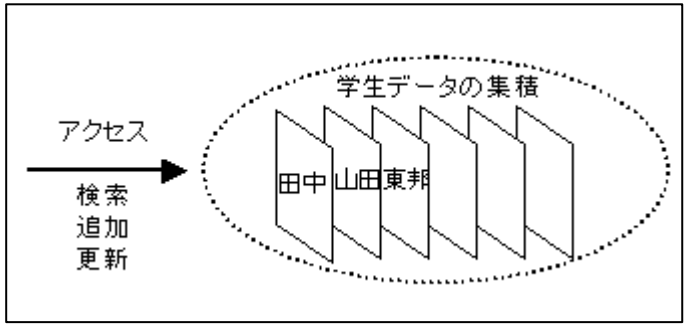
第６章　　データベースの基礎知識

さて、ここからはデータベースを使ったホームページ処理を学ぶ。 まず最初に、データベースが何であるか、データベースをどう使うのか、考えて見ることにする。

データベースとは



データベースとは、同じ形（イメージとしてJava言語でのクラス、C言語の構造体）のデータ（Javaでのインスタンス）がたくさんあるときに、それを効率よく格納し、検索する仕組である。

例としては、学生１人のデータとして名前、生年月日、学籍番号、住所、入学年度があるとしよう。形は、たとえば名前は最大１６文字の文字列、生年月日は年・月・日の3つの整数、学籍番号は7桁の整数、住所は５０文字の文字列、入学年度は整数、を含むクラス（または構造体）として考えることが出来る。

このような形を持ったデータが 5000 人分あるとしよう。Java やC言語のプログラミングで出てくるようなクラスや構造体の配列を定義するのは１つの方法だが、格納も検索も効率が悪いことが多い。データの個数が非常に大きくなると（例えば１万件とか１０万件とか）、検索するのに効率の悪いやり方をしていては、とても実用に耐えなくなるだろう。 かと言って、効率のよい優れたプログラムを毎回作成するのも大変である。

そこで、大量の（同じ形の）データを効率よく扱える仕組として、データベースが用いられている。データベースは、大量のデータを扱うことを目的にして専用に設計された、共通で使える仕組（ミドルウェア）といってもよいだろう。プログラマは、このような仕組を使うことによって、アプリケーション毎に効率向上の工夫をしなくても、効率よいアクセスが出来るようになる。

正確には、「データベース」と「データベース管理システム（DBMS）」は区別して扱わなければならない。「データベース」はデータの集まりそのものを指す。 「データベース管理システム」（DBMS）はデータベースを入れる入れ物（上に説明した効率よいアクセスを実現するためのソフトウェア）を指す。つまりデータベースはDBMS上にデータを集めたものです。たとえば、過去の気象のデータベース、農作物の生産量のデータベース、人口のデータベースなどと呼ぶことが出来る。これらのデータ集合を、DBMSで管理すると効率がよい、ということになる。

データベースでデータ間の関連を表す必要がある。そのための表し方にはいくつかのモデル（データモデル）がある。現在広く使われているのは「関係モデル」で、データ同士の関連を「関係」（relation）として表す。具体的には表の形式で表現される。別のモデルとしては「階層モデル」や「ネットワークモデル」が有名である。この詳細はデータベース論の講義に譲ることにし、ここでは深く追求しないことにする。

名前	生年月日	学籍番号	住所	入学年度
田中太郎	1992/9/9	5510950	船橋市三山. . .	2010
山田一郎	1992/5/5	5510951	習志野市藤崎. .	2010
東邦花子	1993/3/3	5510952	千葉市美浜区. .	2010
佐倉純一	1991/7/7	5510953	柏市. . .	2010

ここでは、広く使われている「関係モデル」によるデータベースを使うことにする。 関係モデルでは、データを表の形式で記述する。 右図は、上記の学生のデータベースを表形式で表した例で、学生１人が１行のデータになっており、その中に名前や生年月日などの情報が横に並んでいる。

データベースをどう使うのか？

データベースに対する操作は、データの挿入・削除や検索などが中心になる。関係モデルのデータベース（関係データベース）では、データ操作のための SQL と呼ばれる言語が標準化されており、広く使われている。SQL に従ったデータベースを SQL データベースと呼ぶことがある。このプロジェクトでは、広く使われている SQL データベースを使って、ホームページから登録されたデータを格納したり、ホームページから与えた条件で検索したりする。

SQL データベースにはいくつも製品がある。商用で有料のソフトでは Oracle 社や IBM 社のデータベース管理システムが有名なほか、 Microsoft 社の Windows サーバー上で動く製品もある。 またフリーのソフトもいくつか出回っており、PostgreSQL や MySQL などはかなり広く使われている。いずれも、標準化された SQL 言語を介してデータにアクセスするので、どの製品でも同じようなものだが、それぞれに拡張が成されていたり、運用管理上の手段がいろいろと提供されていたり （これは SQL 標準には含まれない）、性能や安定性にも差がある （これは同じソフトでもバージョンによって異なる）。 プロジェクトでは、データへのごく簡単なアクセスを、標準的な SQL の文で実現するので、どの種類の SQL データベースで利用できるが、製品固有の拡張を使うと、他の製品では使えないということも起こる。

実際に SQL データベース（データベース管理システム DBMS）を使うには、

（1） 表を作る（定義する）
（2） 表にデータを挿入する
（3） 表を検索し欲しいデータを抽出する
（4） 必要なら表の中のデータを更新する（書き換える）

といった操作をしなければならない。

たとえば、上に示した学生データを管理する場合、まず最初に表を作らなければならない。表を定義するためには、各々の「列」（正式にはフィールドと呼ぶ）のデータの形の定義を DBMS に伝える必要がある。たとえば、「名前を書く欄は name という名前

で呼ばれ、最大 16 文字の文字列データが入る」 といったことである。すべての列について定義する必要がある。この表を作るための SQL の構文として、CREATE TABLE がある。これによって1つ表を作ることが出来る。表には名前をつけておく。また引数として、表の列（フィールド）の名前や型を指定する。なお、表の操作にはこの他に列（フィールド）の変更（ALTER）や表自体の削除（DROP TABLE）などがある。

次に、今作成した表にデータを挿入する。ここで言うデータの挿入は、表の上では行を1つ追加することに当たる。 表の行のこと（たとえば学生データベースにおける山田太郎のデータ）を「レコード」と呼ぶ。 データ（行、レコード）の挿入は INSERT INTO 構文で行う。引数に、対象とする表の名前と挿入する値を与える。1行追加する時に、すべての列の値を与える必要はない。値を与えていない列は、値が決まっていない状態のままとなる。

表にたくさんのデータ（行、レコード）が入ると、その表を検索して特定の行（レコード）を取り出すことが必要になる。

（注意： データベースでは、行（レコード）がどういう順番で入っているか、つまり何行目に入っているかは問題にしない。その代わり、特定の条件に合う行（レコード）を検索して取り出そうとする。）

学生データの例で言えば、

名前が山田一郎の学生のレコードを取り出せ　とか
学籍番号が 5510234 の学生のレコードを取り出せ　とか
生年月日が XX から YY までの学生のレコードを取り出せ

などのように条件を指定して取り出す。 この操作を、SQL では SELECT という構文を使って行う。

たとえば、

```
SELECT * FROM students WHERE name='山田一郎'
```

とすると、表 students の中から、列名（フィールド名） name が値 '山田太郎' である行（レコード）を検索・抽出し、マッチする行のすべての列（フィールド）を表示せよ、という意味になる。

表の中のデータを更新したい場合は、UPDATE 構文を使う。引数で、どの表の中の、どの列（フィールド）の値を書き換えたいかを指定した後、対象となる行（レコード）をWHERE 節で指定する。 WHERE 節の書き方は検索と同じで、条件を満たす行（レコード）を検索し、その検索で当てはまった行に対して、指定された列（フィールド）を指定された値に書き換える、という操作をします。たとえば、山田太郎の生年月日を修正するときには、SQL の上では name が山田太郎である行を探して、その行に対して生年月日をZZに置き換える、というような操作の仕方をする。

この他に重要な操作として、表の結合がある。
例として、上記の学生の表 students の他に、もう1つ点数の表 score があるとしよう。点数の表 score は、学生番号の列（フィールド）id と、点数の列（フィールド）ten を持っているとする。そこで、表 students と表 score を同時に見ることを考える。たとえば「'山田一郎の点数は？」という問い（つまり名前が山田一朗である学生の点数は？という問い）に対して、まず表 students を引いて山田一郎の学生番号が 5510951 であることを求め、更に表 score を引いて学生番号が 5510951 である学生の点数を求めることになる。これを1回で自動的にやってしまおうというのが「結合」である。この例では、

```
SELECT score.ten FROM students, score
    WHERE students.name='山田一郎' AND students.id=score.id
```

のようにする。SELECT の後ろの score.ten はこの検索の出力を「表 score 内の列 ten」にせよ、と指示している。表の名前. 列の名前で書かれている。また最後の students.id=score.id は結合の条件で、表 students の id と表 score の id は同じものとみなせということを指示している。

SQL データベースを使えるようになるためには、SQL の文と機能をひととおり学ぶ必要がある。ここではすべての機能を詳細に説明する余裕がないので、簡単な例を試すだけにし、細かい点はデータベースの講義と自習に譲ることにする。特に、値に関わる機能（この列の値は正でなければならない、とか、この列の値は指定していなければデフォルト値として Z をとるとか）や、行（レコード）をユニークに識別できる「キー」としてどの列を使う（たとえば、学生番号は重複が無いのでキーに出来る）とかの機能は、細かくなるので省略するが、重要である。自分で勉強して欲しい。

参考書として、山本森樹 「体系的に学ぶデータベースのしくみ第2版」 日経BPソフトプレス ISBN978-4-89100-66505 1900 円 を挙げておく。 筆者がデータベースの講義を担当したときの教科書だが、これにこだわる必要もない。

さて、このような SQL の「文」が使えるとして、実際にどのように使うのだろうか？
MySQL の場合、データベースのアクセスは、
(1)プログラムからのアクセス、と
(2)端末から手でコマンドを打って（インタラクティブに）アクセスする、
の2つが提供されている。 前者はこれから作るサーバーのデータの保管所として action の PHP プログラムからアクセスする場合に当たる。 後者は、1つにはデータベースを理解するためにいろいろと試してみるのに使ってみるのと、サーバーの開発中にデータベースの状態を確認するために使うことができる。

終了の確認

この章は、終了の確認はありません。
第2回目の授業で、最低限ここまで終了していることが望まれます。
時間がある人は、どんどん次の章に進んでください。

第７章　　端末から MySQL を使ってみる

SQL を使ってみる

ここでは、端末からコマンドを入力することで、SQL を使ってみる。

SQL データベースは、サーバーのシステムとして常時動作しており、それに対してインタラクティブなアクセスのやり方（どうやって起動するか）は、データベースシステム（DBMS）の製品によって異なる。このプロジェクトで用いる MySQL （ホームページ http://www.mysql.com 参照）では、DBMS が実行されるコンピュータ（本プロジェクトでは venus）の文字端末の上で操作する。

≪venus サーバーの文字端末の使い方≫
この章での作業を行うために、venus サーバーの文字端末を、手元の PC 上に開く。文字端末上では、文字コマンドを入力すること、実行結果を文字で表示することができる。 他のマシン（サーバー）の端末を手元 PC に開く使い方を、「遠隔端末」（リモート端末、リモートログイン）などと呼ぶ。 まずその使い方を簡単に説明する。

venus の場合、ssh と呼ぶ遠隔端末の機能が使える。（よく言及される「telnet」はセキュリティ上問題があるのでネット経由では使ってはならないし、禁止している。）
この ssh の機能を Windows 端末 PC 上で使えるプログラムはいくつか存在し、実習室によってインストールの具合が違う。具体的には、
①TeraTerm （TTSSH）、
②putty、
③Cygwin でコマンド ssh が使える場合、
の3つがあり、基本的な機能は同じである。 自宅で試す場合は①が使いやすい。

どれを使うにしても、venus サーバーへログインするためのユーザ ID とパスワードが必要である。 これは Windows 端末のユーザ ID・パスワードと同じになっている。（同じ ID 管理データを使っている）

① TeraTerm については
http://www.ex.media.osaka-cu.ac.jp/windows/teraterm.html
を参照。元々日本人が作ったので日本語にフル対応。ダウンロードは
http://sourceforge.jp/projects/ttssh2/releases/から。
なおネットで検索すると古いバージョンのページがあるので注意。最近の開発は sourceforge サイトで行われている。
② PuTTY については、たとえば
http://www.ex.media.osaka-cu.ac.jp/windows/putty.html
を参照。元々英文用のソフトだが、日本語化パッチを適用済みの PuTTY があり、それを使うことが出来る。
③ Cygwin で ssh が使える場合。 cygwin を起動後、プロンプト（入力促進）に対して
ssh venus.is.sci.toho-u.ac.jp
のように ssh コマンドをタイプインして Enter を打つと、SSH が起動する。
但し cygwin のインストール状況によっては「ssh が無い」と言われる場合があるが、これは cygwin 環境で ssh がインストールされていない場合である。 なお、Cygwin の ssh では日本語（かな漢字）での入力・出力は出来ないが、このプロジェクトの範囲内では日本語入出力は使わないのがよい。

Putty や TeraTerm の場合、起動するとウィンドウが開き、接続先を指定する画面が出るので、
venus.is.sci.toho-u.ac.jp
を指定し、接続モードに SSH を指定する。

いずれの端末プログラムでも、一番最初に venus に接続したときに、接続してよいかどうか尋ねられるので yes と答える必要がある。その後は、venus のログインIDとパスワード （端末の Windows のログイン ID とパスワードと同じ） を入れて、ログインする。

≪SQL データベースをインタラクティブモードで使う ～ mysql コマンドの起動≫
ここまでで venus の Linux に遠隔ログインできたはずである。 この遠隔端末に対してコマンドを打ち込むことによって、mySQL を使う(データベースにアクセスする)ことができる。 それには、venus のコマンド入力として、ユーザコマンド

```
mysql
```

を次のように打ち込むことによって起動する。

```
mysql -u 55????? -p (ENTER キーを押す)
```

但し、 55?????が mysql のためのユーザ ID である。

Mysql では Linux のユーザ ID＋パスワードとは別に、mysql 用のユーザ ID＋パスワードを持つ。 この mysql 用のユーザ ID＋パスワードは、プロジェクトの開始時に別途配布したプリントを見ること。

-p の後に Enter キーを押すと、次の行に

```
Enter password:
```

と出るので、password: の後にパスワードを打ち込む。

```
Enter password: (パスワード) (ENTER キーを押す)
```

(パスワード)が mysql の為のパスワードである。

(注意) 打ち込んだパスワードは画面上に何も表示されない。 表示されない状態で正しく打ち込む必要がある。 バックスペースキーなどによる修正は、見えない状態なので、かなり難しい。 まちがえたら、そのまま ENTER キーを押して一旦「間違い」とし、新たに打ち直すほうがよいだろう。

パスワードがOKであれば、

```
Welcome to the MySQL monitor.  Commands end with ; or ¥g.
Your MySQL connection id is 4
Server version: 5.0.77 Source distribution
Type 'help;' or '¥h' for help. Type '¥c' to clear the buffer.
mysql>
```

という画面が(文字端末上に)現れる。これ以降、mysql> という入力促進記号の後に、mysql のコマンドをタイプすることが出来る。

≪いくつかの SQL 操作コマンド≫
mysql の「コマンド」にはいろいろとあるが、よく使うのは
(1) SQL 文そのものの後ろに「;」(セミコロン)を置いたもの、
(2) use コマンド、これは「ここから指定したデータベースを使う」というコマンド、
(3) mysql とのやり取りを終了する quit コマンド、
の3つであろう。その他のコマンドは help;と入力するとヘルプが表示される。

use コマンドをタイプインすると、

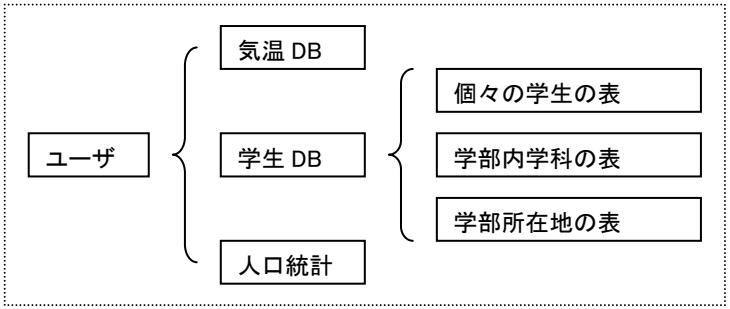
```
mysql> use test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

となるし、quit コマンドをタイプインすると

```
mysql> quit
Bye
```

のような表示が出て mysql とのやり取りは終り、UNIX の入力促進記号(ここでは%記号)が表示され、UNIX コマンドを受け付けるようになる。

use コマンドについて説明しておこう。
mysql では、同じユーザ ID の下で、複数の異なるデータベースを持つことができる。「1つのデータベース」とは、意味のある表の集まりのことで、たとえば私一人が、気温の記録のデータベースと、学生のデータベースと、人口統計のデータベース、というように複数の(この場合3つの)異なるデータベースを、同じ mysql の DBMS の下に持つことができる。use コマンドは、「これから私が定義した学生のデータベースを使いますよ」という宣言になる。 なお、1つのデータベースの中にさらに複数の表をもつことができる(持つのが普通である)。



［例題演習７－１］ 起動から使い始めるまで

早速、mySQL のインタラクティブセッション（端末からコマンドを入力してデータベースを操作するやり方）を使って、データベースを経験してみることにしよう。

venus 上の UNIX にログインし、コマンド mysql を起動する。 ここで使う**MySQL 用のユーザ ID とパスワード**は、プロジェクトの１回目に配布したプリントを参照。

```
% isvenus[yamanouc]%mysql -u 55***** -p (ENTER キーを押す)      が MySQL 用の
ユーザ ID
Enter password:      (ENTER キーを押す)      が MySQL 用パスワード

Welcome to the MySQL monitor.  Commands end with ; or ¥g.
Your MySQL connection id is 4
Server version: 5.0.77 Source distribution

Type 'help;' or '¥h' for help. Type '¥c' to clear the buffer.

mysql>
```

次の手順に沿って、mySQL データベースを使ってみる演習をせよ。 ここから先の SQL 文（SQL コマンド）の意味については、マニュアル

http://dev.mysql.com/doc/refman/5.6/ja/

を見て確かめること。 大半は翻訳されている。 但し一部の章は翻訳されておらず英文のままになっている。英文も読めるようになって欲しい。

使うデータベースを宣言する

まず、利用するデータベースを、use コマンドで宣言する。データベースはあらかじめ DBMS 内に存在しなければならない。システム管理者が、あらかじめデータベースを、あらかじめ作ってあるとする。（データベース名は小文字 d のあとに学生番号）

（場合によっては自分でデータベースを作らなければならないこともあるだろう。マニュアルで CREATE DATABASE を見よ。 venus 上では、学生ユーザはデータベースを作る権限を与えられていないので、作ることは出来ない。

use d5511999 （データベース名 d5511999 は例）

が指定されたデータベース名

use 文によって、これから使うデータベースの名前が d5511999（d5511999 は例）であると宣言する。実行結果は

```
mysql> use d5511999
Database changed
```

のようなメッセージが表示される。

ところで、このシステム（DBMS 全体）にどんなデータベースが登録されているか見てみたくなったら、SHOW DATABASES という SQL コマンドが使える。

```
show databases;
```

最後にセミコロン「；」を打ってから、Enter キーを押す。

セミコロン「；」は mysql でインタラクティブにコマンドを入力する時に、 SQL 文の最後を示すために付ける区切り文字である。 セミコロンを忘れると、「更に次の行を入れて欲しい」、というモードになる。（下記参照）

```
mysql> show databases
    ->                                  （この行が表示される）
```

このときは、次の行にセミコロン「；」だけを書いて Enter を押せば、それでその行の入力が終りになる。

気がついたかもしれないが、文中で SQL コマンドを示すのには大文字で SHOW DATABASES と書いているのに、例では小文字で show databases と書いている。実は、これは「どちらでもよい」。 マニュアルでは大文字で書いてあるが、少なくとも

mySQL では、SQL コマンドに関しては区別しないことになっている。 但し、その他の文字の部分、つまり、表の名前や、フィールドの名前などは、きっちり区別するので、注意が必要である。

show databases の実行結果は、

```
mysql> show databases;
+----------+
| Database |
+----------+
| d5511999 |
| mysql    |
+----------+
2 rows in set (0.00 sec)
```

といったメッセージが出力される。

表(テーブル)を作る

次に、自分のデータベース(たとえば d5511999)の中に、表を作って見よう。

表を作る SQL 文は、CREATE TABLE 文を使う。 ここでは試しに、mytest という名前の表を作ってみよう。

```
create table mytest ...
```

ちょっと待って欲しい。CREATE TABLE では、その表を構成する列(欄、フィールド)の名前と、型(属性)とを教えてやらなければならない。
では例として、学生の名前と学生番号と年齢を含む表を作ってみることにしよう。

学生の名前のフィールドは「name」という名前で20字の文字列 CHAR(20)型とする。

学生番号のフィールドは「id」という名前で6文字の文字列 CHAR(7)型とする(整数型にしてもよいところだが、ここでは文字型にしておく)。

年齢のフィールドは「age」という名前の整数 INT 型とする。
型の種類は、Java やC言語などと違う点がある。どのような型があるのかは、MySQL マニュアルの6．2節を見て欲しい。

では、CREATE TABLE を実行してみよう。name は CHAR(20), id は CHAR(7), age は INT とした。

```
mysql> create table mytest (name CHAR(20), id CHAR(7), age INT);
Query OK, 0 rows affected (0.05 sec)
```

これによって、表 mytest が作られた。
SHOW TABLES 文は、このデータベースの中にどんな表があるかの一覧を表示してくれる。 試してみよう。 今はまだ表が mytest の1つだけである。

```
mysql> show tables;
+------------------+
| Tables_in_d5511999|
+------------------+
| mytest           |
+------------------+
1 row in set (0.00 sec)
```

次に、表 mytest がどのようなフィールドから構成されているのか、忘れてしまった時に、確認する方法を紹介しておこう。 DESCRIBE 文を使う。以下のように「DESCRIBE 表の名前」とすると、その表のフィールドの名前と属性(型)を表示してくれる。Null, Key, Default, Extra の部分は、ここではあまり説明しないので、マニュアルや SQL の参考書を読んで欲しい。 では試してみよう。

```
mysql> describe mytest;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| name  | char(20) | YES  |     | NULL    |       |
| id    | char(7)  | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

表を検索する

表 mytest が出来たので、さっそく使ってみよう。
まず第1にできそうなことは、表の検索である。 前にも見たが、SELECT 文を使うと検索できる。 例えば、

「表 mytest に含まれるすべてのフィールドを検索表示せよ、ただし検索条件は無しなので、すべての行を表示せよ」

という文は

```
mysql> select * from mytest;
Empty set (0.00 sec)
```

のように書く。
SELECT の*は「すべてのフィールドを選べ」という意味である。もし特定のフィールドだけを書き出したければここにフィールド名を'name', 'age'のように指定する。
FROM のあとの mytest は検索の対象となる表の名前である。
いずれにせよ、この表はまだ何も行(レコード)を挿入していないので、どう検索しても「空」(Empty set)という返事が返ってくる。

データ(行・レコード)を挿入する

いつまでも空では仕方ないので、表 mytest にデータを挿入しよう。
挿入は行(レコード)を単位にして行う。 追加には INSERT 文を使う。

```
mysql> insert into mytest values ('tanaka ichiro','5511950','19');
Query OK, 1 row affected (0.00 sec)

mysql> select * from mytest;
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950|   19 |
+---------------+---------+------+
1 row in set (0.00 sec)
```


INSERT INTO の後に表の名前 mytest を書き、その後に VALUES と書いたあと、括弧にくくられた1行分のデータを書く。
1行分のデータは、コンマで区切られた、順に並んだフィールド(この場合3つのフィールド、name, id, age)の値である。個々の値は、データであることを示すために1重の引用符「'」で囲んである。

(注意: 名前のフィールドの値を漢字にしたい時は、正しい漢字のデータを正しい形で与えてやれば、mySQL 自体は漢字データでも格納できる。 残念ながら、インタラクティブモードの場合、UNIX 端末の漢字に対する特性があるので、そのままではうまく入らないことが多い。後で試すように、PHP などのプログラムからデータを挿入する場合は、漢字でもそのままうまく入る。)

挿入した直後の表の内容を、SELECT * FROM MYTEST として表示してみると、上のように正しく挿入されているのが分かる。

これによって、次々とデータ(レコード)を挿入してゆけば、データベースが完成する。
例としてもう2つデータを挿入してみる。そのために INSERT 文をさらに2回繰り返す。

(注) 前と同じコマンドを入力するのに、「↑(上向き矢印)キー」が使える。「↑キー」を1回押すと直前の入力が出るし、繰り返し押すとさらに前の入力が出る。出たところで、「←(左矢印)キー」でカーソルを好きなところへ戻して、入力を書き変えることができる。 バックスペースキーを使えば1文字ずつ消すこともできる。 すべて整ったところで ENTER キーを押せば、それが新しい入力になる。

異なるデータ (yamada taro と toho hanako) を2つ追加したところで、表を表示してみると、レコードが3つになっていることが分かる。

```
mysql> insert into mytest values ('yamada taro','5511951','20');
Query OK, 1 row affected (0.00 sec)

mysql> insert into mytest values ('toho hanako','5511952','19');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from mytest;
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950|   19 |
| yamada taro   | 5511951|   20 |
| toho hanako   | 5511952|   19 |
+---------------+---------+------+
3 rows in set (0.00 sec)
```

表の検索 その2

このように3つのデータが入っている表を対象にして、いろいろな条件を付けて検索をしてみよう。
まず、フィールドの値がある条件に一致するデータ(レコード＝行)を選択・表示するには、SELECT 文で WHERE 節を書く。
その書き方は、

```
mysql> select * from mytest where name='tanaka ichiro';
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950|   19 |
+---------------+---------+------+
1 row in set (0.01 sec)
```

書き方はほとんど自明だろう。 検索の条件として、name='tanaka ichiro'であるようなデータ(レコード)を探している。 等号の右辺はデータなので、1重引用符「'」で括ってある。 条件にマッチする結果(この場合1行だけ)が表示されている。

条件を AND や OR で組み合わせることも出来る。 試してみよう。

```
mysql> select * from mytest where name='tanaka ichiro' or age='20';
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950|   19 |
| yamada taro   | 5511951|   20 |
+---------------+---------+------+
2 rows in set (0.00 sec)
```


条件が OR なので、どちらかの条件を満たす行が選ばれている。
また、値に対する条件を不等号で書くこともできる。年齢 age が 20 より大きいレコード(行)だけを表示するには

```
mysql> select * from mytest where age>='20';
+-------------+---------+------+
| name        | id      | age  |
+-------------+---------+------+
| yamada taro | 5511951|   20 |
+-------------+---------+------+
1 row in set (0.00 sec)
```

次は、検索結果の表示をソートすることを試す。表示をソートするには、ORDER BY 構文を使う。name をキーにしてソートしたい場合、

```
mysql> select * from mytest order by name;
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950|   19 |
| toho hanako   | 5511952|   19 |
| yamada taro   | 5511951|   20 |
+---------------+---------+------+
3 rows in set (0.00 sec)
```

この場合、ORDER BY が指定していないと、どういう順番で表示されるか分からない。データベースの中では、データは「行の集合」とみなされ、行の順番は意味が無いのである。 それを順番付けて表示するのが、ORDER BY である。

名前 name は文字列なので、文字列としてのソートをしている。 文字の順序はあまり意味が無いかもしれないが、一応アルファベット順になる。 2行目と3行目の順番

がひっくり返ったのがわかると思う。(注: 漢字データのソートは厄介な問題で、ここでは触れないでおく)

最後に、一部のフィールドだけを選んで表示する例を示す。
ここの例で名前 name と年齢 age のフィールドだけを表示したい場合は、SELECT のあとに対象フィールドの名前を name,age のように列挙すればよい。

```
mysql> select name, age from mytest;
+---------------+------+
| name          | age  |
+---------------+------+
| tanaka ichiro |   19 |
| yamada taro   |   20 |
| toho hanako   |   19 |
+---------------+------+
3 rows in set (0.00 sec)
```

データの更新

既に存在する行(レコード)の、特定のフィールドの値を書き換える処理である。
新しいデータが入手できた時や、以前レコードを作った時に空白のままにしておいたフィールドに値を書き込むなどの用途があるだろう。

書込みは UPDATE 文を使う。UPDATE 文は、指定したフィールドを指定した値に書き換える。

UPDATE 表の名前 SET フィールドの名前=新しい値

のようにする。

更新する行(レコード)を指定するのは、SELECT と同じ WHERE 節を使う。
たとえば、tanaka ichiro の年齢を 18 に書き換えるためには、WHERE name='tanaka ichiro'として対象となるレコードを選択する。具体的には、


```
mysql> update mytest set age='18' where name='tanaka ichiro';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from mytest;
+---------------+--------+------+
| name          | id     | age  |
+---------------+--------+------+
| tanaka ichiro | 5511950|   18 |
| yamada taro   | 5511951|   20 |
| toho hanako   | 5511952|   19 |
+---------------+--------+------+
3 rows in set (0.00 sec)
```

でできる。最後にSELECT文で表を書き出しているが、確かに値が書き換わっていることが分かる。

UPDATE は、WHERE 節の条件が合致すれば 複数の行でも同時に書き換えられる。
上記の状態から、age が 20 未満の行(レコード)のすべてについて、ID を無理やり'8888***'に書き換えてみる。

```
mysql> update mytest set id='8888***' where age<'20';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 2

mysql> select * from mytest;
+---------------+--------+------+
| name          | id     | age  |
+---------------+--------+------+
| tanaka ichiro | 8888***|   18 |
| yamada taro   | 5511951|   20 |
| toho hanako   | 8888***|   19 |
+---------------+--------+------+
3 rows in set (0.01 sec)
```

この例では age が 20 未満のレコードが2つあり、そのことが「2 rows affected」「Changed: 2」というコメントにも現れている。 念のため SELECT 文で表全体を書き出してみているが、1行目と3行目の id 部分が書き換わっていることがわかる。

修了確認課題　MySQL を使ってみよう

7章のまとめとして、下記の問題を試して見よ。 条件をいろいろと変えてあるので、それに併せてコマンドなどを書き換えよ。

東邦大学理学部の学科、学生数、就職希望者数、内定者数、未内定者数の5つを保持する表 shushoku を作れ。 具体的には CREATE TABLE で表を作り、そこにデータを挿入せよ。 表の名、列の名は（漢字を使わず）英数字を使うこと。

学科	学生数	就職希望者数	内定者数	未内定者数
chem	107	62	62	0
bio	90	54	52	2
biomol	116	63	62	1
phy	90	48	47	1
info	101	75	73	2

次に、この表のデータを検索せよ。

（1）biomol 学科の就職状況（レコード＝行全体）を表示せよ。

（2）学生数が 100 人未満の学科の就職状況を表示せよ。

（3）未内定者が皆無の学科の名前（学科名だけ）を表示せよ。

（4）info 学科の学生数が間違っていたので、102 に変更せよ。

（5）WHERE 節に式を書いてしまおう。（就職希望者数／学生数）が 0.65 より大きい学科を選んで、その就職状況を表示せよ。

（6）未内定者数の多い順（多いほうが上）にソートして、表全体を表示せよ。（同数の場合の順序にはこだわらないことにする） 降順にソートするには

```
select * from mytest order by name desc;
```

のように DESC (descending order)を付ける。

この課題が正しくできたことを、TA に確認してもらって下さい。 これによって、第7章を修了したこととします。

修了したら、次の章へ進んでください。

第８章　　PHP から MySQL を呼び出す

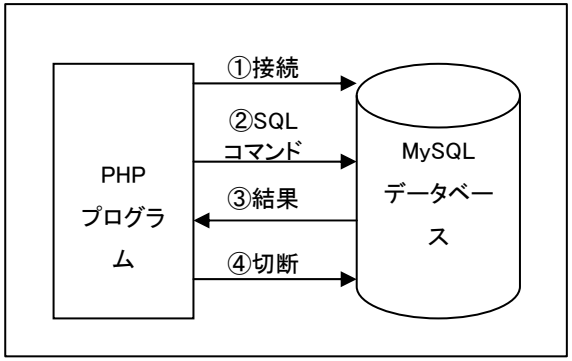
PHP から MySQL を呼び出す枠組み

第7章で、SQL のコマンド（正しくは「キュエリー（＝問合せ）」と呼ぶ）の使い方を、端末から手で打ち込むことによって試した。
ここでは最後の問題として、手で打ち込むのではなく、PHP のプログラムから SQL キュエリーを生成してデータベースにアクセスし、その結果を受け取る方法を学ぶ。 ここまでできれば、ネットから CGI を経由して入力を受け取り、データベースを読んだり書いたりするプログラムが作れる。

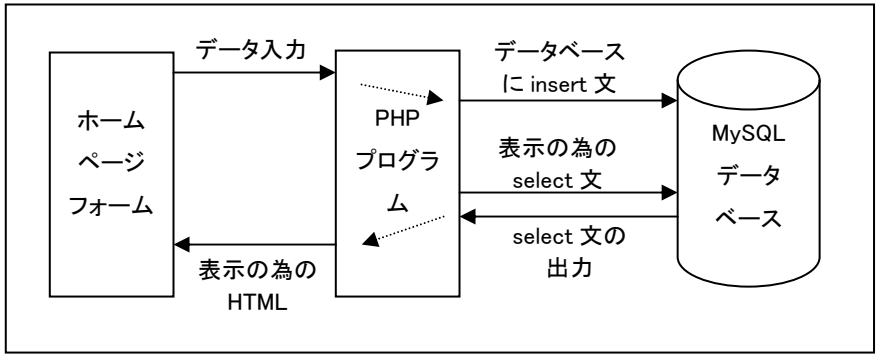
PHP プログラムと MySQL データベースの間のインターフェースは、

① PHP プログラムと SQL データベースを接続する、
② PHP プログラムから SQL データベースへ向かって SQL 文を放る（SQL コマンドを投げる）、
③ SQL データベースから PHP プログラムへ検索結果を送り返す（PHP から見ると結果を拾い上げる）、
④ PHP プログラムと SQL の間の接続を切断する、

という4つからなる。
第8章では、この4つの手順を含む PHP のプログラムを順番に作り上げ、実行してみる（右図）。



この［PHP←→データベース］の仕組を、第4～5章で見た［ホームページ←→PHP］のやり取りと組み合わせると、次の図のようなことができる。



左側に示した［フォームのページ］で入力を入れて「送信ボタン」をクリックすると、中央に示した PHP プログラムが起動される。起動されるプログラムのファイル名は、HTML の form 文の action パラメタに書いたものである（ここまで第4～5章の内容）。起動された PHP プログラムの中で、前の図にあるように①データベースに接続し、次にフォームから受け取ったデータを②SQL の insert コマンドでデータベースに挿入する。Insert コマンド自体は結果は表示しないので③は無視してよい。
次に、データベースの内容がどう変わったかを表示する。接続は（切断していなければ、同じ PHP プログラム内であれば）継続しているので、もう一度②を繰り返すことができる。 ②SQL の select コマンド（データベース読出し）を送り、今度は③結果を受取る。この受取った（読出し）結果を、表示のための HTML に合うように書き変え（書き足し）てやって、左側の端末へプリントする。 後始末として④切断をしておく。

このような手順を踏めば、端末からの入力データをデータベースに書き込んで、その結果（書きこんだ後のデータベース中のデータ）を端末に表示することができる。

第8章では、PHP プログラムと MySQL データベースとのやり取りの仕方を学び、それを応用して最後にブログに相当する仕組を作ってみる。 ブログは、端末からメッセージを投稿してデータベースに蓄え、それを投稿順に表示するシステムだと考えればよい。

SQL データベースを接続する　　mysql_connect と mysql_select_db

PHP のプログラムが mySQL データベースを使うために、初めにデータベースに接続するという作業がある。 これには、**mysql_connect** 関数を使う。
引数は、mySQL データベースのサーバーが置かれているホスト（普通は自ホストだが、リモートホストに置いた mySQL サー バーをアクセスするような設定も出来る）、mySQL 上のユーザ名、パスワードを与える。たとえば

```
$MySQLHost = 'localhost';      これは誰でも同じで localhost
$MySQLUserName = '5511999';    これは自分の mySQL の ＩＤ
$MySQLPassword = 'mypassword';   これは自分の mySQL のパスワード
$MyLink = mysql_connect($MySQLHost, $MySQLUserName,
                                    $MySQLPassword);
```

のようにすることができる。但し、
「5511999」 は自分の MySQL 用の ID、「mypassword」はそのパスワードである。

このプログラム例で分かるように、mySQL のユーザ ID とパスワードを PHP のプログラム中に（明示的に・平文で）書いておかなければならない。もし PHP プログラムのソースコードが見られてしまうと、セキュリティ上具合が悪いので、普通はここでちょっとしたひとひねりを加える。上の3行（ユーザ ID などを変数に代入するところ）は別ファイル 「 .ht_mytest 」（ドットから始まるファイル名である。ここでは拡張子無しにしてある） に置き、それを PHP プログラムで取り込む（インクルードする）ようにする。つまり、form 文の action パラメタで指定して起動する PHP プログラムファイルには書かないでおく。
Include 文は、その場所に、指定したファイルの中身を取り込むものである。

更に Web サーバー（apache）の設定で、「.ht」から始まるファイルは直接外へ見せないという設定にしておくと （実習室では管理者が既にそのように設定してある）、パスワードの値を書いたファイル「 .ht_mytest」 は直接アクセスしようとしても外へ見えない、というようにできる。 PHP のメインのプログラムでは include 文を使ってファイル「.ht_mytest」 の内容を取り込む。


```
include(".ht_mytest");
    これで変数 $MySQLHost,  $MySQLUserName, $MySQLPassword が設定される
$MyLink = mysql_connect($MySQLHost, $MySQLUserName,
                        $MySQLPassword);
```

このトリックはややこしいので、⑭切断は元のプログラムに埋め込んでしまって、後からゆっくり考えるのでもよかろう。 その場合、セキュリティ上のリスクがあることは認識して欲しい。
（注） ファイル「 .ht_mytest」 は独立した PHP プログラムファイルなので、最初に <?php 、最後に ?> が必要である。

```
<?php
$MySQLHost = 'localhost';
$MySQLUserName = '5511999';
$MySQLPassword = 'mypassword';
?>
```

接続が出来たら、次にデータベースを選択する。
第7章で端末から対話的に SQL コマンドをタイプインして動かした時に、「**use** コマンド」を使ってデータベースを選択したが、その機能をプログラムから実行するための処理である。 これをやらないと、その後の SQL コマンドを実行するときに、エラーになる。
データベースはの選択は、関数 **mysql_select_db** を呼び出すことで行う。 第1引数は選択するデータベースの名前を入れ、第2引数は上の mysql_connect 呼出しの戻り値 （ここでは$MyLink） を入れる。 この戻り値 $MyLink は後で SQL 呼出しをするときに、常に必要になる（引数で指定する）。

```
$MySQLDatabase = 'd5511999';
mysql_select_db($MySQLDatabase, $MyLink);
```

これら **mysql_connect** や **mysql_select_db** 関数についての詳細は、PHP のマニュアルの MY SQL サーバー関数の章 http://www.php.net/manual/ja/ref.mysql.php を参照して欲しい。

PHP から SQL へコマンド(問い合わせ)を投げる　mysql_query

上記の、接続 （**mysql_connect**） と、データベース選択 （**mysql_select_db**） が正常に済むと、SQLのコマンド(問い合わせ、クエリ)を投げてデータベースにアクセスすることが出来る。

PHP プログラムから SQL データベースをアクセスするには、PHP のプログラムの中で、SQL のコマンド(クエリ＝問合せ)を 「その通りの形」 で生成して、mySQL に投げる。 つまり、前章で試したmySQLを端末から対話モードで使う時にタイプインしたのと、**まったく同じ SQL 文を文字列** として用意し （最後のセミコロン「;」は除く）、それを関数 **mysql_query** の引数に与えて呼び出す。

（注: 対話モード時の行末のセミコロン「;」は、対話モード時に入力の行の最後を示すためのもの）

たとえば、次の例では文字列を格納する変数 $Query に、SQL の問合せ文

SELECT * FROM mytest WHERE age=' 19'

を(文字列として)代入しておき、(ここまでが1行目)、 次に関数 **mysql_query** を mysql_query ($Query, $MyLink) として呼び出すことで、mySQL データベースにこの SQL 文を投げる。

```
$Query = "SELECT * FROM mytest WHERE age='19'";
$Result = mysql_query($Query, $MyLink);
                    $Query を送出し、結果を$Result に受取る
```

これによって、mySQL データベースに SELECT 文が送られる。 もしどんな SQL 文が送られているかを見たければ、print 文で変数 **$Query** の内容を表示するとよい。

```
$Query = "SELECT * FROM mytest WHERE age='19'";
print("<p>Query: " . $Query . "</p>¥n");  //  $Query を表示
$Result = mysql_query($Query, $MyLink);
```

関数 **mysql_query** についても、マニュアルの MY SQL サーバー関数の章

http://www.php.net/manual/ja/ref.mssql.php

を見て欲しい。

SQL から検索結果を受け取る　mysql_fetch_array など

SQL コマンド(問い合わせ)には、

＊データベース側に作用を及ぼすが戻り値は利用しない操作
（たとえば新しい表を作るとか、データの更新をするとか)と、

＊結果を戻す操作(たとえば検索をして結果を返すような操作)

がある。

戻り値を利用しない操作の場合は、mysql_query 文で操作のための SQL 文を投げたままでよく、特に結果を取り出す必要はない。(つまり戻り値$Result を使う必要がない。) もちろん、操作がうまく完了したかどうかをチェックするべきである(引数に与えた値がおかしいために操作がうまく行かないということがありえる)が、今はサボっておく。

他方、結果を使いたい場合、たとえば検索結果を利用したい場合は、mysql_query の戻り値(検索結果)を一旦変数(下の例では変数$RESULT)に代入し、それを後からうまく読むことにする。 検索の結果(戻り値)は、普通は、単一の値(3とか文字列 "ABC"とか)ではなくて、下のような表の形になっている。 (前の節で試したように、検索結果は表になっているから)

```
+----------------+---------+------+
| name           | id      | age  |
+----------------+---------+------+
| tanaka ichiro  | 5505950 |   18 |
| yamada taro    | 5505951 |   20 |
| toho hanako    | 5505952 |   19 |
| yoshida takuro | 5505953 |   19 |
+----------------+---------+------+
```

このような結果(＝戻り値)は、何行あるかが検索結果によって変わる(条件にヒットした数による)ため、取り出す関数に工夫がされている。次の例を見て欲しい。

```
$Query = "SELECT * FROM mytest WHERE age='19'";
$Result = mysql_query($Query, $MyLink);

$count = 0;
```

```
while ($Row = mysql_fetch_array($Result, MYSQL_ASSOC)) {
    $count = $count+1;
    print("<p>$count  " . $Row["id"] . " " .
      $Row["name"] . " " . $Row["age"] . "</p>¥n");
}
```

この例では、SQL 文による問合せ **mysql_query** の結果(＝複数の行を含むかもしれない) を始めにいったん1つの変数 **$Result** に格納しておくが、その **$Result** に含まれている(表の)1行1行を取り出すために、**mysql_fetch_array** という関数を、while 文を使って繰り返し呼び出している。

もしこの関数が「空」(プログラム上は while の条件が false になる)を戻せば while ループを脱出するというプログラムになっている。 よく見かける、ファイルからの入力が空になるまで while で繰り返して呼び出す、というプログラムと同じ考え方の構造である。それによって、(複数行ある)SQL 検索結果**$Result** を、 **mysql_fetch_array** 関数を用いて、1回につき1行ずつ読み出している。

読み出した1行分の結果は、**$Row** という変数に代入しているが、

```
$Row = mysql_fetch_array($Result, MYSQL_ASSOC)
```

$Row 自身は更に配列になっており、1つの行内の各フィールド(たとえば id とか name とか age とか)の値を持っている。つまり、検索結果のとある1行 の、**$Row["id"]**は学生番号のフィールドの値を、**$Row["name"]**は名前のフィールドの値を、**$Row["age"]**は年齢のフィールド の値を、保持している。 ここでの"id"とか"name"とか"age"とか(配列変数**$Row** の列の名前)は、データベース上の表(テーブル)を **create table** で作った時の列(欄)の名前であり、忘れてしまっていれば、SQL のコマンド(問合せ)の **describe** 表の名前; とすれば確認できる。

```
mysql> describe mytest;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| name  | char(20) | YES  |     | NULL    |       |
| id    | char(7)  | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

サンプルプログラムでは、print 文でこれらのフィールドの値を画面に表示している。

```
print("<p>$count  " . $Row["id"] . " " .
      $Row["name"] . " " . $Row["age"] . "</p>¥n");
```

print 文では書きたい(画面に表示したい)文字列を指定する。前にもふれたが、print されるのは Web ブラウザの中なので、HTML で書かれたデータであると解釈されて表示される。 またここでは、

文字列 . 文字列 . 文字列

といった形をしているが、間にあるピリオドは2つの文字列同士を繋ぐ操作であることを思い出して欲しい。 更には、** ** は HTML の表記で空白を表す定数である。 文字としての半角空白「 」を複数書いても、HTML では空白1つにしか解釈されない。

例として、データベースは下記のように1行(1レコード)分を増やしてあるとする。

```
mysql> select * from mytest;
+----------------+---------+------+
| name           | id      | age  |
+----------------+---------+------+
| tanaka ichiro  | 5505950 |   18 |
| yamada taro    | 5505951 |   20 |
| toho hanako    | 5505952 |   19 |
| yoshida takuro | 5505953 |   19 |
+----------------+---------+------+
4 rows in set (0.00 sec)
```

実行結果は2行だけ表示される。 なぜなら、条件 age='19'に合致する行は2つしか無く、それが表示されるからである。 もし、WHERE 節を取り除いて、すべての行を表示するようにすれば、下図のように4行表示される。

```
$Query = "SELECT * FROM mytest";
```

```
1  00r950 tanaka ichiro 18
2  00r951 yamada taro 20
3  00r952 toho hanako 19
4  00r953 yoshida takuro 19
```

全体のイメージは

```
include(".ht_mytest");     変数$MySQLHost, UserName, Password が設定される
$MyLink = mysql_connect($MySQLHost, $MySQLUserName, $MySQLPassword);
$MySQLDatabase = 'd5511999';
mysql_select_db($MySQLDatabase, $MyLink);
$Query = "SELECT * FROM mytest WHERE age='19'";
$Result = mysql_query($Query, $MyLink);

$count = 0;
while ($Row = mysql_fetch_array($Result, MYSQL_ASSOC)) {
    $count = $count+1;
    print("<p>$count  " . $Row["id"] . " " .
        $Row["name"] . " " . $Row["age"] . "</p>¥n");
}
```

のような感じになる。

演習課題　　PHP から MySQL をアクセスしてみよう

第7章の最後の課題で作成したデータ(表 shushoku)に対して、PHP からのアクセスを試してみる。

7章の課題で使った後の状態は info の学生数が 102 になっているので、元の 101 に戻しておくこと。

(1) この表のデータに対して、次の検索を行う PHP プログラムを作ってみよう。

(1)biomol 学科の就職状況(レコード＝行全体)を表示せよ。

(2)学生数が 100 人未満の学科の就職状況を表示せよ。

(3)未内定者が皆無の学科の名前(学科名だけ)を表示せよ。

(4)値(就職希望者数／学生数)が 0.65 より大きい学科を選んで、その就職状況を表示せよ。

(2) 他のコマンド(問合せ、キュエリー)も同様に、PHP プログラムにして試してみよう。 要するに、問い合わせの文字列 **$QUERY** の中身を、自分のやりたい問合せに書き直し、かつ、結果の表示を適当なものに置き換えればよい。

(1) データベースに入力するプログラムを作ってみよう。 第7章で使ったテーブル mytest

```
+---------------+---------+------+
| name          | id      | age  |
+---------------+---------+------+
| tanaka ichiro | 5511950 |   19 |
| toho hanako   | 5511952 |   19 |
| yamada taro   | 5511951 |   20 |
+---------------+---------+------+
```

に対して、1行分のデータを追加する PHP プログラムを作れ。

```
$xname = 'daisuke tanaka';
$xid = '5505961';          <<< この欄は文字列であることに注意
$xage = 22;                <<< この欄は文字列では無く int 型なことに注意
$Query = "insert into mytest values ('" . $xname . "', '" . $xid . "', '" .
    $xage . "')"
```

正しく挿入できたかどうか、データベースを直接見てみよ。

（3） 第5章で試した、端末のブラウザから入力してプログラムに渡す仕組と、ここでのデータベースとを組合せてみよう。
（1）と同じように、表 mytest に1行追加するが、その値の組はブラウザから入力できるようにしよう。
まず form を含むページを作る。

```
<HTML><BODY>
<FORM action="myaction.php" method="post">
  <P>名前<INPUT size="20" type="text" name="name"></P>
  <P>学生番号<INPUT size="20" type="text" name="id"></P>
  <P>年齢<INPUT size="20" type="text" name="age"></P>
  <P><INPUT type="submit" name="send" value="送信"></P>
</FORM>
</BODY></HTML>
```

このフォームの「送信」ボタンにより起動されるプログラム myaction.php を作る。

```
（前は省略）
$xname = $_POST["name"];
$xid   = $_POST["id"];
$xage  = (int) $_POST["age"];
$Query = "insert into mytest values ('" . $xname . "', '" . $xid .
    "', '" . $xage . "')"
$Result = mysql_query($Query, $MyLink);
（後ろも省略。 表全体を表示するような部分を追加せよ。 表示は HTML の形に整えてから print する必要がある。809 ページの例を参照）
```

これを実行すると、起動するたびにデータベースに1行ずつ追加するようになる。

動作確認

上記の ≪演習問題 PHP から MySQL をアクセスしてみよう≫ の課題が正しくできたことを、TA に確認してもらって下さい。 続いて次の、前半最後の課題に進んでください。

前半最後の演習課題 簡単な掲示板を作ってみよう

5章の課題5－4で作ったフォーム入力を参考にして、この章で学んだデータベースアクセスを使って、簡単な掲示板を作ってみよう。 これはプロジェクトの後半（ホームページサービスを作ってみよう）に直結した演習なので、なるべく全員が挑戦してほしい。

第3回目の授業時間内に終わらない場合は、4回目以降（後半）のグループ制作の中で完成させるとよい。 最初はなるべく凝らずに、簡単な最低限のシステムから始めるのがよいだろう。 凝るのは第4回目以降に作るシステムでするほうがよい。

動作原理は、入力フォーム画面からメッセージと投稿者名を取り込み、その組をデータベースの1行に書き込む（SQL の INSERT を使えば出来るはずだ）。 そのためには予めメッセージと著者名を入れられる表を作っておかなければならない（CREATE TABLE で作れるはずだ）。 あとは、投稿結果を表示する画面を作ればよかろう。表示はデータベースの内容を読み出して、データベースの1行ずつの内容を HTML の形に整えて、print してゆけばよかろう。以下にもう少し細かく考えて見る。

まず書込みは、最初に HTML ファイルで課題5－4のようなフォーム入力の投稿画面を用意し、表示させる。投稿画面で「送信」をクリックすると、 action で指定された PHP ファイルが起動され、その中で$_POST[...]を使って、入力情報を取り出す。ここまで課題5－4と同じである。
次に、得られた投稿タイトル、投稿者氏名、投稿本文と、更にシステムの時間を取り出して投稿時間とし、これらの情報をデータベースの1行（レコード）として書き込む。データベースには、あらかじめ （この掲示板システムをインストールした時点で） これらの項目をフィールドとして持つような表を作っておくものとする。 書込みの action として呼び出された PHP プログラムは、この表に1行（レコード）分のデータを書き込む。次々に書き込んでいくと、次々に行が増えてゆく。

データベースの表の列（フィールド）は、始めはあまり凝らないことにする。**タイトル**はたとえば最大40文字ぐらいの文字列としてしまおう。**名前**は最大20字ぐらいの文字列でよかろう。**本文**は TEXT 型というのを使える。文字列型（CHAR や VARCHAR）では255文字までしか格納できないので、もう少し長くてもよい TEXT 型（65535 文字まで格納できる）にする。TEXT 型のバリエーションもいろいろあるが、MySQL マニュアルの「各フィールド型の所要容量」の節を参照して欲しい。

投稿時刻は、SQL にもいろいろな型があるが、単純な UNIX システム時刻（1970 年 1 月 1 日からの秒数）を格納することにしよう。これだと整数として比較・ソートして、時間の順に表示することが出来る。しかし、そのまま表示すると意味がわからないので、表示する時に UNIX 時刻を普通の日時の表記に変換する関数 date を呼び出すことにする。5章に示した手順

```
$now = gettimeofday(); $time = $now[sec]; $datestring = date("r", $time);
```

は、関数 gettimeofday()によって現在の UNIX 時刻を取り出して配列$now へ代入し、$now の秒単位の表記要素である$now [sec]を使って（UNIX 時刻の秒表記を）変数 $time に格納し、最後に date 関数によって時刻$time を通常の文字表記に変換している。変換の仕方は"r"つまり普通に使われている日時の表記を指定している。今回はユーザが時刻を指定する機能はサボることにするので、逆向きに文字表記の日時から UNIX 秒時刻への変換は使わずに済むだろう。

これらの方法で準備したフィールドの情報を使って、SQL の INSERT INTO 文を作り、mySQL へ送りつけて格納する。 行の挿入の操作では、mySQL からの戻り値は気にしなくてよいだろう。（失敗することもないではないが。）

次に読出しであるが、いろいろと凝ることが出来るが、今回は簡単に、データベース上の投稿者が入力された名前と一致するデータ（行＝レコード）を抽出し、それを投稿時刻の順に並べて表示する、というだけにしてみよう。

これをするためには、検索に使う投稿者名を入力するための フォーム画面を表示する HTML ファイルと、そのフォームの action で起動される PHP プログラムが必要となる。 PHP プログラムの中では、「データベース内の名前のフィールドのデータが、

フォームから$_POST[...]で得られた名前と一致するレコードを取り出せ」という SQL 文を作って、データベースに投げる。 一致するレコードは複数ある可能性があるので、この章の前半で見たような while 文を使った方法で、複数の検索結果レコードを1つずつ読み出して、print 文で表示させる。

検索結果を日時の順に並べて（つまりソートして）古いものから表示するには、データベース上 の投稿日時のフィールドが UNIX 時刻（＝1970 年を起点とする秒数）で書かれているので、これを整数とみなして大きさの順にソートすれば、投稿日付の順に並べられる。 SQL では ORDER BY という構文を使うと、指定したキーの昇順（もしくは降順）にソートして結果を返してくれるので、それを while で1つずつ読み出してゆくとソートされた順に読み出せる。 なお、ソートの向きは何も指定しないと昇順なので、降順にするためには ORDER BY "date" DESC のように DESC を指定すればよい。

ここまでで、どのような画面が必要で、どのような PHP プログラムが必要になるのか、よく整理した上で、全体の関係を絵に書いてみよ。 フォームの action= で指定されるファイルを、矢印で結んでおくとよい。
ややこしい点として、INSERT をする流れと、データベースの内容を表示する流れ（SELECT をして結果を表示する流れ）と、2つの流れがあることである。INSERT をした後にその結果が表示されて欲しいが、一方で INSERT をしないままで表示だけをする流れも欲しい。どうしたらできるか、いろいろと考えて工夫してみるとよいだろう。

動作確認には、インタラクティブ端末を使ってデータベースを読み出したり書き込んだりしながら、変化を追跡して確認するとよい。

修了確認

この ≪前半最後の演習課題≫ が正しく完成できたかどうかを、TAに確認してもらってください。 これによって、第8章、ならびにプロジェクトの前半部分を修了したこととします。