9.「99 I love you.」と表示した後、変数 i の値は 100 になる。
10.「$i<100」を満たさなくなるので for 文は終了する。

もし 1000 回「I love you」と出したい場合は、「$i<1000」に変更します。i の値をどんどん減らしたい場合は、「$i++」の代わりに「$i−−」を使用します。これは「$i=$i-1」と同じ意味です。

2.5 配列型変数

コンピュータで大量の情報を処理したい場合、それに応じて変数も大量に必要になります。そのような場合にそれぞれの変数に別の名前を与えるのは大変です。そこで配列型の変数と言うものがプログラミング言語では使えるようになっています。PHP では配列型の変数も変数ですので「$」から始まります。そして変数の名前の後に「[ ]」がありその中に添え字と呼ばれるものを指定します。普通のプログラミング言語では添え字は数値ですが、PHP では文字列も使用できます。

```
$data[1]=123;$data[2]=456;$data[3]=789;
for ($i=1;$i<=3;$i++) {
    echo "$data[$i]<Br>";
}
$data['name']="三木";$data['address']="名古屋市";
echo $data['name'],"の住まいは",$data['address'],"です。";
```

普通の変数ではまずありえませんが、配列型変数では誤って存在しない変数を使ってしまうことがあります。例えば先程の例で$data[1]〜$data[3] については数値を入れてますが、$data[4] には何も入れていませんので、$data[4] は存在しません。うっかり for を回しすぎて使ってしまうとエラーになります。次の節のフォームからの入力の際にも、選択されなかった場合対応する変数が存在しないことがあります。変数が存在するかどうか確認する際は次のようにします。

```
if (isset($x)) {
   $x が存在し、NULL でないものが入っていた場合の処理
}
```

NULL は特殊な値で、通常これを代入して入れない限りまず変数には入りません。

2.6 フォームからの入力

フォームの入力欄に利用者が入力した内容を PHP で処理するためには、一旦サーバーに入力した内容を送り返してもらわなければなりません。そのためにはクリックしたらフォームの内容を送信するボタンが必要になります。このボタンは HTML で次のように記述します。

```
<Input type="submit" value="送信">
```

value で指定した内容はボタンの上に書かれる文字ですので任意のものが可能ですが、利用者が最後にこれをクリックしなければならないことがわかるようなものにします。またこれだけではサーバーに送った内容をどの PHP のファイルが処理をすれば良いのかわかりません。この処理する PHP のファイルの指定は、Form のタグのところで行います。

```
<Form method="POST" action="prog.php">
```

method については POST 以外に GET も指定可能です。GET の場合は URL の部分に入力された内容が追加されます。そのため GET の方はサーバーに送れるデータ量に制限がありますし、URL を見ると何を送ったのか判ってしまいます。action ではサーバーで処理するプログラムを指定します。

指定された PHP のファイルの方では、特別な配列型変数を用いることにより、簡単にフォームの中で入力した内容を取り出すことができます。例えば「namae」と言う名前を付けた入力欄の値は、「$_POST['namae']」で取り出すことができます。もし GET で送った場合は、「$_GET['namae']」で取り出すことができます。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>入力欄の例</Title>
</Head>

<Body>
<Form method="POST" action="prog.php">
名前：<Input name="namae"><Br>
<Input type="submit" value="送信">
</Form>
</Body>
</HTML>
```

この入力欄の例には PHP の記述は何も含まれて居ませんので、通常の拡張子が.htm のファイルに入れます。(例えば「input.htm」) そして次のは必ず「prog.php」と名前を付けて保存してください。(上記の例で action=でこの名前を指定しているので。)

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>入力欄処理の例</Title>
</Head>

<Body>
<?php
   echo "名前には、",$_POST['namae'],"が入っていました。<Br>";
?>
</Body>
</HTML>
```

文字や数字を入れられる入力欄以外にフォームにはラジオボタン、チェエクボックス、選択メニュー等があります。これらの選択状況は次のようにして知ることができます。

- ラジオボタンの場合：

  ```
  <Input type="radio" name="sex" value="男"> 男<Br>
  <Input type="radio" name="sex" value="女" checked> 女<Br>
  <Input type="radio" name="sex" value="宇宙人"> その他<Br>
  ```

  このような場合、選択したものは$_POST['sex'] の内容でわかります。この場合変数の内容は、「男」、「女」、「宇宙人」のいずれかになります。

- チェックボックスの場合：

```
<Input type="checkbox" name="sugar" checked> 砂糖<Br>
<Input type="checkbox" name="lemon"> レモン<Br>
<Input type="checkbox" name="milk"> 牛乳<Br>
<Input type="checkbox" name="brandy"> ブランデー<Br>
```

このような場合例えば$_POST['sugar'] があるかどうかで選択したかどうか分かります。通常次のように isset() を使います。

```
if (isset($_POST['sugar'])) {
   砂糖を選択していた場合の処理
}
```

- 選択メニューの場合：

```
<Select name="selone">
   <Option value="sleep">寝る
   <Option value="eat" selected>食べる
   <Option value="run">走る
</Select>
```

このような場合、選択したものは$_POST['selone'] の内容でわかります。この場合変数の内容は、「sleep」、「eat」、「run」のいずれかになります。

- 選択メニュー (複数選択可) の場合：
  次のように name で指定する名前の後に [ ] を追加します。

```
<Select name="selmul[]" multiple>
   <Option value="sleep">寝る
   <Option value="eat" selected>食べる
   <Option value="run">走る
</Select>
```

このような場合、最初に選択したものは$_POST['selmul'][0] の内容でわかります。そして 2 つ選択した場合は、2 つ目に選択したものが$_POST['selmul'][1] の内容でわかります。やっかいなのは 1 つしか選択しなかった場合は$_POST['selmul'][1] は存在しないので isset() で確認をしてから内容を見る必要があることです。

2.7 URL で入力内容を送る

URL に入力内容を付けて送ることができます。この方法を利用するとブラウザの URL を表示するアドレスのところに入力内容も表示されてしまうので、知られては困るような内容を送ることには使えません。また、余り大きなデータは URL の長さの制限に引っかかる恐れがあるので使えません。それでもリンクをクリックするだけで送るようなことも可能なので、この方法を利用する場面も少なくありません。いくつかの送り方がありますが、それを受け取る PHP の方は同じ方法で受け取ります。

まず一つ目の方法ですが、Form タグの method を GET にする方法です。前節の例で

```
<Form method="POST" action="prog.php">
```

となっていたところを

```
<Form method="GET" action="prog.php">
```

にするだけです。

2 つめの方法として URL に直接記述する方法です。例えば次のようにします。

```
<A href="prog.php?namae=aiueo">名前は aiueo</A>
<A href="prog.php?namae=aiueo&age=20">年齢は 20 歳</A>
```

1 つ目の例では、クリックすると「namae」という入力欄に「aiueo」と入力して prog.php を呼び出すのと同じ事になります。2 つ目の例は、2 つの入力欄に対応した場合で、「&」を追加してより多くの入力欄にも対応できます。この方法で注意しなければならないのは、送る内容に記号や漢字が含まれる場合は、それらを置き換える必要があることです。幸い PHP には urlencode と言う変換をする関数があるのでこれを利用します。

```
echo '<A href="prog.php?namae=',urlencode("椙山花子"),'">名前は椙山花子</A>';
```

PHP 側でこのような方式で送られてきたデータを取り込むには、$_POST の代わりに$_GET['**入力欄の名前**'] と言う変数を使います。

2.8 他のページへの移動

PHP で内部的な処理を行った後、別のページへ移動したい場合は、次のような header() を使用します。

```
header("Location: http://www.sugiyama-u.ac.jp/");
```

この例では椙山のトップページに移動しますが、同じディレクトリにある別のファイルでも構いません。その場合はファイル名のみ指定します。使用上注意することは、これ以前に echo などでブラウザに他の内容を送ると無効になる点です。<HTML>などのタグを<?php の前にうっかり書いてしまう事が多いので注意しましょう。

条件によって移動したり、しなかったりする場合は、header() の次に exit; を入れておくとここで、PHP の処理が終わって以降の部分は実行されなくなります。そのため大きな else が不要になって見やすくなります。

2.9 ファイルへの入出力

利用者からの入力を保存するためにはファイルを作成して、そこに入力内容を入れなければなりません。また逆にファイルに入っている情報を取り出して表示するようなこともしばしば行われます。ここではそのようなファイルに関する操作をどのように PHP では記述するのかについて説明します。

ファイルを開く

ファイルを利用するためにはまず「ファイルを開く」という操作が必要になります。このとき、これから扱うファイルの名前とファイルに対してどのような操作を行いたいかを指定します。

```
$file=fopen("aaa.txt","w");
```

fopen が「ファイルを開く」関数です。fopen の最初に指定しているのがこれから操作するファイルの名前です。この例では「aaa.txt」という名前のファイルが対象となります。PHP のすごいところは、このファイルを指定しているところに URL を書けば、他のサーバーにあるファイルを読み込むこともできることです。さすがに書き込むことはできませんが。インターネットには Web ページの形で非常に多くの有益な情報があり

ます。そのような Web ページを開き、取り込み、その中の本当に必要な部分だけ取り込むということが PHP では比較的容易に行うことができます。

次に”w”という指定がありますが、これはファイルに情報を書き込む (write) ことを示します。もし「aaa.txt」がなければ、ここで「aaa.txt」と言う空のファイルが作成されます。もし既に存在した場合は、これまで入っていた内容は消されてしまうので注意します。

これまで入っていた内容に追加 (append) をしたい場合は、”w”の代わりに”a”を指定します。ファイルがなかった場合は”w”を指定したのと同じ動作になります[*6]。

ファイルの内容を読み出し (read) をしたい場合は、”r”を指定します。もし存在しないファイルに対してこの指定をすると fopen は「false」と言う特殊な値を返し、以下の読み書きはできませんのでででご注意ください。

fopen の返す値をこの例では$file という変数に入れています。これは後で使用しますので、適当な名前の変数に必ず入れてください。なお、同時に複数のファイルを扱うことができますが、その場合は fopen の結果をそれぞれ異なる変数に入れてください。

ファイルへの書き込み

ファイルへ何かデータを書き込む場合は、次のように fwrite を使用します。

```
fwrite($file,"書き込むデータ¥n");
```

この例では先ほど fopen で開いたファイルに「書き込むデータ」と言う文字列と改行 (¥n) が書き込まれます。echo と同様に変数を指定すれば変数の内容を書き込むこともできます。fwrite の () の中には一つしか書き込みたい内容を指定できないので、文字列と変数の内容を両方書きたい場合は、文字列の結合を示す「.」でつなぎます。書き込みたい内容が多数ある場合は fwrite を必要な数だけ繰り返し記述します。複数のデータを書き込む場合は、データとデータの間に空白や改行が入るようにします。例えば「123」と「456」を続けてファイルに書き込むとファイル中には「123456」が残るので、後でこれを読み込んでもどこで区切れば良いのかわからなくなります。次のようにファイルに書き込むと、

```
$a=123;
fwrite($file,$a."¥n");
fwrite($file,"ありがとう¥n");
```

ファイルには次のような内容が入ります。

```
123
ありがとう
```

ファイルからの読み出し

ファイルからデータを読み出すには、次のように fgets を用います。

```
$data=fgets($file,256);
```

これで先ほどの fopen で開いたファイルより 1 行分読み出され、変数$data に入ります。256 はデータの最大長です。もしファイルにこれより長い行があった場合は、とりあえずここで指定した分だけ変数に入ります。この fgets を繰り返すことによりファイルの先頭から順番に 1 行ずつ読み出すことができます。

[*6] PHP 本来の機能ではこれで説明終わりなのですが、実際に使用してみると「Permission denied」と言うエラーが生じることがあります。これは許可がないと言う意味で、Web サーバーのソフトが PHP を動かしてファイルを作成しようとしたところ、利用者の領域だったのでファイルを作成できなかったためです。そのような事が生じないように ruid2 と言う Web サーバーのモジュールでこの問題を回避するように設定してあります。

ファイルを閉じる

ファイルに関する操作が終わったら次のようにしてファイルを閉じます。ファイルの読み出しの際は大きな問題になることは少ないですが、書き込みの際に忘れると色々問題を生じることがあります。

```
fclose($file);
```

ファイルの内容の確認

peditor でファイルを編集することにより、ファイルに書き込んだ内容を確認することができます。また mars に接続しメニューの「アクセサリ」にある「Kate」でファイルの内容を見たり、編集したりすることができます。

ファイルと配列型変数のやりとり

通常はファイルから行単位で取り出したり入れたりするのですが、PHP にはファイルの内容を配列型変数に一気に取り込んだり、逆に配列型変数の内容を一気にファイルに入れることが可能です。

```
$lines=file("aaa.txt");
$lines2=file("aaa.txt",FILE_IGNORE_NEW_LINES);
```

これで「aaa.txt」という名前のファイルの内容が 1 行毎に$lines 変数に入ります。つまり「aaa.txt」の 1 行目が$lines[0]、2 行目が$lines[1] に入ります。2 つ目の例では$line2 変数に「aaa.txt」の内容から各行にあった行末の改行 (￥n) を除いたものが入ります。ファイル名の代わりに URL を指定すると、他のサーバーのファイルを取り込むこともできます。

逆に配列型変数$lines に入っている内容を「bbb.txt」と言うファイルに一気に書き込むのであれば、

```
file_put_contents("bbb.txt", $lines);
```

とします。このとき「bbb.txt」が存在しなければ新たにファイルが作られます。また既存の場合は、入っていた内容は消えて$lines の内容と置き換わります。

2.10 繰り返し (2)

for 文以外にも while 文と言われ繰り返し文があります。

```
while（条件）{
   条件が成立している間に繰り返し実行する内容
}
```

for 文でも時々終了しない繰り返しになって困りますが、while 文は繰り返される内容の中で条件が変化するようなことをしないと簡単に終わらないものになってしまうので注意します。また、do–while 文と呼ばれる繰り返し文もあり、次のような形をしています。

```
do {
   条件が成立している間に繰り返し実行する内容
} while（条件）;
```

while 文と異なるのは条件を調べるのが実行をしてからと言う点です。つまり繰り返し実行される内容は少なくとも一回は実行されます。

あらかじめ何行分のデータがファイルにあるとわかっている場合は、その数だけ fgets を書けば良いのですが、実際はどのくらい入っているかわからない例も多数あります。そういう場合はファイルから読み出しができなかった場合に fgets が「false」と言う値を返すのを利用して次のような while による繰り返しを使います。

```
while ($data=fgets($file,256)) {
   $data に読み出されたデータの処理
}
```

前節で file 関数を使用してファイルの内容を配列型の変数に取り込めると言う説明がありましたが、ファイルの大きさが不明の場合、配列変数の何番目まで入っているのか分かりません。そういう場合は次のような繰り返しを使用します。

```
foreach ($lines as $data) {
   $data に読み出されたデータの処理
}
```

foreach が$lines から順番に一つずつ取り出して$data 変数に入れて処理に回してくれます。全て取り出しが終わったら繰り返しが終了します。

2.11 ファイルとディレクトリの操作

ファイルの中身とのやり取りの方法は既に述べましたが、ここではファイル自体の操作について説明します。また複数のファイルに対して何か操作を行いたい場合、それが入っているディレクトリは判っているが、どのような名前のファイルがいくつあるのか判らない場合もあります。そのような場合はディレクトリの情報を取り出して調べることになります。

ファイルの存在の確認

指定した名前のファイルが存在するかどうかを調べることができます。ファイルからデータを読み込む場合に、ファイルが存在しない場合 fopen の際にエラーが発生します。エラーを避けるためには file_exists 関数で確認してから fopen するようにしましょう。

```
if (file_exists("aaa.txt")) {
   aaa.txt が存在した場合の処理
}
```

ファイル名の変更

ファイルの名前を変更したい場合は次のように rename を使います。この例の場合、「aaa.txt」の名前が「bbb.txt」になります。ただし元のファイル (aaa.txt) が無い場合はエラーになりますし、変更先のファイル (bbb.txt) が既に存在している場合もエラーになりますのでご注意ください。

```
rename("aaa.txt","bbb.txt");
```

ファイルの削除

不要になったファイルを消すことができます。次のように unlink を用いて指定したファイルを削除することができます。

```
unlink("aaa.txt");
```

ディレクトリの読み出し

ディレクトリを開いて、その中にあるファイルを調べることができます。以下は PHP のマニュアルに出ていた例を簡略化したものです。

```
$dir=".";
if (is_dir($dir)) {
   $d=opendir($dir);
   while ($file=readdir($d)) {
      echo "ファイル名: ",$file," タイプ: ",filetype($dir."/".$file),"<Br>\n";
   }
   closedir($d);
}
```

1. 最初にこれから調べようとしているディレクトリ名を$dir と言う変数に入れています。「.」は今居るディレクトリーのことで、これが書かれている PHP のファイルがあるディレクトリのことになります。
2. is_dir 関数は指定したディレクトリ名が本当にディレクトリかどうか判定します。
3. opendir 関数で指定したディレクトリを開きます。ファイルを開いた時と同様にその結果を$d と言う変数に入れています。
4. readdir 関数で、ディレクトリ内のファイル名を一つ取り出します。readdir 関数を繰り返すことによりディレクトリ内の全てのファイルの名前を取り出すことができます。
5. filetype 関数は、指定した名前がファイルだった場合「file」、ディレクトリであった場合は「dir」になります。どこのディレクトリにあるファイルかを示すために、$dir と$file をくっつけています。「/」はディレクトリとファイル名の区切りの記号です。
6. opendir 関数で開いたディレクトリは、closedir 関数で閉じてください。

2.12 条件判断

PHP の条件判断も JavaScript と全く同じ形をしています。() の中の条件の書き方も同様です。例えば「変数 a の内容が 10 に等しくない」は「$a!=10」のように書きます。

```
if（条件）{
   条件が成立したときの内容
} else {
   条件が成立しなかったときの内容
}
```

条件が不成立の場合の内容がないときは、else 以下は省略できます。PHP が JavaScript と大きく異なるところは内容のところに HTML などの内容を直接記述する方法があることです。例えば次のような感じです。

```
if ($tokuten==100) {  // 得点の変数($tokuten)が100ならば画像を表示する。
?>
   <Img src="manten.gif">
<?php
}
```

つまり一度?>で PHP の記述を終わらせて通常の HTML のタグなどを記述することができます[*7]。ただ閉じる「}」などを忘れるとエラーになります。

2.13 文字列関数

コンピュータは、数値の計算をするから電子計算機と呼ばれたのですが、現在の使われ方は計算以外の仕事の方が多いようにも思われます。数値以外のデータとして例えば文字列があります。ここでは PHP で文字列を扱う際に使われる関数のいくつかを紹介します。

- strlen 関数：文字列の長さを求める関数です。文字列の長さは、文字列を構成する文字の数です。

```
echo strlen("abcdefg");    // 7と表示される
```

- strpos 関数：文字列の中に、指定した文字列が含まれるかどうか調べる関数です。最初に指定した文字列の中に、次に指定した文字列が含まれれば、何文字目からかを答えます。もし含まれて居なければ false を返します。

```
echo strpos("abcdefg","cde");    // 2と表示される
```

実は先頭をゼロと数えるので上記の例では 3 ではなく 2 が表示されます。PHP ではゼロと false は同等に扱われます。そのために含まれなければ何かすると言う場合は次のような書き方をします。

```
if (strpos($x,"abc")===false) {
    $xにabcが含まれなかった場合の処理
}
```

等号が 2 つでも妙な感じでしょうが、等号が 3 つ必要となります。これの否定の条件では「!==」になります。

- substr 関数：文字列の一部を切り出す関数です。何文字目から切り出すのかと、何文字分切り出すかを指定します。何文字分の指定が省略された場合は、指定された文字以降全てが切り出されます。

```
echo substr("abcdefg",2,3);   // cdeが表示される
echo substr("abcdefg",4);     // efgが表示される
```

何文字目かの指定の際も先頭の文字がゼロなので注意します。また何文字目かのところを負の数にすると、文字列の後ろから指定することもできます。

```
echo substr("abcdefg",-3,2);   // efが表示される
```

[*7] JavaScript ではこのような場合、document.write を使用する必要がありました。

- `str_replace` 関数：文字列の置き換えをする関数です。探す文字列、それを置き換える文字列、探して置き換えられる文字列の 3 つを指定する必要があります。置き換える文字列が複数含まれていても、全て置き換えてくれます。

```
echo str_replace("ab","x","abcdabc");   // xcdxc が表示される
```

同時に複数のものを探して、各々別のものに置き換えることもできます。次の例では”ab”が”x”に、”c”が”y”に全て置き換えられます。

```
echo str_replace(["ab","c"],["x","y"],"abcdabc");   // xydxy が表示される
```

また以下の操作は既にやりました。

- 文字列の連結：文字列と文字列を連結したい時には「.」を使います。
- 文字列の比較：二つの文字列が等しいかどうか調べる時は「==」、等しくないかどうかを調べる時は「!=」を使用します。アルファベット順で前にあるものが、より小さいと判断されるので、「<」や「>」も使用可能です。

なお、処理する対象の文字列が、半角の英数字だけでなく、日本語の漢字などを含む場合はそういう文字に対応できる関数を用います。例えば「strlen(”あいう”)」の結果は 6 になります。一方表 1 にある「mb_strlen(”あいう”,”UTF-8”)」ならば結果は 3 になります。正しい結果を求めるためには、正しい encoding の設定 (前述の例では”UTF-8”の部分) を行わないとならないので、日本語限定でも結構面倒です。

表 1 日本語文字列対応の関数

内容	半角英数字用	日本語にも対応
文字列の長さ	strlen	mb_strlen
文字列の位置	strpos	mb_strpos
文字列の切り出し	substr	mb_substr

2.14 Web ページ間の情報のやり取り

Web ページの間で情報をやり取りする必要がある場合、表 2 のようにいくつかの方法があります。状況に合わせて方法を一つ選択したり、複数の方法を併用します。

表 2 Web ページの間で情報をやり取りする方法

状況	方法
submit の際に追加情報を送る	hidden を使う
以前来た事があることを伝える	cookie を使う
Web ページ間で情報の共有	session を使う

hidden を使う方法

<Form>〜</Form>の間に入力用のタグを入れることにより、利用者が入力した内容を action で指定した Web ページに送ることができます。この際に利用者が入力した以外の情報も送ることができます。例えば次の

ようなタグを<Form>〜</Form>の間に入れることにより、action で指定した Web ページでは$_POST['id']または$_GET['id'] で「abc123」という情報を受け取ることができます。

```
<Input type="hidden" name="id" value="abc123">
```

このとき、この入力欄はブラウザの画面に表示されないため、Web ページのソースを確認しない限り利用者は気が付きません。逆に言えばソースの表示で送る内容がばれてしまうので、利用者に知られては困る内容を送る際には使えません。ブラウザの画面に表示されないという点が異なるだけで、他は type の指定のない<Input>と同じですので、JavaScript で値を読み取ったり、書き換えたりすることができます。

cookie を使う方法

Web サーバーから Web ページの内容を送る際に、cookie と呼ばれる情報も送ることができます。cookie を受け取ったブラウザは、次回**同じディレクトリにある Web ページ**へアクセスする際にこれを送り返します。よって初回のアクセスの際に cookie を送っておけば、次にアクセスがあった場合に cookie の有無で二回目以降かどうかわかります。

特に期限を設定しない場合は、ブラウザを終了すると cookie は消えます。よって次の日以降の再訪問を調べたいのであれば期限を設定するべきでしょう。一方パスワード認証を通過した証拠として cookie を使用する場合は、期限を設定しないことにより、正しく終了処理 (cookie の消去) をしなかった際に、他の利用者が使ってしまうことをある程度防ぐことができます。

まず「namae」という名前の cookie に「12345」を設定したい場合は次のように setcookie() を使用します。ただし header() と同様に他の出力が実行される前に使用する必要があります。後になると cookie が設定されません。

```
setcookie("namae","12345");
```

名前が違えば setcookie を複数用いて複数の cookie を設定することができます。この例では有効期限の設定がないので、ブラウザを閉じるまで有効になります。期限を設定する場合は、現在の時刻に期限までの秒数を加えたものを指定します。現在の時刻は time() で得ることができるので次のような形になります。

```
setcookie("namae","12345",time()+3600);
```

これでページを表示してから 1 時間後まで有効になります。期限が表示してからではなく、ある決めた日までと言うのであれば、指定した日時を秒単位に変換する mktime() を使用して次のようにします。

```
setcookie("namae","12345",mktime(13,25,45,6,30,2021));
```

mktime() のところに数字が並んでいますが、これで 2021 年 6 月 30 日 13 時 25 分 45 秒と言う期限が設定されます。順番がわかりにくいですが、時・分・秒・月・日・年の順番に指定します。一方有効期限としてゼロを指定すると cookie を消去することができます。このとき設定する値としては何を指定しても無視されますが省略はできません。

```
setcookie("namae","",0);
```

このように設定した cookie の値は、$_COOKIE['namae'] のような形の配列型の変数で取り出すことができます。取り出しはできても違う値を設定することはできませんので注意します。設定する際は setcookie() を使います。また cookie が設定されていない場合にこの変数を使うとエラーになるので、isset() で存在を確認してから使いましょう。

session を使う方法

これも内部的には cookie を使用していますが、記憶する内容は Web サーバー上にあるので、内容が利用者にばれる恐れが低くなります。使い方は最初に session_start() をすれば、$_SESSION['namae'] のような配列型変数が使えるようになります。ただし setcookie() と同様に他の出力が実行される前に session_start() を使用する必要があります。この変数に入れたものは、別の web ページでも取り出すことができるようになります。

```
<?php
session_start();  // <HTML>より先にする
?>
<HTML>
<Head>
... 中略 ...
<?php
$_SESSION['namae']=12345;          // 他のページで 12345 を取り出せるようになる
echo $_SESSION['bango'],"<Br>\n"; // 他のページで入れたものが読める
```

session の標準的な設定は Web サーバーで行われていて、状況によっては問題を引き起こすことがあります。例えば mars では session で使用する cookie の有効期限が 3 時間になっているので、それ以上アクセスの間隔が開くと session で設定した情報が失われてしまいます。また cookie の設定場所が web サーバーのトップになっているので、同じサーバ内で違うことに session を使おうとしても、同じ cookie を使うために設定内容が混ざってしまいます。session で設定した情報は web サーバーの中にファイルの形で保存されますが、その場所も共通になっています。このファイルはほっておくとどんどん貯まるので、設定次第では自動的に消すようにもできますが、共通の場所を利用している場合、誰かが自動的消去の設定をすると一緒に消されてしまう恐れがあります。これらの設定を自分の今使っているディレクトリのみ変更することができます。そのためには「php.ini」と言う名前のファイルを作成して、次のような内容を入力します。

```
; Cookie を 100 日後まで有効に
session.cookie_lifetime = 8640000
; session ファイルを 100 日後まで有効に
session.gc_maxlifetime = 8640000
; このディレクトリ以下で session 情報を共有する
session.cookie_path = ./
; session ファイルは ses ディレクトリの中に（予め ses と言うディレクトリを作っておくこと）
session.save_path = ./ses/
; 期限を過ぎた session ファイルを削除するのは 100 回に 1 度
session.gc_probability = 1
session.gc_divisor = 100
```

「;」で始まる行はコメントなので入力しなくても構いませんが、入れておいた方が設定した内容がわかりやすくなります。この php.ini をブラウザでアクセスされて見られると、大事な設定内容がばれてしまうので、peditor の「モード変更」で「rw-------」を設定します。

この php.ini の例では session_start() を 100 回使うと古い session ファイルを削除します。session.gc_probability をゼロにするとファイルがどんどんたまるようになるので、何らかの別の方法で消すようにしないと、忘れた頃にディスクが溢れて慌てることになります。

2.15 送信前のチェック

人は誰でも間違いを犯します。Web ページの入力に誤りがあった場合、その誤りが機械的に検出可能であっても、PHP は送信ボタンをクリックして入力内容がサーバーに来ない限り対応できません。そして誤りがあったことをまた送り返さないと利用者には伝わりません。ネットワークの往復をする代わりに、送信ボタンをクリックすると、まず JavaScript でチェックを行い、問題が無ければ本当にサーバーに送信することが考えられます。ネットワークを使用しない分即座に応答を返すことができます。

そのためには、色々手直しする必要があります。元は次のようになっていたとします。

```
<Form method="POST" action="syori.php">
... 中略...
<Input type="submit" value="送信">
... 後略...
```

Form のタグには、「name=”bbb”」のように名前を付けます。これは後で JavaScript からこのフォームに対して送信の指示ができるようにするためです。また、「type=”submit”」のボタンを「type=”button”」に変更し、クリックしても即座に送信されないようにします。さらに onClick を利用して内容をチェックする関数を指定します。そうすると次のような感じになります。

```
<Form method="POST" action="syori.php" name="bbb">
... 中略...
<Input type="button" value="送信" onCLick="check()">
... 後略...
```

ここでは内容をチェックする関数として check() を指定していますが、もちろん check() の内容も定義しなければなりません。入力欄などの内容を調べて、何か問題などがあれば、そのことを警告画面などを使用して利用者に知らせてから retrurn 文を実行するか、関数を終了します。入力の内容に問題がなければ、次のようにしてフォームの submit() 関数を呼び出すことによって、入力された内容をサーバーへ送ることができます。

```
function check(){
   ... 中略...
   document.bbb.submit();   // bbb はフォームにつけた名前
   ... 後略...
}
```

3. SQL

SQL は**関係データベース** (Relational Database) を操作するための言語です。データベースはどのようなデータをどのように記憶するかによって様々な方式が考えられており、関係データベースは IBM 社の E.F.Cood によって 1969 年に提案されました。データは二次元の表 (table) の形で記憶されます。複数のテーブルの間に関係 (relation) を設定することにより、データ間の複雑な関係を扱うことができます。1 年生の「コンピュータと情報 II」に出てきた「Access」も関係データベースを基にしたデータベースソフトです。他にも関係データベースを基にしたデータベースソフトは多数ありますが、大抵 SQL が使えるようになっています。SQL を使用すると、データベースを作成し、データを入れ、データを修正し、データを検索するなどの操作を行うことができます。「Access」の使い方を学んでも「Access」しか使えませんが、SQL を学べば様々なデータベースソフトが使えるようになります。

関係データベースにおいてデータを入れる**テーブル**は、図 1 のように「コンピュータと情報 II」に出てきた Excel の「リスト」と同じ形をしています。1 番上の行が列の見出しで、その下にデータが並んでいます。1 行が 1 件のデータでこれを**レコード**と呼びます。テーブルと「リスト」の大きな違いは主キーの有無です。テーブルには必ず**主キー** (primary key) と呼ばれる列があり、そこにあるデータは全て異なる必要があります。この列のデータを用いてデータベースは各レコードを識別します。

列名 →	id	name	pass
レコード →	1	miki	12345
レコード →	2	maki	abcde
レコード →	3	mika	abc123

↑主キー

図 1 テーブルの例

SQL を用いてデータベースに対して様々な操作を行うことができます。表 3 はデータベースの操作を行う際によく使われる文の一覧です。

表 3 SQL の文の一覧

名前	操作内容
CREATE	テーブルの作成
SELECT	データの検索
INSERT	テーブルへ新しいレコードを追加する
DELETE	テーブルのレコードを削除する
UPDATE	テーブルのデータを更新する

3.1 DB Browser for SQLite

以下実際にデータベースを操作しながら、関係データベースや SQL を理解できるように、簡単に SQL を実行できるアプリケーションとして DB Browser for SQLite (以下 DB Browser) を使用します。これは SQLite と呼ばれる関係データベースソフトを視覚的に操作することができるものです。SQLite は小規模なデータベースを構築するためのデータベースシステムで、同時に複数の利用者で扱うことを想定していません。その代わりに一つのデータベースはテーブルがいくつあってもファイルが一つであり、利用者管理に関するところがなくて扱いやすくなっています。PHP は SQLite を内蔵しており、PHP が使えるならば必ず SQLite も使

えます。また Android も SQLite を標準的に使用しています。

DB Browser は mars のデスクトップのメニューの中の「プログラミング」の中にあります。起動すると図 2 のような画面になります。

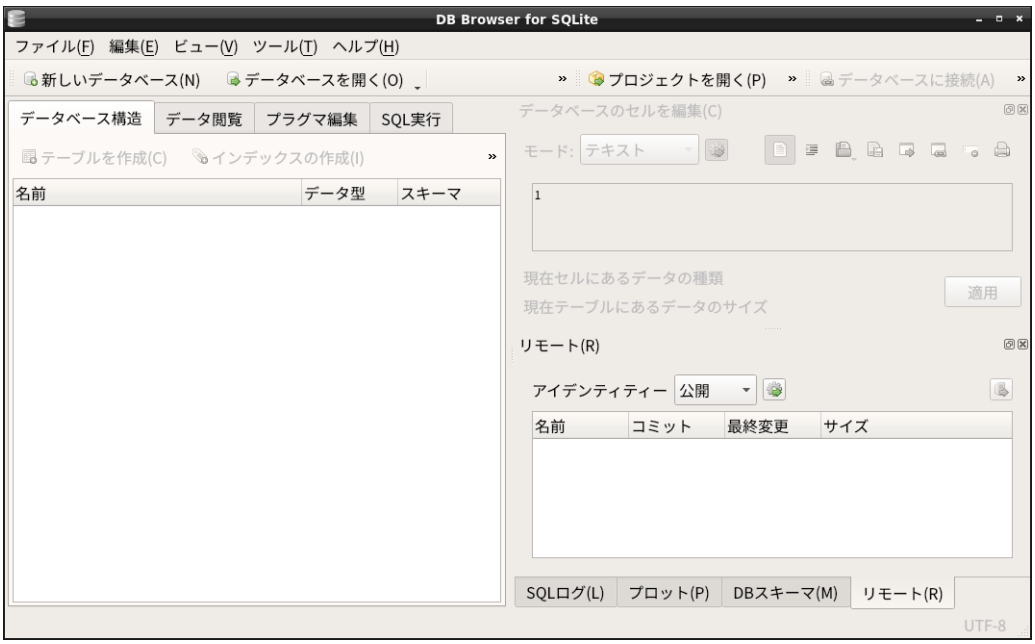

図 2 DB Browser for SQLite の最初の画面

まずデータベースファイルを指定します。既にデータベースがあるならば「データベースを開く」、新規のデータベースであれば「新しいデータベース」をクリックし、保存先 (例えば「デスクトップ」) を開き、ファイル名は「aaa.db」などにします。実は拡張子は決まっていません。データベースファイルを PHP で使用する場合は、PHP のファイルと同じディレクトリの方がわかりやすいと思いますが、直接データベースファイルをアクセスされると問題があるので注意します。

3.2 テーブルの追加

新しいデータベースファイルを指定すると、データを入れるテーブルを作成する図 3 のような画面が出ます。2 つ以上のテーブルを作成する場合は「テーブルを作成」をクリックすると同じ画面になります。ここでテーブルの名前と列を設定します。

テーブルの名前は①の欄に英数字で入力します。次に②の「追加」ボタンをクリックすると列の設定が一つ入れられるようになります。「名前」のところに英数字による列の名前を入力し、「データ型」は「Text」、「 Integer」、「Real」などから選択します。文字列ならば「Text」、小数点のない数値は「Integer」、小数点付きの数値は「Real」になります。SQLite はデータ型に関してはかなり適当で他のデータベースシステムではもっと細かい設定をすることになります。主キーの列ならば「PK」をチェックします。さらにデータを追加するたびに自動的に異なる数値を入れるならば「デフォルト」のところに「autoincrement」を入力します。この主キーの設定は一つのテーブルにつき一つ以上ないといけません。列の情報を全て追加したら「OK」をクリックしてテーブルの定義を保存します。

表 4 のような設定の「student」と言う名前のテーブルを追加してみましょう。autoincrement を設定する際は「AI」にチェックを付けます。

作られたばかりのテーブルには何も入っていません。DB Browser でテーブルの中にデータを入れたり、中のデータを修正したり、削除したりすることができます。図 4 のように①の「データ閲覧」のタブをクリック

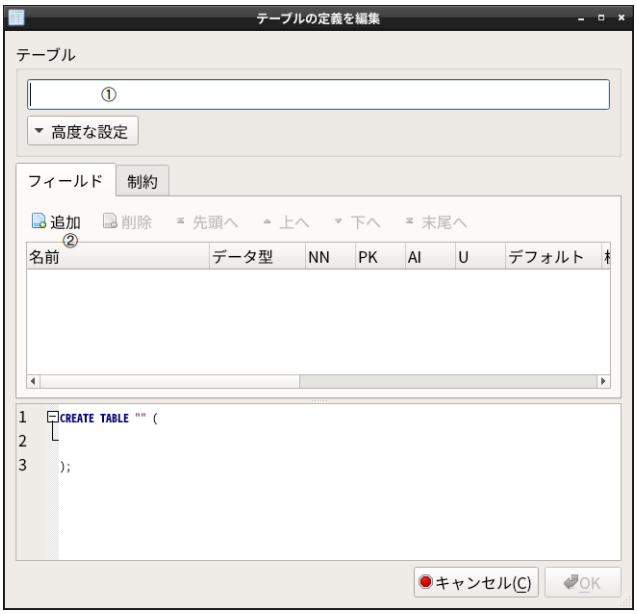

図 3 テーブルの設定画面

表 4 テーブルの例

名前	型	オプション	デフォルト
id	Integer	PK	autoincrement
name	Text		
pass	Text		

して、②のところでデータを扱いたいテーブルを選択すると現在のテーブルの内容が表示されます。修正したいところをクリックすれば内容を変更することができます。また③のボタン[*8]をクリックすると新しいレコードを追加することができます。

autoimcrement の設定がある「id」には数字が自動的に入りますが、それ以外の項目には「NULL」という空を示すものが入ります。修正したいところをクリックしてデータを入れることができます。右隣りの項目に引き続き入力したい場合は Tab を押します。図 1 と同じ内容を入れてみましょう。

データの入力や修正、レコードの追加や削除を行ってもすぐにはデータベースのデータは書き換わりません。図 4 の④の「変更を書き込み」が黒くなっているときはまだ書き換わっていないので、ここをクリックしましょう。クリックするまでは書き換わらないだけでなく、データベースファイルを他のプログラムから触れないようにロックもかけているので注意しましょう。

3.3 SELECT 文

SQL の中で最もよく使われるのが SELECT 文と言われるものです。これによってデータベースの検索が行えます。またここで扱う検索条件は、データの更新や削除の際にも利用します。つまりどのデータを更新したり削除するのかを指定する際に検索条件を設定しなければ、全てのデータが書き換わったり、削除されてしまいます。

まず一番基本的な形の SELECT 文の例は次のようになります。SQL のキーワードとなる語は大文字です。

[*8] このボタンが表示されていない場合は、DB Browser のウィンドウを横に伸ばすか、右の方にある「>>」をクリックすると更にメニュー項目が表示されるので、「新しいレコード」→「新しいレコード」と選択すると新しいレコードが追加されます

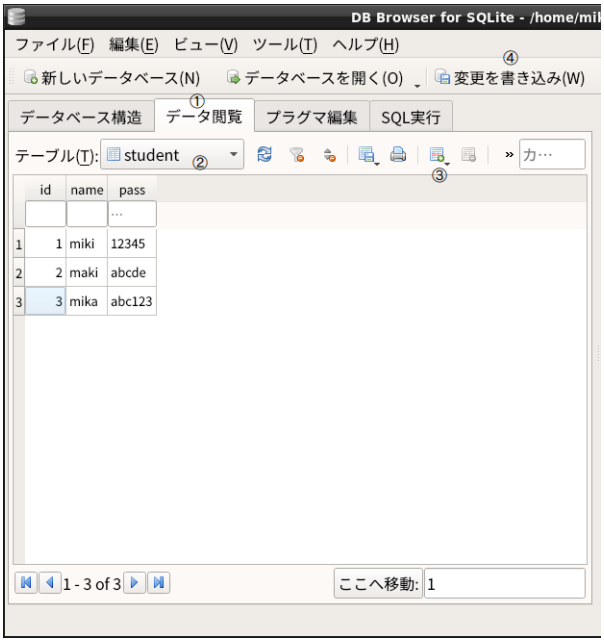

図 4　テーブルのデータ編集画面

```
SELECT * FROM student
```

SELECT と FROM の間には取り出す列の名前 (列名) を並べます。複数の列を指定する場合は「,」で区切る必要があります。この例のように「*」を指定すると全ての列を指定したことになります。FROM の後には使用するテーブルの名前を並べます。この場合も複数のテーブルを指定する場合は「,」で区切る必要があります。

これを DB Browser で実行する際には、図 5 のように①の「SQL 実行」のタブをクリックして、②のところに SELECT 文を入力し、③のボタンをクリックします。この際 SQL のキーワードを小文字で入力しても問題はありません。

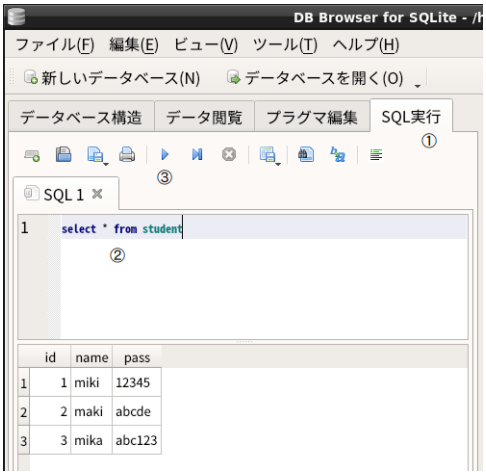

図 5　SELECT の実行

通常はテーブルの全ての列を取り出すような事はしません。pass の列だけ必要であれば次のようにします。

```
SELECT pass FROM student
```

さらに条件を設定することにより、取り出すレコードを限定することができます。

```
SELECT pass FROM student WHERE name='miki'
```

WHERE の後に取り出したいレコードの条件を指定します。この場合は「name='miki'」となっていますので、name の列に miki が入っているレコードのみが出てきます。JavaScript や PHP などと異なり等しい場合は「=」を使います。比較の対象が文字列の場合は「'」で囲います。条件によっては、複数のレコードが出てきたり、全く何も出てこないこともあります。

複数の比較は AND や OR でまとめます。AND は全ての比較が成立した場合、OR はどれか一つの比較が成立すれば良い場合に用います。例えば次のようにすると name が miki で pass が 12345 であるレコードの id のみが取り出せます。

```
SELECT id FROM student WHERE name='miki' AND pass='12345'
```

3.4 PHP による SQL の実行

PHP でデータベースに対して SQL の命令を送り、その結果を変数で受け取ることができます。データベースとしては SQLite は内蔵しているためにいつでも利用可能です。他には MySQL や PostgreSQL などの企業の基幹システムでも使われることのあるデータベースなどにも対応しています。対応方法も各々のデータベース専用の方法と共通の方法が用意されており、共通の方法を使って記述すると、後で容易に別のデータベースシステムに乗り換えることができるようになります。

データベースとの接続

データベースを利用する際には、まずデータベースとの接続を行う必要があります。MySQL や PostgreSQL などの利用者の管理もするデータベースシステムでは、この際に利用者の ID やパスワードが必要になりますが、SQLite の場合はデータベースのファイル名のみが必要です。リスト 1 はデータベースのファイル名が data.db の場合です。

リスト 1 データベースとの接続の部分

```
1  $db=new PDO('sqlite:data.db');
2  if (!$db) { die('接続失敗です。'); }
3  $db->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING);
4  $db->setAttribute(PDO::ATTR_TIMEOUT,10);
5  if (!($db->query("BEGIN EXCLUSIVE"))) { die("Transcation が開始できません"); }
```

1 行目でデータベースファイルを指定しています。開いたデータベースを変数$db に入れてますが、ファイルの指定に誤りがあるような場合は、この変数の中身が NULL になるので 2 行目の if で調べています。「!」で条件を反転しているので、本来 false 扱いとなる NULL の場合に die() を実行します。die() は () 内で指定したものを表示した後 PHP の実行を停止します。

3 行目はエラーメッセージを出すレベルの設定を行っています。

4 行目で実行時間を 10 秒に制限しています。これ以降の SQL の実行に 10 秒以上かかるとエラーになります。

5 行目で「BEGIN EXCLUSIVE」と言う SQL の指示をデータベースに送っています。これはこれ以降の処理が排他的 (Exclusive) な一連の処理 (transaction) であることを示しています。同じデータベースファイルをを他のプログラムが利用したり、同じプログラムが多重に起動されると、データベースの内容の整合性が

取れなくなる可能性が生じます。これはそれを防ぐための処置で、これ以降他のプログラムは同じデータベースファイルには手が出せなくなります。ただ他のプログラムが既にデータベースファイルを使用中だった場合は、その終了まで待つことになります。永遠に待つことを避けるために、4 行目の実行時間制限をしています。

検索の実行

次のようにして「select * from student」と言う SQL の SELECT 文を実行することができます。検索した結果は$result に入ります。SELECT 文に誤りがあった場合は、その内容を表示して停止します。

```
$sql="select * from student";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
```

$result は二次元配列型の変数になります。つまり検索した結果は通常表のような二次元の形になるので、単純な配列では対応できないからです。foreach を利用すると検索結果を 1 レコードずつ取り出すことができます。

```
foreach ($result as $data) {
   $data['id']、$data['name']、$data['pass'] にデータが入っている
}
```

検索結果を全て同じ形で扱う場合はこの方法が良いと思いますが、最初のレコードだけ別扱いと言うような場合は次のようにします。

```
while ($data=$result->fetch(PDO::FETCH_ASSOC)) {
   $data['id']、$data['name']、$data['pass'] にデータが入っている
}
```

while の中にある取り出しの部分を while の前や後で、1 レコードの取り出しとして使うことができます。while の前で使う場合は検索結果が何もない時への対応、while の後で使う場合は while で全て取り出しが終わっている時への対応に注意します。

データベース処理の反映・終了

データベースの検索はデータベースファイルを変更しませんが、後に出てくる UPDATE 文などはデータベースファイルの内容を変更します。しかし実際は$db->query($sql) で UPDATE 文を実行してもまだデータベースファイル自体は変更されません。一連の処理が無事終わった場合は最後に、「COMMIT」を送ることによりデータベースファイルを変更することができます。

```
$db->exec("COMMIT");
```

これを忘れると一見処理は正常に終了しているのに、データベースファイルの内容が変わらないと言うバグに頭を悩ませることになります。

もし他のプログラムが同じデータベースファイルを使おうと待っていた場合は、これで排他的使用が終わるので動きだします。一方処理中に問題が生じてデータベースファイルを更新する必要がなくなった場合は、次のようにして「ROLLBACK」を送ります。

```
$db->exec("ROLLBACK");
```

大抵の場合最後にこれらのどちらかを行いますが、データベースの処理の後に延々と別の処理を行う場合は、その前に COMMIT しましょう。

3.5 INSERT 文

INSERT 文を用いてデータベースのテーブルにレコードを追加します。例えば次のような形です。

```
INSERT INTO student (name, pass) VALUES ('mike','999abc')
```

最初の ( ) の間に内容を入れる列名を示します。複数ある場合は「,」で区切ります。autoincrement の指定によって自動的に入るものや、入っていないことを示す NULL が入ってよい列名は省略して構いません。2 つめの ( ) の間に入れる値を示します。数字や NULL 以外の値は「'」で囲います。順番は最初の ( ) の列名の順番と同じにします。

PHP による INSERT 文の実行は次のようにします。

```
$sql="INSERT INTO student (name, pass) VALUES ('mike','999ab'c)";
if (!($db->exec($sql))) { die("データの挿入失敗: ".$sql); }
```

内容の指定の無い id には autoincrement 機能により、これまで使用した id の最大値 +1 が設定されます。この id の値が知りたいことがよくあります。INSERT 文で入れた内容が他のレコードにはないものであれば、SELECT 文で検索して求めます。SQLite に限定されますが、次のような SELECT 文で管理用のテーブルである sqlite_sequence を検索しても求めることができます。

```
SELECT seq FROM sqlite_sequence WHERE name='student'
```

これで一番最後に実行した INSERT 文で挿入されたレコードの主キーに autoincrement で入れた値がわかります。

3.6 DELETE 文

テーブルのレコードを DELETE 文を使用して削除することができます。削除したものを戻すことは SQL ではできませんし、条件を間違えると一気に多くのレコードが削除されるので注意します。心配な場合はデータベースのファイルのコピーを作っておくとよいでしょう。

例えば次のような形で DELETE 文を使用します。

```
DELETE FROM student WHERE id=3
```

WHERE の条件次第で複数のレコードを削除することも可能です。PHP で次のように DELETE 文を実行した場合、変数`$result` には削除されたレコードの数が入ります。DELETE 文に問題があった場合はゼロではなく false が入るので if の条件では「===」を使用します。

```
$sql="DELETE FROM student WHERE id<3";
$result=$db->exec($sql);
if ($result===false) { die("データの削除失敗: ".$sql); }
```

削除した数が必要ない場合は次のように「`$db->exec($sql)`」を if の条件の中に入れても構いません。

```
$sql="DELETE FROM student WHERE id<3";
if ($db->exec($sql)===false) { die("データの削除失敗: ".$sql); }
```

3.7 UPDATE 文

既に入っているテーブルの中のデータを書き換えるのが UPDATE 文です。レコードの中の指定した項のみ書き換えることができますし、条件次第で複数のレコードの書き換えも一度に行えます。例えば次のような形で name が miki の pass を 5555 に書き換えることができます。

```
UPDATE student SET pass='5555' WHERE name='miki'
```

もし student テーブルの中に miki が複数あった場合は全て書き換えられてしまいますし、逆に miki がなければ何も書き換えられません。DELETE 文と同様に以下のように PHP で実行した場合は、`$resulut` 変数に書き換えられたレコードの数が入ります。

```
$sql="UPDATE student SET pass='5555' WHERE name='miki'";
$result=$db->exec($sql);
if ($result===false) { die("データの更新失敗: ".$sql); }
```

WHERE 以降を省略すると条件が無いので全てのレコードが書き換えの対象になります。また複数の項を書き換える場合は、次のように「,」で区切って指定します。

```
UPDATE student SET name='kiki', pass='5555' WHERE name='miki'
```

3.8 レコードの並び替え

SELECT 文でテーブルの内容を取り出す際に並び替えの指示を追加することができます。例えば name の昇順であれば次のようにします。

```
SELECT * FROM student ORDER BY name
```

もし WHERE があるならばその条件の後に「ORDER BY」の指示を付けます。降順にする場合は次のように DESC を追加します。

```
SELECT * FROM student ORDER BY name DESC
```

name に同じ値を持つレコードがあり、その場合は pass の値で並び替えたいような場合は次のように「,」で区切って複数の列名を指定します。

```
SELECT * FROM student ORDER BY name DESC, pass
```

表 5 money テーブルの例

id	sid	date	amount
1	1	11/27	1000
2	1	11/28	500
3	1	11/29	350
4	1	11/30	500
5	2	11/29	350
6	2	11/29	400

3.9 テーブルの結合

ちょっと複雑なデータを考えると、一つのテーブルでは対応できない事が分かります。例えば学生の住所や電話番号のような一人の学生に一つだけ対応するものであれば、student テーブルに新しい列を追加するだけで対応できます。しかし学生のお小遣いの管理をするために、学生がいつ、いくら使用したかを複数記録したいと言う場合、一人の学生が何回使用するかはわからないために、予め列を追加する方法では対応できません。このような場合、お金の使用記録は別のテーブルで扱います。例えば表 5 のような形です。

money テーブルの id は主キーです。他の列では一意性を保証することはできないので別に設けています。(同じ日に同じ人が同じ金額を使うと他の列の内容は同じになるため。) 次の sid は sudent テーブルの id の値です。最初の 4 件の sid は 1 なので、student テーブルの内容が図 1 のままだとすると、miki さんの記録ということになります。student テーブルの name の値である miki を使わず、student テーブルの id の値を使うのは、student テーブルの id は主キーなので複数のレコードが対応する恐れがないからです。

student テーブルの name と money テーブルの date と amount を元に、だれがいつ、いくら使用したかの一覧を求める SQL は次のようになります。

```
SELECT name, date, amount FROM student, money WHERE student.id=money.sid
```

2 つのテーブルを使用するために FROM の後にテーブルの名前が 2 つ必要になります。そして WHERE のところに「student.id=money.sid」と指定することによって 2 つのテーブルが結合されます。student テーブルの id と money テーブルの sid が同じレコードが結合して残ります。表 5 には sid の値が 3 のレコードがありません。そうすると student テーブルの内容が図 1 のままだと、mika のレコードは出てきません。対応するレコードが無い場合も残したい場合は、外部結合という方法で対応することができますがここでは省略します。

テーブルのどの列が他のテーブルのどの列に対応しているか、と言う条件を複数指定すれば 3 つ以上のテーブルを結合することもできます。ところが複数のテーブルに同じ名前の列名が使われていると、曖昧な条件になってしまいます。それを避けるために「student.id」のようにテーブルの名前を付けます。先程の例では name、date、amount は片方のテーブルにしか出てこないためにテーブル名を省略していますが、両方のテーブルにある id を表示したい場合は、次のようにテーブル名も付ける必要があります。

```
SELECT student.id, money.id FROM student, money WHERE student.id=money.sid
```

これらを PHP で実行する際に一つ問題になるのは、SELECT 文の結果を取り出そうとする際に、student.id のようなテーブル名が付いた列は取り出せない点です。そのために AS を使用して別名を付けます。次の例では money.id に mid と言う別名を付けて取り出しています。

```
$sql="SELECT money.id AS mid FROM student, money WHERE student.id=money.sid";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
foreach ($result as $data) {
   $data['mid'] に money.id の値が入っている
}
```

なおここで説明したテーブルの結合の書き方は古い書き方です。新しい書き方では INNER JOIN を使用して次のように書きます。テーブルの結合の条件が WHERE の検索条件と混ざらないため、わかりやすいと言われますがいかがでしょうか。3 つ以上のテーブルを結合する際には INNER JOIN 以降を繰り返すことになります。

```
SELECT name, date, amount FROM student INNER JOIN money ON student.id=money.sid
```

3.10 レコードのグループ化

テーブルのある項目について、同じ値が複数入っている時に、同じ値ごとに集計したくなることがあります。例えば表 5 の場合、sid が同じ値のものについて amount の合計を出せば、各自が使用した金額が得られます。同じ値でまとめるということはグループにすることとして、次のように「GROUP BY」でまとめて、さらに SQL の SUM 関数を使ってグループごとの合計の計算をすることができます。

```
SELECT sid, SUM(amount) AS goukei FROM money GROUP BY sid
```

関数の結果を PHP で取り出すために AS を利用して goukei という別名を付けています。SUM 関数以外に表 6 のような関数があります。

表 6 SQL の関数

関数の形	関数の働き
AVG(列名)	指定した列の値の平均値を求めます
COUNT(列名)	指定した列の値が NULL 以外の行数を求めます
COUNT(*)	行数を求めます
MIN(列名)	指定した列の値の最小値を求めます
MAX(列名)	指定した列の値の最大値を求めます
SUM(列名)	指定した列の値の合計を求めます

3.11 インデックスの設定

テーブルの列ごとにインデックス (索引) を設定することができます。よく検索対象として使われる列にインデックスを設定すると検索の高速化ができます。インデックスが設定されてない場合、ある列に指定した値が入っているレコードを探すためには、テーブルの全てのレコードを順番に見る必要があるため時間がかかります。テーブルとテーブルを結合する場合も、対応するレコードを全て探す必要があるため大変時間がかかります。インデックスがあれば、テーブルの内容を見なくてもすぐにどこにあるか分かるので、その効果はレコード数が多い場合絶大的ものになります。ただテーブル内のデータの変更に合わせて、インデックスの内容も更新しなければならないので、データの変更が頻繁に大量に行われる場合は、システムへの負担が問題になるかもしれません。

インデックスの設定は SQL でも行うことができますが、ここでは DB Browser で設定する方法を説明します。まず DB Browser でインデックスの設定をしたいデータベースファイルを開き、図 6 のようにまず (1)「データベース構造」のタブを選択し、(2)「インデックスの作成」をクリックします。すると図 7 のような設定のウィンドウが出てきますので、(1) 適当な名前を入力し、(2) のところでインデックスを設定したいテーブルを選択し、(3) のところでインデックスを設定したい列名をクリックして、(4) のボタンで右側に列名を送ります。すると (5) のところの「OK」ボタンがクリックできるようになるので、クリックします。

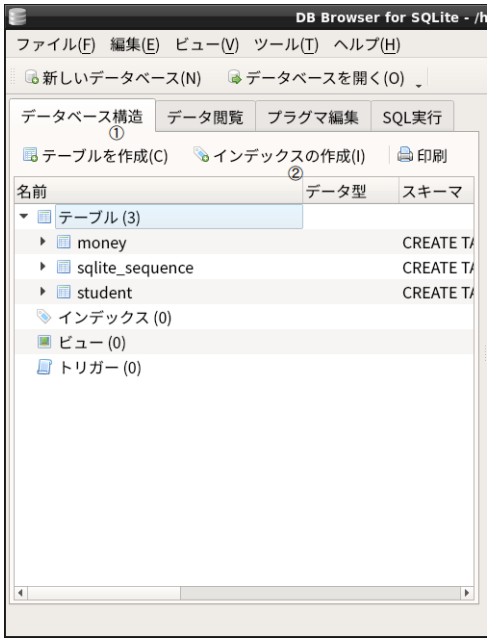

図 6　インデックスの設定の呼び出し

図 7　インデックスの設定画面

4. QR コードの利用

様々な商品に付けられているバーコードは、十数桁の数字を棒の太さや間隔で示したもので、通常会社コードと商品番号を合わせた数字を示しています。コンビニなどではこれをレーザー光などを利用して読み取り、データベースに照会して商品名や価格の情報を得て、レシートに印字します。バーコードの形と数字の内容が規格化されており、パッケージなどにバーコードを印刷するだけで安く使用できるのと、読み取り機が安価なため、広く使われています。ただバーコードで表現できる内容が数字で十数桁しかないために、個々の商品に異なる番号を振る事ができませんし、情報量が少なすぎるので商品番号以上の商品に関する情報を含めることもできません。QR コードは、1994 年（平成 6 年）に自動車部品メーカーであるデンソーの開発部門が開発したマトリックス型二次元コードです。デンソーは QR コードを広く普及させることを考えて、特許をオープンにしました。最大約 7 千桁の数字が表現できる他、用途に合わせて誤り訂正レベルが設定でき、スマホなどで読み取り可能なため、現在 QR 決済などで様々な分野で広く使われています。

この章ではこの QR コードを扱う方法の説明や、簡単な QR 決済のシステムを作成します。

4.1 PHP による QR コードの出力

Web ページに QR コードを表示させる場合、いつも同じ内容の QR コードであれば、QR コードの画像ファイルを用意して、HTML の Img タグでその画像ファイルを指定すれば可能です。一方お客さんにお互いに異なる座席を指定する電子チケットのような QR コードは、全て異なるものでないと困ります。そしてそれを予め全て画像として用意しておくより、必要に応じて作成する方がよいでしょう。

画像を作成できるプログラムが書けるのであれば、基本的に QR コードの画像を生成するプログラムを書くことはかのうです。しかしそのプログラムを書くためには QR コードの細かい仕組みを理解する必要があり大変です。幸いなことに様々なプログラミング言語用の QR コード生成プログラムが公開されているので、それを利用させていただくことにします。基礎演習のテキストの AR システムでは JavaScript によるものを使用していますが、PHP 用の QR コードを生成するプログラムの一つをここでは紹介します。

プログラムの作者は Y.Swetake 氏で https://www.swetake.com/で公開されています。このサイトからダウンロードしたものはまとめて圧縮されているので、解凍・展開をします。出てきたファイルの中で必要なものは、php ディレクトリの中の `qr_img.php` と data ディレクトリと image ディレクトリです。この 3 つを QR コードを表示する HTML や PHP のファイルがあるところにコピーし、`qr_img.php` の最初の方にある以下の部分を書き換えます。

```
$path="./../data";           /* You must set path to data files. */
$image_path="./../image";    /* You must set path to QRcode frame images. */
       ↓
$path="data";           /* You must set path to data files. */
$image_path="image";    /* You must set path to QRcode frame images. */
```

そして QR コードを表示したいところに、以下のように HTML の Img タグを記述します。

```
<Img src="qr_img.php?d=http://www.sugiyama-u.ac.jp/&e=L&s=3">
```

- d=で QR コードで表したいデータを指定します。例のように URL でも良いし、単なる数値などでも構いません。漢字は読み取り側がどのような漢字コードと解釈するのか明確でないので避けましょう。
- e=で誤り訂正のレベルを指定します。L、M、Q、H のどれかを指定します。L が一番誤り訂正レベルが低く、H が一番誤り訂正レベルが高くなります。訂正レベルが高いほど QR コードの汚れなどに強く

なりますが、図 8 を見ても分かるように、QR コードが大きくなります。何も指定しない場合は e=M を指定したことになります。

- s=で QR コードの大きさを指定します。1 の場合が一番小さくなります。何も指定しない場合は s=4 を指定したことになります。

パラメタを変えると図 8 のように QR コードを表示することができます。QR コードで表しているのは全て同じ「http://www.sugiyama-u.ac.jp/」です。

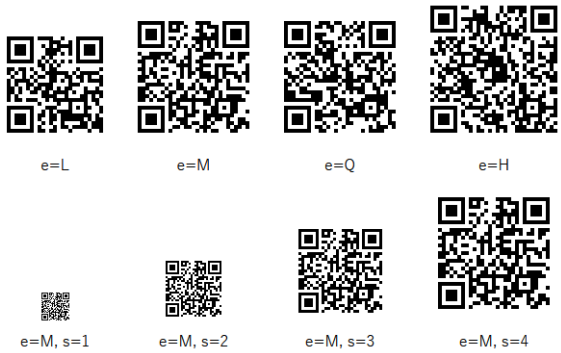

図 8　QR コード出力の例

4.2　QR コードによる決済

我々がお店で何か買い物をする際には代金を現金で支払うのが普通でした。支払いが高額な場合現金を持ち歩くのは危ないので、クレジットカードによって支払うという方法が考えられました。クレジットカードを見せて、サインすると支払いの手続きは終わります。お店はクレジットカードの番号を元にクレジット会社に請求し代金を得ます。クレジット会社は銀行の口座からお店に渡した代金を引き落とします。銀行から代金を引き落とすためには手数料が必要になります。その手数料はクレジットカードの持ち主のカード代から支払いたいところですが、クレジットカードの持ち主がクレジットカードをどんどん利用すると足らなくなります。またお店に代金を送るためにも通常お店の銀行口座に振り込むという形になります。よってお店から数 % のクレジットカードの利用料を取ります。お店の売上がその分減りますので、クレジットカードの使用ができない店やクレジットカードの支払いの際には割引ができない店があります。

スマホの決済や QR コードによる決済も基本的にはクレジットカードと同じです。QR コードによる決済の利点は、お店側に高価な設備が必要でないところでしょう。スマホで済ませることができます。お店にとっての問題はクレジットカードと同様に利用料を取られるところです。銀行の手数料がある限り、代金の大きさに関わりなくある程度の手数料を取られます。この問題を解決するのに一番簡単な方法は、お店や利用者の口座を QR コードの決済システムの中に持つことです。同じシステムの中で数字が動くだけですので、コストはほぼゼロになります。現在の日本ではいくつかの決済システムが覇権を争っているところです。大きくなるほどシステム内でのお金のやり取りになるのでコストが減ります。赤字覚悟で利用者に還元しても将来取り返せるということでしょう。他社が無くなっていれば、取り返すために手数料を上げても逃げられませんから。

クレジットカードの利用はある程度高額な支払いでしたが、QR コードによる決済などではより少額の支払いにも使われます。誰がいつ何を購入したかがわかると、売る側は次に何が売れるか予想を立てることができます。そういう情報の収集にも役立つと考えているようです。

ここでは次のように使う QR コード決済システムを作ってみましょう。

1. お店の人が決済システムにアクセスすると、既に認証済みであれば自分宛の支払いの QR コードと、入出金リストが表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると

同じものが表示されます。

2. お客がお店の人の QR コードを元にアクセスすると、既に認証済みであれば、残額が表示された支払額の入力画面が表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると同じものが表示されます。
3. お客が支払額を入力して、支払い処理をしたら、お店の人は入出金リストを更新し、支払いがあったことを確認してから商品をお客に渡します。

お客に QR コードを見せる際に入出金リストが見えてしまいますが、この QR コードの示す内容は変化しないので、QR コードの部分のみ取り出して印刷しておけば良いでしょう。これらを実現するために図 9 のような構成にします。

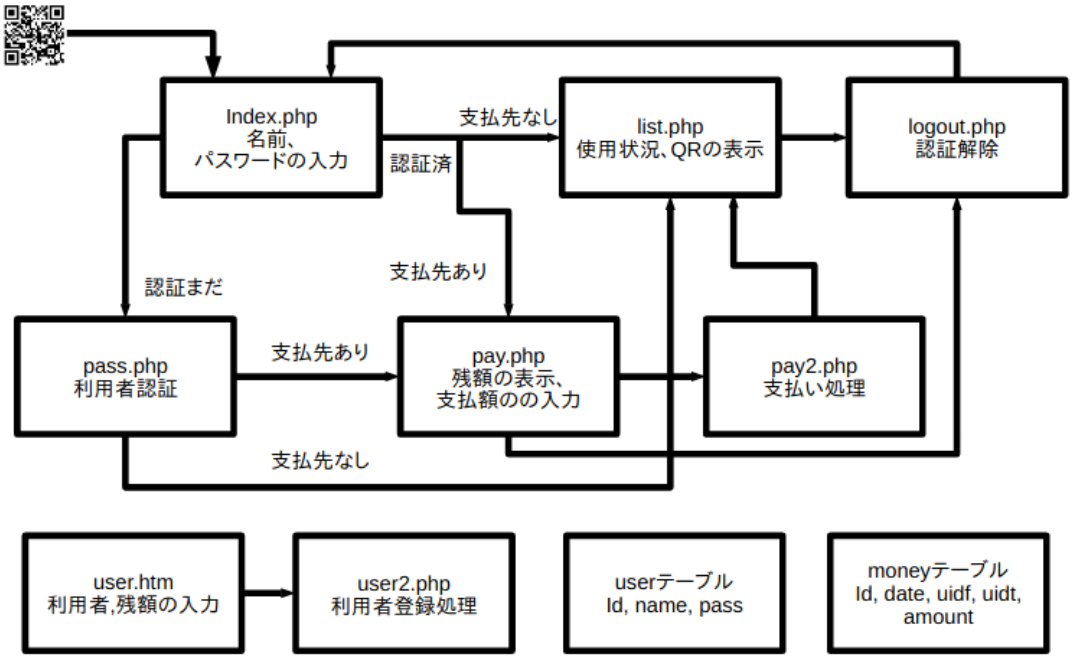

図 9 QR コード決済システムの構成

以下は各部分の説明です。

- user テーブルで、名前、パスワードを管理します。
- money テーブルで、いつ、誰が、誰に、いくら送金したか記録します。
- 決済システムに入る際には利用者認証を行います。session の情報がある程度の期間残るようにして、残っている場合は利用者認証を省略できるようにします。
- user.htm で入力した内容を user2.php で user テーブルと money テーブルに登録することにより利用者登録をします。本来ならばきちんと認証をする必要がありますが、ここでは省略しています。
- list.php では、自分宛の支払いの QR コードと入出金リストが表示されます。
- お店の人が index.php にアクセスした場合、既に認証済みであれば list.php へ、もし認証がまだであれば index.php で名前とパスワードを入力し、pass.php で認証を行い user ターブルの内容と合えば list.php へ行きます。
- お客がお店の人の QR コードを元にアクセスすると、index.php へ行き、既に認証済みであれば pay.php に行きます。もし認証がまだであれば index.php で名前とパスワードを入力し、pass.php で認証を行い user ターブルの内容と合えば pay.php へ行きます。
- index.php では、認証済みかどうかと支払先の指定があるかどうかで行き先が変わります。また

pass.php へ行く際は、支払先の情報も引き継ぐ必要があります。

- pay.php では、お客に残額を表示し、支払額を入力してもらい、支払いの指示をすると pay2.php へ行きます。
- pay2.php では、お客からお店へ支払額が動いたと言うデータを money テーブルに追加し、user テーブルの残額も修正します。そして list.php へ行きます。
- 支払い処理が終わったら、お店の人は list.php を更新し、支払いがあったことを確認します。
- logout.php では、認証済み情報を削除します。これによって共有の端末で使用した場合に、他の人に使われてしまうことを防ぎます。
- 図 9 にはありませんが、データベースを利用する際に必要になる部分を common.php に入れて共有します。