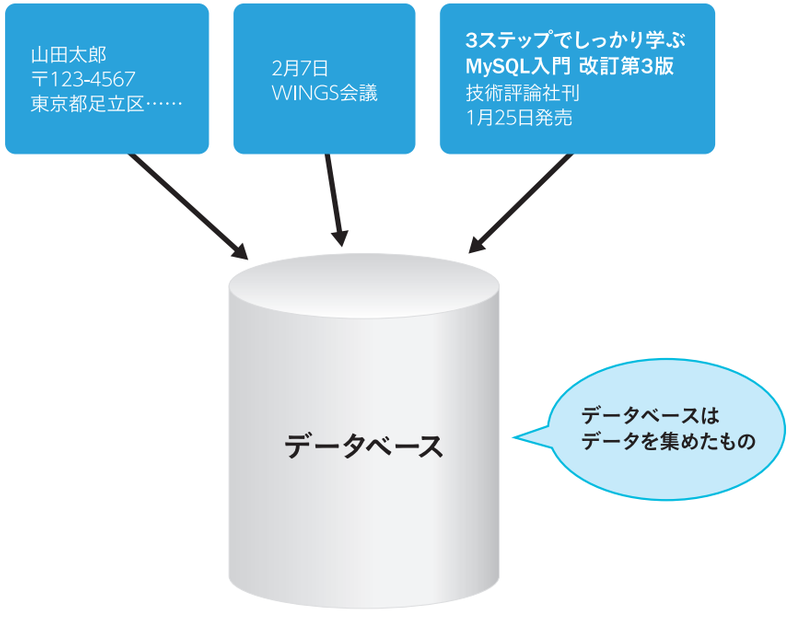
1 データベースとは

予習 データベースについて理解する

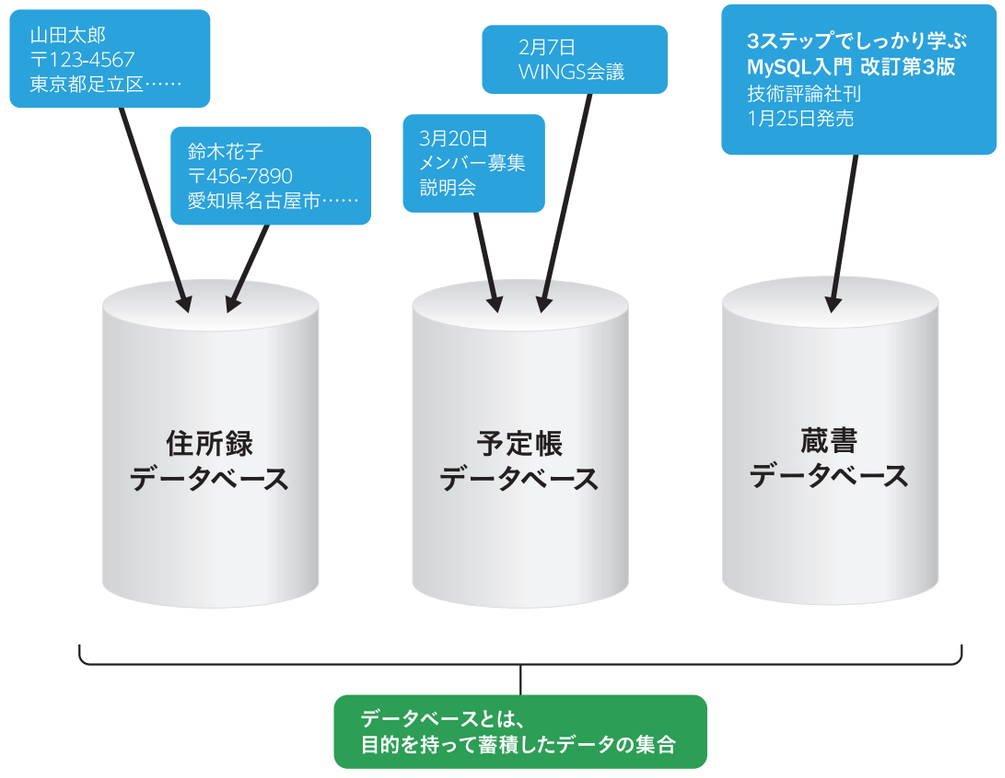
本書のテーマである**MySQL**は、もっとも人気のあるデータベース製品の１つです。
データベースとは「コンピュータに蓄積されたデータの集合」のことです。ここでは、データベースの定義や種類について、もう少し詳しく見ていくことにしましょう。



理解 データベースについて

ステートメントについて

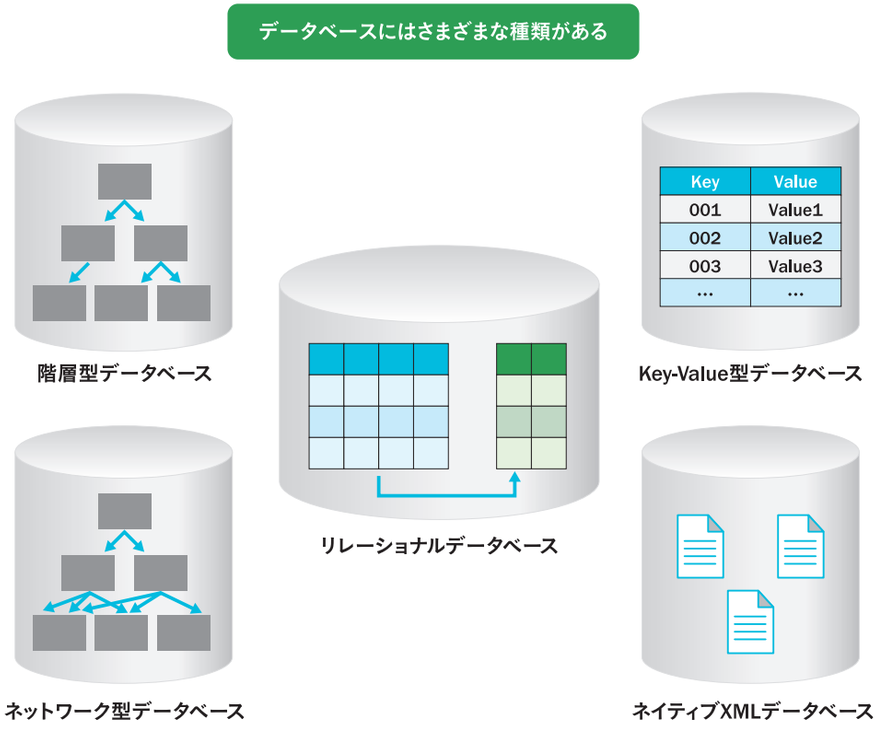
データベース（DataBase）とは、「コンピュータに蓄積されたデータの集合」のことです。ただし、ただ無作為にデータが集められているだけでは意味がありません。たとえば、人の名前や住所、買った本の値段、今日の予定など互いに関係のないデータをいくら集めてもデータベースにはなりません。しかし、人の名前と住所だけをまとめれば「住所録」データベースになりますし、予定情報に絞って集めれば「予定帳」データベースに、本の名前や値段、感想などを集めれば「蔵書」データベースになるでしょう。
つまり、データベースとは、「コンピュータに**＜目的を持って＞蓄積したデータの集合**」と言えます。



2 データベースの種類

予習 データベースのさまざまな種類を知る

ここまででデータベースがどのようなものなのか、だんだんイメージできてきたでしょうか。実は、一口にデータベースと言っても、その形態はさまざまです。データをどのように格納するかによって、データベースにはさまざまな種類があるのです。ここでは、具体的にどのようなデータベースがあるのかを見てみましょう。

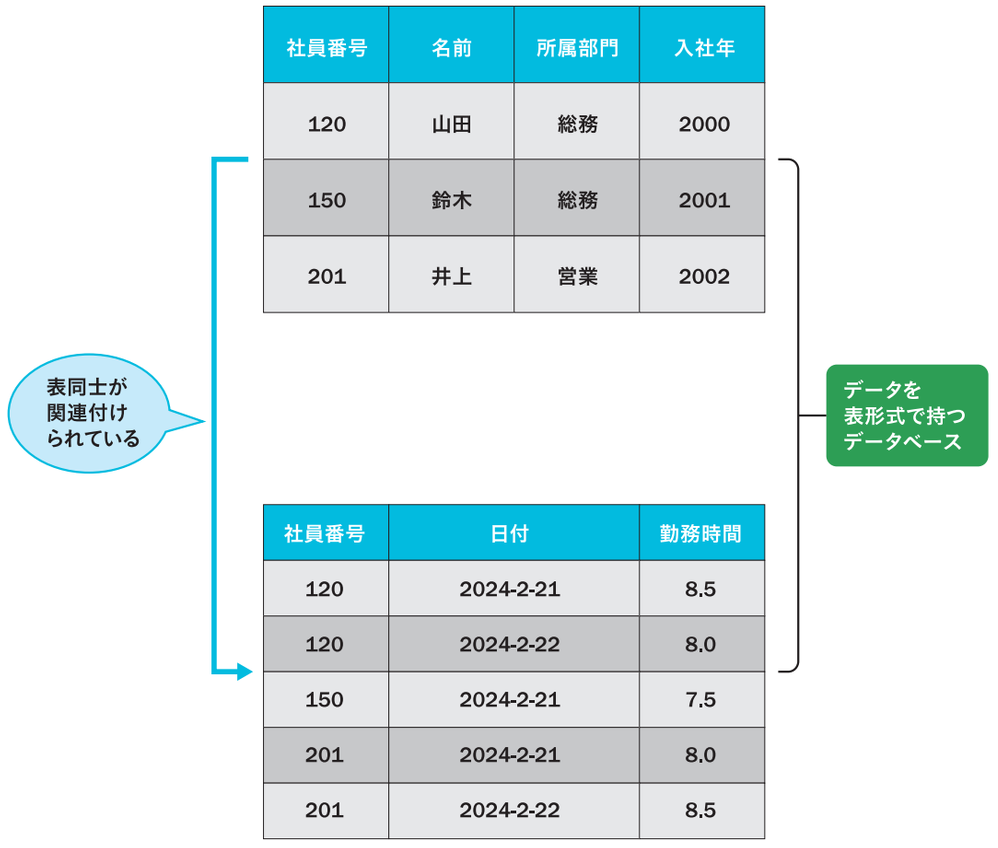


理解 さまざまなデータベース

リレーショナルデータベース

データをExcelのような表形式で持つデータベースです。表同士が関連付けられているのも特徴です。現在、もっとも主流とされているデータベースで、無条件にデータベースといった場合には、リレーショナルデータベース（RDB：Relational DataBase）を指すと考えてよいでしょう。

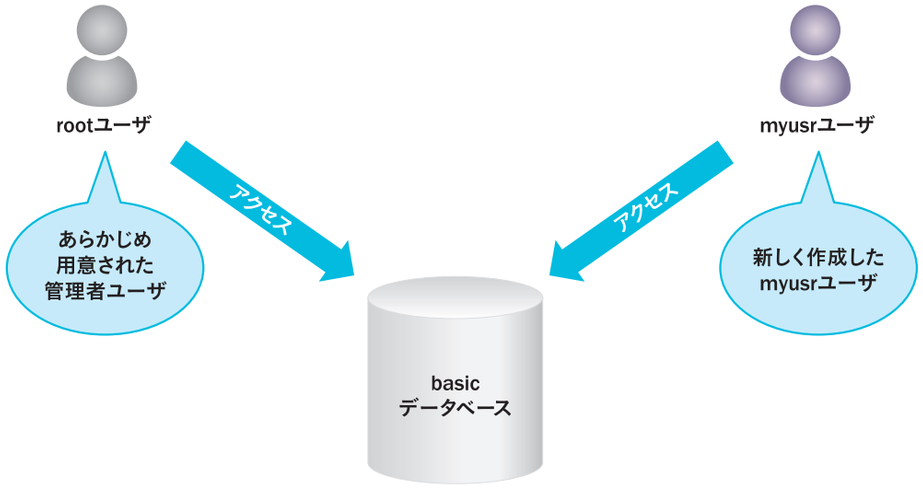
本書で扱うMySQLも、リレーショナルデータベースに分類されます。リレーショナルデータベースについては、次の節で詳しく解説します。



4 ユーザの作成

予習 ユーザの概念と作成方法を理解する

ここまでは、あらかじめ用意された管理者ユーザ（root）を利用してきましたが、セキュリティなどの事情を考えれば、なんでもできてしまうrootユーザを日常的に利用することは好ましくありません。そこでここでは、自分で作成したbasicデータベースを操作するためのmyusrユーザを作成しておきましょう。
ユーザを作成する手順を通じて、ここでは、ユーザや権限という概念についても理解します。



体験 ユーザを作成しよう

1 mysqlクライアントを起動する

2-2の手順に従って、mysqlクライアントを起動します。

```
PS C:\Users\nami-> mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

❶ コマンドとパスワードを入力してそれぞれ Enter キーを押す

```
PS C:\Users\nami-> mysql -u root -p↵
Enter password: *****
```

2 新規のユーザを作成する

新規のユーザを作成します。右のように入力して、CREATE USER命令を実行します❶。

Tips
ここでは、パスワードが「12345」であるmyusrユーザを新規に作成しています。

```
mysql> CREATE USER myusr@localhost IDENTIFIED BY '12345';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

❶ 入力して Enter キーを押す

```
mysql> CREATE USER myusr@localhost IDENTIFIED BY '12345';
```

3 myusrユーザにすべての権限を与える

手順2で作成したmyusrユーザに、basicデータベースに対するすべての権限を付与します。右のように入力して、GRANT命令を実行します❶。

```
mysql> GRANT ALL ON basic.* TO myusr@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

❶ 入力して Enter キーを押す

```
mysql> GRANT ALL ON basic.* TO myusr@localhost;
```

4 mysqlクライアントを終了する

新規に作成したユーザでログインし直すため、いったんmysqlクライアントを終了します。右のように入力して、exit命令を実行します❶。図のように元のプロンプトに戻ります。

```
mysql> exit;
Bye
PS C:\Users\nami->
```

元のプロンプトに戻る

```
mysql> exit;
```

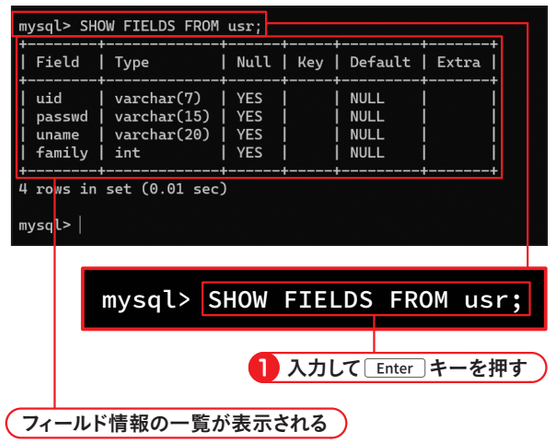
❶ 入力して Enter キーを押す

4 テーブルを確認する

usrテーブルが作成されたことを確認してみます。右のように入力してSHOW TABLES命令を実行します❶。すると、データベースに含まれるテーブルの一覧が表示されるので、「usr」が含まれていれば成功です。

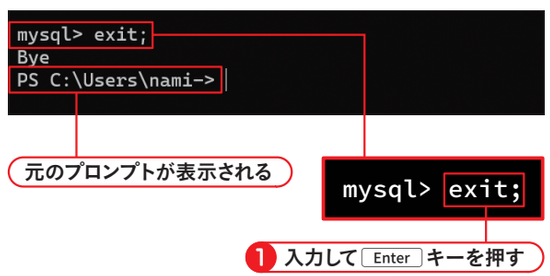
5 フィールド情報を確認する

usrテーブルのフィールド情報を確認してみます。右のように入力してSHOW FIELDS命令を実行します❶。すると、「usr」テーブルに含まれるフィールドの情報の一覧が表示されます。



6 mysqlクライアントを終了する

mysqlクライアントを終了します。右のように入力して、exit命令を実行します❶。図のように元のプロンプトに戻ります。



理解 テーブルの作成方法

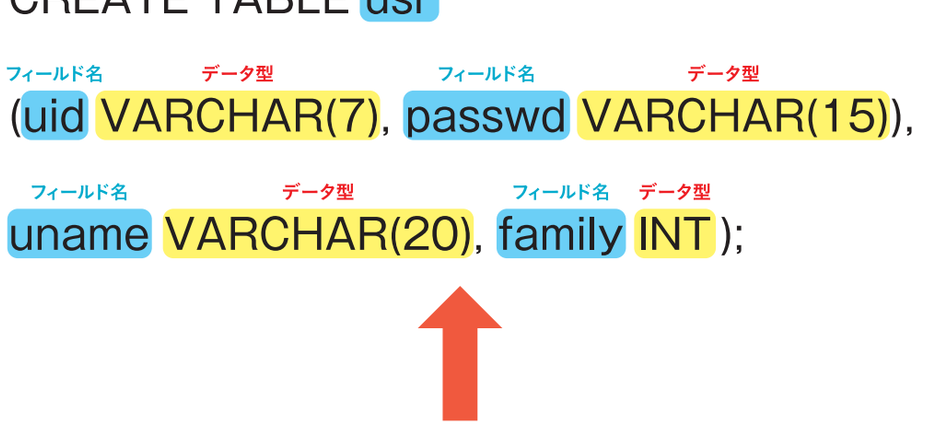
テーブルを作成しよう

新たにテーブルを作成するには、**CREATE TABLE**という命令を使います。CREATE TABLE命令の基本的な構文は、次のとおりです。

▼構文

```
CREATE TABLE テーブル名 (フィールド名1 データ型1 属性1,
  フィールド名2 データ型2 属性2, ...)
```

やや長めなので難しく見えるかもしれませんが、カッコの中はフィールドの定義を繰り返しているだけです。命令は、途中で改行しても構いません。**2-3**で解説したとおり命令の終了はセミコロン「;」で表します。下図のように命令と表を比較してみれば、難しいことはありませんね。なお、属性については、ここでは必要ないので設定していません。**4-3**で紹介します。



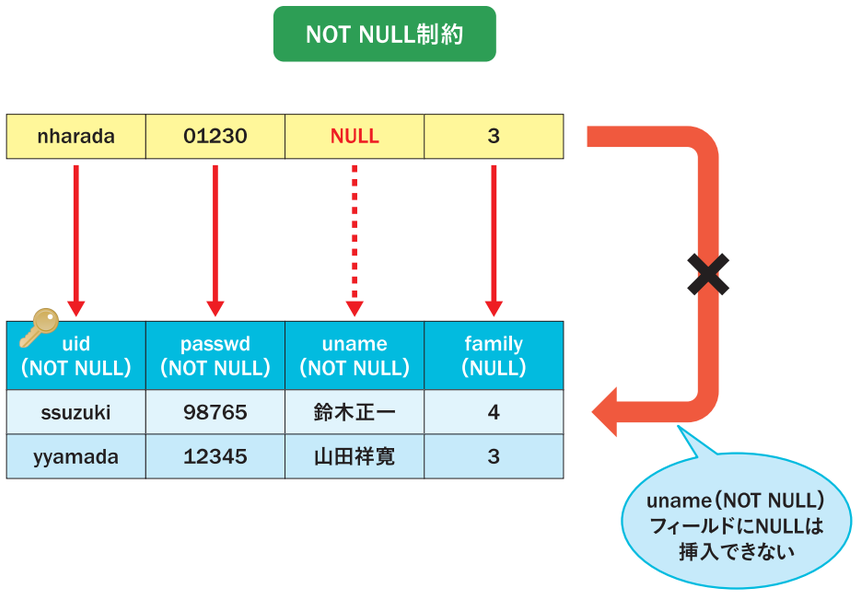
フィールド名	データ型	概要
uid	VARCHAR (7)	ユーザID
passwd	VARCHAR (15)	パスワード
uname	VARCHAR (20)	ユーザ名
family	INT	家族の人数

4 NOT NULL制約

予習 NOT NULL制約について理解する

NULL値とは、値が何も定義されていない状態（未定義な値）を意味します。しかし、フィールドによっては必ず何かしら値を入力してほしいということもあるでしょう。

たとえば、先ほどのusrテーブルでpasswdフィールド（パスワード）やunameフィールド（ユーザ名）に意味のある値が入力されていないのは不都合です。このような場合には、フィールドに**NOT NULL**制約を設定しておきましょう。NOT NULL制約を設定することで、フィールドの値としてNULL値を禁止することができます。



体験 NOT NULL制約を設定しよう

1 mysqlクライアントを起動する

mysqlクライアントを起動してパスワードを入力し❶、basicデータベースに移動します❷。

```
PS C:\Users\nami-> mysql -u myusr -p;
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE basic;
Database changed
mysql>
```

```
PS C:\Users\nami-> mysql -u myusr -p↵
Enter password: *****
```

❶ コマンドとパスワードを入力してそれぞれ Enter キーを押す

```
mysql> USE basic;
```

❷ 入力して Enter キーを押す

2 NOT NULL制約を設定する

usrテーブルのpasswd、unameフィールドに対して、NOT NULL制約を設定します。右のように入力してALTER TABLE命令を実行します❶。図のように「Query OK, …」と表示されれば成功です。

```
mysql> ALTER TABLE usr
    -> MODIFY passwd varchar(15) NOT NULL,
    -> MODIFY uname varchar(20) NOT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

成功メッセージが表示される

❶ 入力して Enter キーを押す

```
mysql> ALTER TABLE usr↵
    -> MODIFY passwd varchar(15) NOT NULL,↵
    -> MODIFY uname varchar(20) NOT NULL;
```

3 フィールド情報を確認する

usrテーブルのフィールド情報を確認してみます。右のように入力してSHOW FIELDS命令を実行します❶。すると、usrテーブルに含まれるフィールド情報の一覧が表示されるので、passwd、unameフィールドのNull欄に「NO」と表示されていることを確認してください。

```
mysql> SHOW FIELDS FROM usr;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| uid    | varchar(7)  | NO   | PRI | NULL    |       |
| passwd | varchar(15) | NO   |     | NULL    |       |
| uname  | varchar(20) | NO   |     | NULL    |       |
| family | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql>
```

「NO」が表示される

❶ 入力して Enter キーを押す

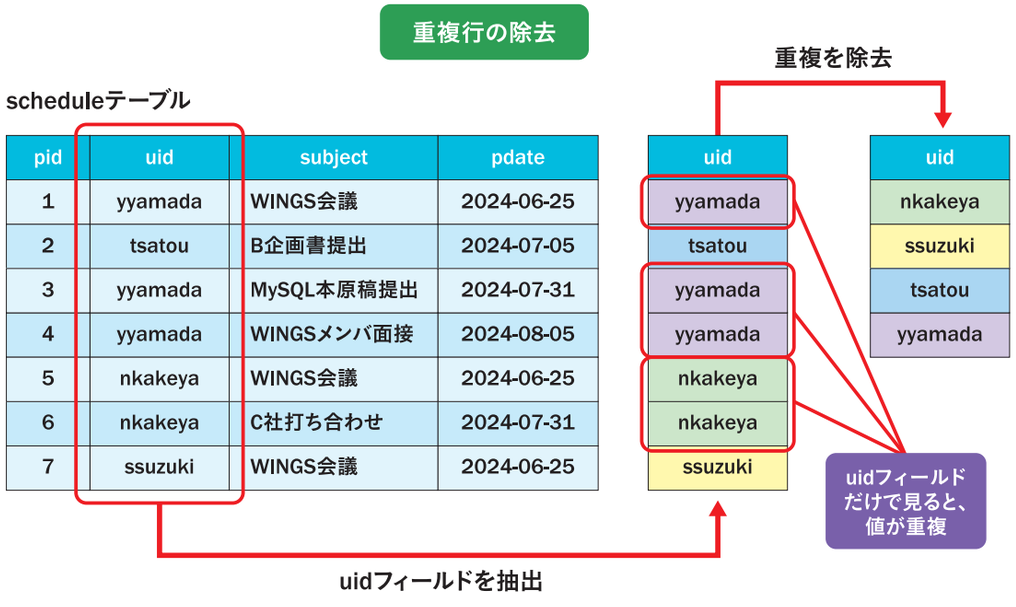
```
mysql> SHOW FIELDS FROM usr ;
```

2 重複の除去

予習 重複行を除去する方法を理解する

テーブルから特定のフィールドだけを取り出した場合、レコードの内容が重複することがあります。このようなケースでは、多くの場合、重複は取り除いて、一意なレコードだけにまとめたいことがほとんどです。

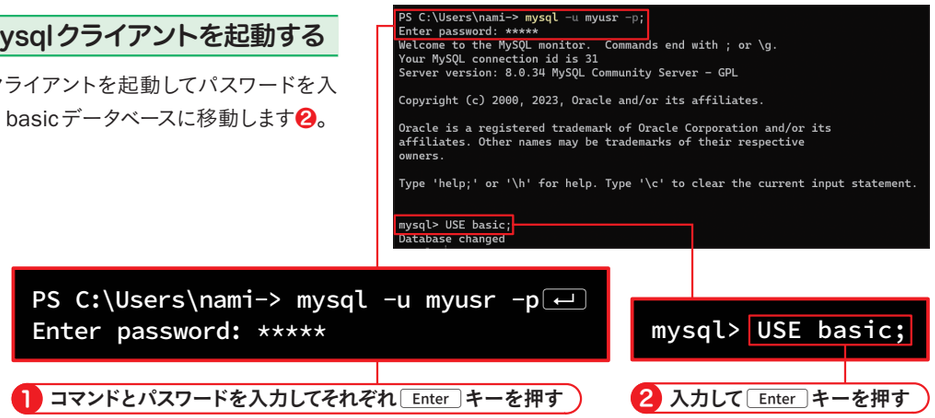
ここでは、SELECT命令で取り出したレコードから重複行を取り除く**DISTINCT**というキーワードについて学びます。構文自体は難しいものではありませんが、よく利用する構文ですので、ここできちんと使い方を覚えておきましょう。なお、レコードを取り除くといっても、レコードそのものが削除されるわけではないので、安心して動作を確認してみてください。



体験 重複を除去しよう

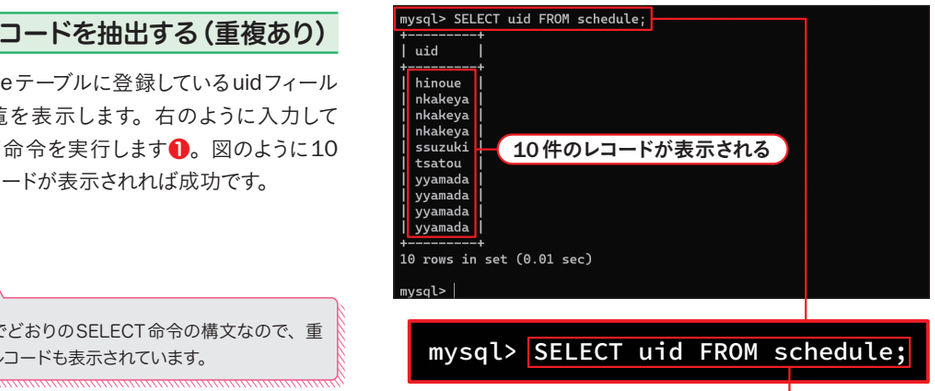
1 mysqlクライアントを起動する

mysqlクライアントを起動してパスワードを入力し❶、basicデータベースに移動します❷。



2 レコードを抽出する（重複あり）

scheduleテーブルに登録しているuidフィールドの一覧を表示します。右のように入力してSELECT命令を実行します❶。図のように10件のレコードが表示されれば成功です。



Tips

これまでどおりのSELECT命令の構文なので、重複したレコードも表示されています。

3 レコードを抽出する（重複なし）

scheduleテーブルに登録しているuidフィールドの一覧を、重複なしで表示します。右のように入力してSELECT命令を実行します❶。図のように5件のレコードが表示されれば成功です。

Tips

SELECT命令にDISTINCTというキーワードを追加しています。

理解 GROUP BY句と集計関数

レコードを集計するための構文(GROUP BY句)

テーブルの内容を特定のキー(フィールド)でグループ化するには、GROUP BY句を使います。GROUP BY句の構文は、次のとおりです。

▼構文

```
SELECT フィールド名, ..., 集計式, ... FROM テーブル名 GROUP BY グループキー
```

「<テーブル名>の内容を<グループキー>でグループ化し、その結果を<フィールド名, ..., 集計式, ...>のように取り出しなさい」という意味ですね。

グループキーは、その名のとおり、グループ化のときに使用するキーとなるフィールドのことです。手順2ではグループキーとして「uid」を指定しているので、uidフィールドの値が等しいレコードが1つのグループとしてまとめられました。その結果、5つのグループが表示されています。

pid	uid	subject	pdate	
1	yyamada	WINGS会議	2024-06-25	…
2	tsatou	B企画書提出	2024-07-05	…
3	yyamada	MySQL本原稿提出	2024-07-31	…
4	yyamada	WINGSメンバ面接	2024-08-05	…
5	nkakeya	WINGS会議	2024-06-25	…
…	…	…	…	…

uidをキーとしてグループ化

uid = 'hinoue'のグループ
→ 1件のレコード

uid = 'nkaneya'のグループ
→ 3件のレコード

uid = 'ssuzuki'のグループ
→ 1件のレコード

uid = 'tsatou'のグループ
→ 1件のレコード

uid = 'yyamada'のグループ
→ 4件のレコード

さまざまな集計関数

一般的に、GROUP BY句は集計関数とセットで利用します。先ほどの構文で「集計式」とある部分は、集計関数を用いて記述しました。手順2では「COUNT(*)」という集計式を記述しています。これは、「各グループに属するレコードの件数を数える」という意味です。

このように、集計関数を利用することで、GROUP BY句で束ねたレコードの件数や最大/最小値、平均値などを求めることができます。関数については改めて**6-5**で紹介しますが、とりあえずは決まった機能を持った道具とでも思っておきましょう。おもな集計関数は、次のとおりです。

関数	求める値
AVG(フィールド名)	平均値
COUNT(フィールド名)	件数
MAX(フィールド名)	最大値
MIN(フィールド名)	最小値
SUM(フィールド名)	合計値

なお、COUNT関数は指定したフィールドの「NULLでない件数」のみを数えますので、要注意です。たとえば、手順2で「COUNT(memo)」のようにNULLが含まれるフィールド名(memo)を指定してしまうと、結果が違ってしまいます。NULLを意識せずにレコード件数を無条件にカウントしたい場合には、「*」(アスタリスク)を指定するか、主キー列を指定してください。

まとめ

- レコードをグループ化するには、SELECT命令のGROUP BY句を使う
- レコードを集計するには、COUNT、AVG、SUMなどの集計関数を使う

理解

外部結合について

左外部結合と右外部結合

外部結合とは言っても、結合の構文は内部結合とほとんど同じです。

▼構文

```
SELECT フィールド名, ... FROM テーブル名1 LEFT OUTER JOIN テーブル名2
  ON テーブル名1.主キー = テーブル名2.外部キー
  [WHERE／ORDER BY句など]
```

基本的に、先ほど紹介したINNER JOINの部分がLEFT OUTER JOINに変わっただけです。これによって、「双方のテーブルでキーが一致するレコード」＋「左テーブルのすべてのレコード」を取り出すことができます。ここで言う「左」とは、LEFT OUTER JOINの左に記述したテーブル（つまり、＜テーブル名１＞）のことです。
例では、scheduleテーブルとusrテーブルとを外部結合しており、usrテーブルが左テーブルに該当します。

SELECT命令ではscheduleテーブルのsubjectフィールド、scheduleテーブルのpdateフィールド、左テーブルであるusrテーブルのunameフィールドを、それぞれ取得しています。しかし、取り出したレコードの中には、scheduleテーブルに対応するキーがないレコードもあります。その場合、scheduleテーブルのフィールドは、実行結果のように「NULL」で表示されます。

LEFT OUTER JOIN句の代わりにRIGHT OUTER JOIN句を使うことで、「双方のテーブルでキーが一致するレコード」＋「右テーブルのすべてのレコード」を取り出すこともできます。

LEFT OUTER JOIN句による結合を**左外部結合**、RIGHT OUTER JOIN句による結合を**右外部結合**と呼んで区別する場合もあります。以下は、例のSELECT命令を右外部結合で書き換えたものです。

```
mysql> SELECT s.subject, s.pdate, u.uname FROM schedule AS s
    -> RIGHT OUTER JOIN usr AS u ON s.uid = u.uid;
```

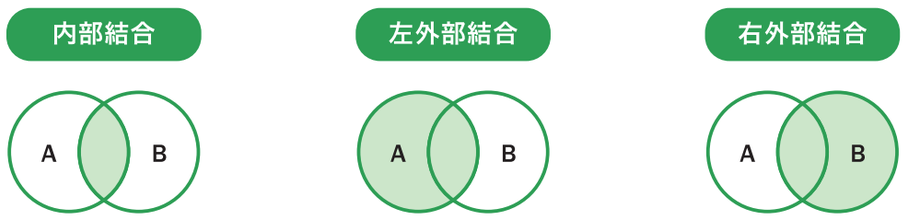

SELECT命令を見てもわかるように、左／右外部結合はデータを左／右どちらのレコードを中心に取り出すかの違いだけで、本質的な違いはありません。できるだけどちらかを統一して使うほうが混乱も少ないでしょう。
以下に、内部結合と外部結合の考え方を図にまとめておくことにします。

テーブルA

aid	memo	bid
1	○	B
2	△	D
3	×	F
4	□	H

テーブルB

bid	note
A	あああ
B	いいい
C	ううう
D	えええ



aid	memo	bid	note
1	○	B	いいい
2	△	D	えええ

aid	memo	bid	note
1	○	B	いいい
2	△	D	えええ
3	×	F	NULL
4	□	H	NULL

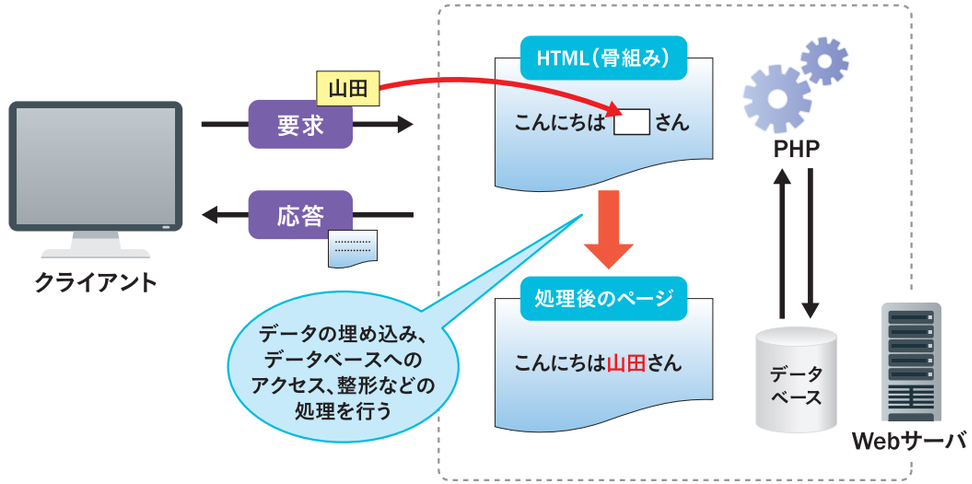
aid	memo	bid	note
NULL	NULL	A	あああ
1	○	B	いいい
NULL	NULL	C	ううう
2	△	D	えええ

まとめ

- 外部結合は、結合するテーブルの両方に存在するレコードと、左右いずれかのテーブルのレコードすべてを取り出す結合のこと
- 外部結合を行うには、SELECT命令のOUTER JOIN句を使う

PHP

HTMLは、あくまでページをどのように表示するかを決めるだけの言語です。しかし、データベースと連携するには、データベースにアクセスして結果を受け取ったり、その結果を見やすい形に整形したりといった機能が必要となります。これらのしくみを記述するための言語が**PHP**（PHP:Hypertext Preprocessor）です。ほかにもC#やJava、Ruby、Pythonといった言語を使うこともできますが、PHPは、そのわかりやすさと高機能性から初心者から上級者まで幅広い層に人気のある言語です。

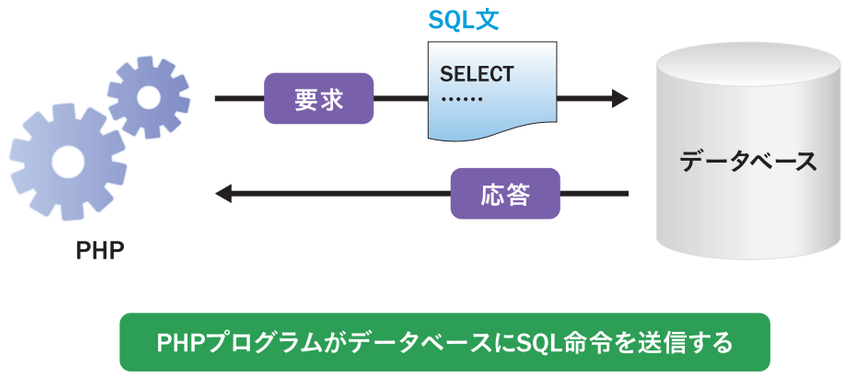


Webサーバは何かしら要求を受け取ると、PHPで書かれたプログラムを動作し、データベースと通信を行い、最終的な結果をHTML文書に整形したものをクライアントに返します。HTMLはあらかじめ用意しておくこともできますし、PHPが動的に出力することもできます。具体的な動きは後から見ますが、「しくみ」を担当するのがPHP、「骨組み」を担当するのがHTMLと考えると、わかりやすいかもしれません。

なお、PHPはあらかじめWebサーバにインストールしておく必要があります。Apacheをインストールしたあとに、付録の**A-3**を参照してPHPをインストールしてください。

SQLとPHP

これまで学んできたSQLは、アプリの世界では不要なのでしょうか。
いえいえ、そのようなことはありません。Webブラウザが理解できる言語がHTML、Webサーバが理解できる言語がPHPであるように、データベースが理解できる言語はSQLです。つまり、PHPからデータベースにアクセスする場合もSQLは必須なのです。
これまでは人手で直接にSQLを書いていたわけですが、アプリではPHPプログラム（**スクリプト**とも言います）が代わりにSQL命令をデータベースに送信します。PHPを使うからと言って、SQLがなくなるわけではありません。



PHPプログラムがデータベースにSQL命令を送信する

まとめ

- ▶Webブラウザでページをどのように表示するかを決めるのはHTML
- ▶Webサーバ側でデータベースへのアクセスなどの処理を行うのはPHP
- ▶PHPからデータベースにアクセスする場合もSQLは必要

さまざまなフォーム要素

<form>～</form>の中には、テキストボックスやラジオボタン、ドロップダウンリストなど、さまざまな入力要素を配置できます。ここでは、その中でも代表的な要素であるテキストボックスとサブミットボタンを使っています。

入力要素の多くは、<input>タグで作成できます。出力する要素を切り替えるには、type属性の値を変更するだけです。手順1で入力したコードの11行目では「<input type="text"～>」としてテキストボックスを、12行目では「<input type="submit"～>」としてサブミットボタンを出力しています。サブミットボタンとは、フォームの内容を<form>タグのaction属性で指定された相手（ここではform.php）に送信するための「送信」ボタンのことです。

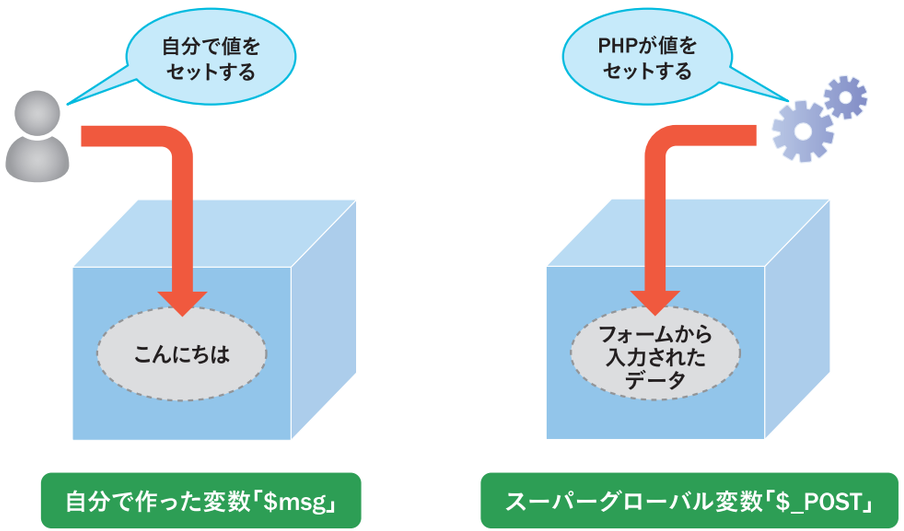
そのほか、「type="password"」とすればパスワード入力ボックス（入力文字が伏字になる）、「type="radio"」とすればラジオボタン、「type="checkbox"」とすればチェックボックスを出力します。

<input>タグのname属性は要素の名前を表す属性で、あとから入力値を取得する場合のキーとなります。手順1で入力したコードの10行目では「<input type="text" name="name"～>」としていますので、このテキストボックスの内容が「name」というキーでアクセスできるようになります。

そのほかにも、その要素の値やデフォルト値を表すvalue属性、ボックスの幅を表すsize属性などがありますが、本書では割愛します。詳しくは、8-3で紹介した「HTMLクイックリファレンス」を確認してみるとよいでしょう。

フォームから送信されたデータを取得する

フォームの書き方を理解できたところで、いよいよPHPスクリプトで、フォームから送信した値を取り出してみましょう。入力値を取得するのは、**スーパーグローバル変数**と呼ばれる特殊な変数の役割です。
スーパーグローバル変数は、普通の変数とは異なり、自分で用意しなくてもあらかじめ値がセットされているという特徴があります。



スーパーグローバル変数にはいくつかの種類がありますが、<form method="POST">形式のフォームから送信された入力値は、$_POSTという変数から取り出せます。

▼構文

```
$_POST[キー名]
```

ここで言う「キー名」は、先ほど<input>タグのname属性で指定された値です。つまり、「$_POST['name']」で、テキストボックス「name」に入力された値を取得できるわけです。
ちなみに、<form method="GET">形式で送信された値は、$_POSTの代わりに$_GETというスーパーグローバル変数で取得できます。

続いて、「$_POST['name']」の前にある「htmlspecialchars」について説明します。