

データベース入門 資料 12

データ定義とデータ更新

2011 年 7 月 25 日

目 次

1	準備—データベースへの接続	1
2	データ定義とデータ更新	1
2.1	データ定義	1
2.1.1	データ型名	1
2.1.2	テーブルの作成 — CREATE TABLE	1
2.1.3	テーブルの削除 — DROP TABLE	2
2.2	データ更新	3
2.2.1	行を挿入: INSERT	3
2.2.2	値の変更: UPDATE	3
2.2.3	行の削除: DELETE	3
3	テキストファイルの内容をテーブルに挿入する	4
3.1	psql の copy コマンド	4
3.1.1	基本的な使い方	4
3.1.2	区切り文字の指定	5
3.1.3	参考—テーブルからファイルへの書き出し	5
3.2	テキストファイルを加工してテーブルに入れる	5
4	演習問題	6

1 準備—データベースへの接続

この資料では、新たに表（テーブル）¹を作成して、そこにデータを入れる方法を学びます。そのために、これまでの学習に使ってきたデータベース “gamera” に代えて、データの書き込みが許可されているデータベース “dbnyumon” を使います。

psql の引数には、使いたいデータベースを指定できるので、

```
psql dbnyumon
```

を実行して、データベース dbnyumon に接続しましょう。

dbnyumon に何もテーブルがないことを、テーブルの情報を調べる psql のコマンドで、確認してみます。

```
dbnyumon=> \d
No relations found.2
```

なお、今回も受講者全員が同じ名前のデータベースを利用しますが、その実体は受講者毎に異なっています。そのため、他の人が作ったテーブル等は自分のデータベースからは見えません。

2 データ定義とデータ更新

2.1 データ定義

この節では、テーブルの作成と削除の方法を扱います。

テーブルを作成するには、テーブルにどんな列を用意するか、また列に入れるデータの型や制約条件をどうするか、などを決める必要があります。これらをスキーマ (schema) と呼びます。スキーマは、いわばデータの入れ物であるテーブルの「骨格」です。

2.1.1 データ型名

テーブルの各列のデータ型 (type) は、テーブルの設計・作成の段階で定めます。したがって、テーブルを作成するためには、SQL で使えるデータ型の名前を知る必要があります。ただし、データ型の詳細は、使っているデータベースソフトウェア毎に異なりますので、ここでは次の二つの型のみを使用することとします。

1. 整数型：int
2. 可変長文字列型：varchar

2.1.2 テーブルの作成 — CREATE TABLE

テーブル作成の基本的な構文は次のとおりです。

```
CREATE TABLE テーブル名 (列名 型, 列名 型, ...)
```

¹関係データベースモデルの用語ではリレーションスキーマ (relation schema)

²テーブルが見つかりません。

例 名前 (namae) 列のみを含む野菜テーブル (yasai) を作る。

```
dbnyumon=> CREATE TABLE yasai (namae varchar) ;
CREATE TABLE
```

```
dbnyumon=> \d
          List of relations
 Schema | Name   | Type  | Owner
-----+-----+-----+-----
 (user) | yasai  | table | (user)
(1 row)
```

```
dbnyumon=> \d yasai
          Table "(user).yasai"
 Column |          Type          | Modifiers
-----+-----+-----+-----
 namae  | character varying     |
```

例 名前 (namae) 列と値段 (nedan) を含む果物テーブル (kudamono) を作る。

```
dbnyumon=> CREATE TABLE kudamono (namae varchar, nedan int) ;
CREATE TABLE
```

```
dbnyumon=> \d
          List of relations
 Schema | Name     | Type  | Owner
-----+-----+-----+-----
 (user) | kudamono | table | (user)
 (user) | yasai    | table | (user)
(2 rows)
```

```
dbnyumon=> \d kudamono
          Table "(user).kudamono"
 Column |          Type          | Modifiers
-----+-----+-----+-----
 namae  | character varying     |
 nedan  | integer                |
```

2.1.3 テーブルの削除 — DROP TABLE

DROP TABLE テーブル名

例 yasai テーブルを削除する。

```
dbnyumon=> DROP TABLE yasai ;
```

実行後、\d コマンドで結果を確かめてください。

2.2 データ更新

テーブルを作成すれば、そこにデータを入れることができます。テーブル内のデータをインスタンス (instance) と呼びます。すなわち、インスタンスはテーブルの各行です。この章では、テーブルへの行の挿入と削除、および値の変更方法を扱います。

psql コマンドには SQL のヘルプを見るコマンド \h がありますので、

```
\h insert
```

などで適宜、書式を確認してください。

操作の結果は、必ず SELECT 文で確認してください。

2.2.1 行を挿入: INSERT

```
INSERT INTO テーブル名
(列名 1, 列名 2, 列名 3 ...)
VALUES
(値 1, 値 2, 値 3 ...)
```

例

```
INSERT INTO kudamono (namae,nedan) VALUES ('banana', 280);
INSERT INTO kudamono (namae,nedan) VALUES ('budo', 350);

SELECT * FROM kudamono;
```

2.2.2 値の変更: UPDATE

```
UPDATE テーブル名
SET 列名 1 = 値 1,
    列名 2 = 値 2,
    ...
WHERE 検索条件
```

例

```
UPDATE kudamono SET nedan = 300 WHERE namae = 'banana';
```

2.2.3 行の削除: DELETE

```
DELETE FROM テーブル名
WHERE 検索条件
```

「WHERE 検索条件」を省略すると、テーブル内の全ての行が削除される。

例

```
DELETE FROM kudamono WHERE namae = 'budo';
```

3 テキストファイルの内容をテーブルに挿入する

ここでは、データが記入されているテキストファイルがあるときに、その内容を一度の操作でテーブルに挿入する方法を紹介します。

3.1 psql の copy コマンド

PostgreSQL の psql には、ファイルの内容をデータベースのテーブルに挿入したり、逆にテーブルの内容をファイルに書き出すために、コマンド `\copy` が用意されています。

3.1.1 基本的な使い方

テキストファイルの内容をデータベースのテーブルに挿入する際の基本的な `\copy` コマンドの書式は

```
\copy テーブル名 from ファイル名
```

です。この場合、ファイルの中身は、各列がタブ文字で区切られた表形式になっている必要があります。ファイル名として、パス名を指定することも可能です。copy は SQL コマンドではありませんから、行末の ; が不要であることも注意しておきましょう。

例 次の内容を持つテキストファイル `/pub/db/banana` があるとします。(列間の空白はタブ文字)。

```
4      banana
5      ichigo
6      nashi
```

また、次の SQL コマンドで、二つの列を有するテーブル `shina` が作成済みであるとします。

```
CREATE TABLE shina (
    code varchar,
    shohin varchar
);
```

テーブル `shina` にファイル `banana` の中身を挿入するには、

```
dbnyumon=> \copy shina from /pub/db/banana
```

とします。ここで `dbnyumon=>` は、もちろん、データベース `dbnyumon` を利用している際の psql コマンドが表示するプロンプトです。

結果は次のようになります。

```

dbnyumon=> SELECT * from shina ;
  code | shohin
-----+-----
    4  | banana
    5  | ichigo
    6  | nashi
(3 rows)

```

注意 \copy コマンドはテーブルにファイルの内容を新たに追加します。そのため、既にデータが入っているテーブルを新しいデータで置き換えたい場合には、予め DELETE でデータを全て削除するか、DROP TABLE, CREATE TABLE でテーブルを作り直しておく必要があります。

3.1.2 区切り文字の指定

ファイルの中に、タブ以外の列の区切り文字が使われている場合には、次の形式で \copy を実行します。

```
\copy テーブル名 from ファイル名 using delimiters '区切り文字'
```

例 ファイル /pub/db/kaki の中身が次のように — で区切られているとします。

```

7|kaki
8|kuri

```

これをテーブル shina に入れるには

```
\copy shina from /pub/db/kaki using delimiters '|'
```

とします。

3.1.3 参考—テーブルからファイルへの書き出し

\copy コマンドを実行するときに from の代わりに to を指定すれば、データベースのテーブルからファイルにデータを書き出すことができます。

3.2 テキストファイルを加工してテーブルに入れる

次の内容をもつファイル /pub/db/lemon があるとします。列はタブ文字で区切られています。

```

lemon    9
pineapple 10

```

これを先ほどのテーブル shina に入れることを考えましょう。単純に

```
dbnyumon=> \copy shina from /pub/db/lemon
```

とすると、次の実行結果のとおり、列 code に商品 (果物) の名前が入り、列 shohin にコード (数字) が入ることになってしまいます。

```
dbnyumon=> SELECT * from shina ;
      code      | shohin
-----+-----
      4          | banana
      5          | ichigo
      6          | nashi
      7          | kaki
      8          | kuri
      lemon      | 9
      pineapple  | 10
(7 rows)
```

このような場合、予め UNIX のコマンドを使って、/pub/db/lemon の中身を加工してから、\copy コマンドを実行すればいいです。

例えば、UNIX のコマンド行で (psql を一度終了する³)

```
gawk '{print $2, $1}' /pub/db/lemon >rlemon
```

を実行すれば、中身が

```
9 lemon
10 pineapple
```

であるファイル rlemon が出来上がります。ここで、gawk の出力における列の区切りが、デフォルトでは (特に指定しない場合には) 空白であることに注意してください。このため、ファイル rlemon 内の列は空白で区切られます。

その後、psql を起動して、

```
\copy shina from rlemon using delimiters ' '
```

とすれば、列の区切り文字が空白であると解釈されて、各列に正しくデータが入ります。

テキストファイルの中から特定の列を取り出したり、列の順番を入れ替えるには awk が使えますし、特定の行のみを取り出すには grep が使えます。これらの出力をファイルに入れたければ、リダイレクト (>) が使えます。

これらの UNIX コマンドの使い方を忘れていたら、本授業前半の資料を参照してください。

4 演習問題

1. 下記のテーブルを作成し、正しくできたか確認しなさい。

- テーブル名： yasai
- 列名とデータ型

shohin	varchar
nedan	int
- 行 (インスタンス)

³実行中のコマンドを中断して再開する方法を知っていれば、それでも可。

```
imo 100
ninjin 110
tomato 390
```

2. yasai テーブルから, ninjin と tomato の行のみを出力しなさい。その際, 値段を検索の条件として利用すること。
3. ファイル /pub/db/postcode/yubinpost.tab は, 郵便屋さんが, 郵便番号ごとに郵便を配達する順番をあらわしたものである。

このファイルの各行には, 新郵便番号と配達順がタブ文字区切りで格納されている。(less コマンドで確認せよ)

このファイルの内容を, psql の \copy コマンドを使って, すべてデータベースに入れなさい。格納先のテーブル名は junban とする。テーブル junban は次の列 (列名・型) を持つこととする。

```
code varchar
junjo int
```

4. テーブル junban に, 郵便番号が 0420916, 配達順序が 7 の行を挿入しなさい。
5. テーブル junban の中の, 郵便番号 0410802 の配達順序を 10 に変更しなさい。
6. 以下の指示に従って, テキストファイルからデータベースにデータを入れなさい。
 - (a) ファイル /pub/db/postcode/rensyuTable の内容を less コマンドを使って確認しなさい。
このファイルの各列は

自治体コード, 旧郵便番号, 新郵便番号, 都道府県 (カナ), 市町村 (カナ), 町名区分 (カナ), 都道府県 (漢字), 市町村 (漢字), 町名区分 (漢字)

を表している。

- (b) ファイル rensyuTable の中の, 01202 を含む行のみから, 次の列を抜き出し, 適当な名前のファイルに入れなさい。

自治体コード, 新郵便番号, 市町村 (漢字), 町名区分 (漢字)

- (c) 前項で作成したファイルの内容を, psql の \copy コマンドを使ってデータベースに入れなさい。

格納先のテーブル名は main とする。main は, 自治体コード, 新郵便番号, 市町村 (漢字), 町名区分 (漢字) を格納するために, 次の列 (名前・型) を持つこととする。

```
code varchar
zip varchar
city varchar
chomei varchar
```

7. テーブル main と junban を使って, 配達順序と町名の一覧を配達順序順に表示するための SQL 文を実行しなさい。