# 情報科学演習 資料 13
# データ定義とデータ更新

2019 年 7 月 22 日

## 目 次

# 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）を作成して[1]，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 “db_a” 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 “eis” 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 eis

を実行して，データベース eis に接続しましょう。

eis 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。

```
eis=> \d
リレーションがありません。( No relations found. )
```

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] データの入った表をいきなり作るのではなく，まずは表の枠組みを作る。この枠組みを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) という

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

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

eis=> \d
          リレーションの一覧
 スキーマ | 名前  |    型     | 所有者
---------+-------+---------+--------
 (user)  | yasai | テーブル | (user)
(1 行)

eis=> \d yasai
       テーブル "(user).yasai"
 カラム |         型          | 修飾語
--------+--------------------+--------
 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。

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

eis=> \d
           リレーションの一覧
 スキーマ |   名前   |    型     | 所有者
----------+----------+----------+--------
 (user)   | kudamono | テーブル | (user)
 (user)   | yasai    | テーブル | (user)
(2 行)

eis=> \d kudamono
      テーブル "(user).kudamono"
 カラム |         型          | 修飾語
--------+--------------------+--------
 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。

```
eis=> 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。この節では，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。

以下の操作の結果は，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
```

**注意**　「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ます。

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。この方法は大量のデータがテキストファイルの形で既に存在しているときに便利です。

## 3.1　psql の copy コマンド

SQL にはファイルとテーブルの間でデータをコピーするために COPY というコマンドがありますが，この授業で利用しているデータベースソフトウェア PostgreSQL では，このコマンドは一般のユーザーが自由に使えるコマンドではありません。

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ので，この授業ではこちらを使います。

### 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/eis/sql/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，

```
eis=> \copy shina from /pub/eis/sql/banana
```

とします。ここで eis=> は，もちろん，データベース eis 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。

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

**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 delimiter '区切り文字'
```

**例** ファイル /pub/eis/sql/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/eis/sql/kaki delimiter '|'
```

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/eis/sql/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

```
eis=> \copy shina from /pub/eis/sql/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。

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

このような場合，予め UNIX のコマンドを使って，/pub/eis/sql/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[2])

```
awk '{print $2, $1}' /pub/eis/sql/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，awk 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 delimiter ' '
```

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。

[2]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。

# 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)

     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/eis/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/eis/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。