データベース入門 資料 10
データ型とテーブルの結合

2011 年 7 月 4 日

目 次

1 データ型に関する注意

1.1 データ型とは

テーブル population には，函館市周辺の自治体の名前，人口，世帯数が入っています (平成 17 年国勢調査時のもの)。

```
SELECT * FROM population ;
  name   | popul  | house
---------+--------+--------
 函館市  | 294264 | 128411
 七飯町  |  28424 |  10363
 北斗市  |  48056 |  17779
 鹿部町  |   4919 |   1646
 森町    |  19149 |   7363
```

このテーブルでは，列 name (自治体名) が文字列であるのに対し，列 popul (人口) と列 house (世帯数) は整数です。文字列や整数のような値の種類を**データ型**といいます。なお,「値」という言葉は数値以外のものに対しても使われます。

本章では，SQL に用意されているデータ型の詳細には触れませんが，データ型には文字のための型と数値のための型があることは覚えておいてください。

1.2 SQL 文における値の記述法

SQL ではデータ型に応じて値の指定方法が異なります。

1. 文字を指定するには，それを ' (単一引用符) で囲む。

   これまでに使ってきた方法です。このルールに反すると，次のようなエラーが起きます。

   ```
   SELECT * FROM population WHERE name = 函館市;
   ERROR:  column "函館市" does not exist
   ```

   (エラー：列 “函館市” は存在しない — 函館市を ' で囲まなかったので列名とみなされた)

2. 数値等の文字以外の値は ' で囲まない。

   ```
   SELECT * FROM population WHERE popul > 200000 ;
     name   | popul  | house
   ---------+--------+--------
    函館市  | 294264 | 128411
   ```

1.3 データ型は列毎に決められている

テーブルの各列に格納されるデータのデータ型は，テーブルの設計・作成の段階で，列毎に定められています。

psql には，テーブルの情報を表示する，psql 固有のコマンド

```
\d テーブル名
```

があります。これを使うとテーブル population における各列のデータ型がわかります。\d は SQL 文ではないので，末尾に ; を付けないことに注意してください。

```
gamera=> \d population
       Table "public.population"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 name   | character varying |
 popul  | integer           |
 house  | integer           |
```

ここで Column は列名を，Type はデータ型を意味します。これより，テーブル population では，列 name は character varying (可変長文字列) 型であり，列 popul と列 house は integer (整数) 型であることがわかります。したがって，テーブル population では，列 name のデータを SQL で指定する場合には' (単一引用符) で囲む必要があり，列 popul と house ではその必要はありません。

1.4 数字だから数値とは限らない

テーブル area_code の中身は次のとおりです。

```
SELECT * from area_code;
  name  | code
--------+-------
 函館市 | 01202
 七飯町 | 01337
 北斗市 | 01236
 鹿部町 | 01343
 森町   | 01345
```

一見，列 name は文字の型に，列 code は数値の型に見えますが，実は違います。

```
gamera=> \d area_code
        Table "public.area_code"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 name   | character varying |
 code   | character(5)      |
```

列 name は character varying (可変長文字列) 型で，列 code は character(5) (5 文字からなる固定長文字列) 型です。つまり，数字のみが格納された列であっても，その列が数値の型とは限りません。

SQL 文における単一引用符の必要性は，列のデータ型によって決まりますので，テーブル area_code における列 code の値を指定する場合には，' (単一引用符) で囲む必要があります。

```
SELECT * FROM area_code WHERE code = '01202';
  name  | code
--------+-------
 函館市 | 01202
(1 row)

SELECT * FROM area_code WHERE code = 01202;
 name | code
------+------
(0 rows)
```

なお，列 code (地域コード) が整数型にされていない理由としては，次の点が挙げられます。

- 特定の桁の数字で検索条件を指定する場合，文字列として扱った方が便利である。like を使った文字列パターンの検索は，文字列に対する操作なので，code を整数型にすると like による検索は（原則として）できない。
- 数値ならば，数値演算 (例えば加算等) の結果は，意味のある数値として解釈できるべきである。しかし，地域コードを整数としても演算の結果には数値としての意味はなく，地域コードを整数にする必要性はない。

ここで扱う地域コードはすべて 5 桁に決まっているので，列 code は文字 5 つから成る固定長文字列としています。

2 テーブルの結合

2.1 目的

この章では，複数のテーブルから意味のある結果を得るための方法を学びます。説明に使うテーブルは次の二つです。

1. テーブル area_code

   列 name (地域名): 可変長文字型 (varchar),
   列 code (地域コード): 固定長文字型 (char(5))

```
SELECT * FROM area_code ;
  name  | code
--------+-------
 函館市 | 01202
 七飯町 | 01337
 北斗市 | 01236
 鹿部町 | 01343
 森町   | 01345
(5 rows)
```

2. テーブル n_hospital

   列 code (地域コード): 固定長文字型 (char(5)),
   列 num (地域の病院数): 整数型 (int)

```
SELECT * FROM n_hospital ;
 code  | num
-------+-----
 01202 | 284
 01337 |  19
 01236 |  27
 01343 |   3
 01345 |  14
(5 rows)
```

この二つのテーブルは，地域コードの入った列によって互いに関連づけられています。例えば,「函館市の病院数が 284 である」ことは，二つのテーブルを利用しなければわかりません。本章の目的は，SELECT 文を使って，このような結果を得る方法を習得することです。

2.2 交差結合

SELECT 文において，FROM に続くテーブルの名前を複数指定することができます。

FROM テーブル 1，テーブル 2，...

これにより，FROM に続いて指定したテーブル (テーブル 1，テーブル 2，...) に含まれる行の，すべての組み合わせが得られます。これを交差結合（直積[1]）と呼ぶことがあります。

```
SELECT *
FROM area_code, n_hospital;

  name  | code  | code  | num
--------+-------+-------+-----
 函館市 | 01202 | 01202 | 284
 函館市 | 01202 | 01337 |  19
 函館市 | 01202 | 01236 |  27
 函館市 | 01202 | 01343 |   3
 函館市 | 01202 | 01345 |  14
 七飯町 | 01337 | 01202 | 284
 七飯町 | 01337 | 01337 |  19
 七飯町 | 01337 | 01236 |  27
 七飯町 | 01337 | 01343 |   3
 七飯町 | 01337 | 01345 |  14
 北斗市 | 01236 | 01202 | 284
```

[1]出力の各行は集合論における順序対 (または順序 n 組) に対応します。それを全て集めたものは直積です。

```
 北斗市 | 01236 | 01337 |  19
 北斗市 | 01236 | 01236 |  27
 北斗市 | 01236 | 01343 |   3
 北斗市 | 01236 | 01345 |  14
 鹿部町 | 01343 | 01202 | 284
 鹿部町 | 01343 | 01337 |  19
 鹿部町 | 01343 | 01236 |  27
 鹿部町 | 01343 | 01343 |   3
 鹿部町 | 01343 | 01345 |  14
 森町   | 01345 | 01202 | 284
 森町   | 01345 | 01337 |  19
 森町   | 01345 | 01236 |  27
 森町   | 01345 | 01343 |   3
 森町   | 01345 | 01345 |  14
(25 rows)
```

2.3 等結合

2.3.1 等結合

先の交差結合の結果を元に地域毎の病院数を得るためには，WHERE による検索条件指定を行って，二つのテーブル area_code と n_hosptal に含まれる列 code の値が同じ行だけを抽出します。

複数のテーブルから列を指定する場合，

テーブル名. 列名

の形で，テーブルと列を . (ドット) で区切って記述します。

```
SELECT *
FROM area_code, n_hospital
WHERE area_code.code = n_hospital.code;

  name  | code  | code  | num
--------+-------+-------+-----
 函館市 | 01202 | 01202 | 284
 七飯町 | 01337 | 01337 |  19
 北斗市 | 01236 | 01236 |  27
 鹿部町 | 01343 | 01343 |   3
 森町   | 01345 | 01345 |  14
(5 rows)
```

このように，一方の列の値と，もう一方の列の値が等しい行をつなぎ合わせる操作を等結合といいます。

2.3.2 参考 — JOIN ON

この操作は JOIN と ON という SQL キーワードを使って，次のように書いて実現することもできます。

```
SELECT *
FROM area_code JOIN n_hospital
ON area_code.code = n_hospital.code;
```

2.3.3 参考 — 自然結合 (NATURAL JOIN)

同名の列で等結合した結果から，重複する列を省くことを自然結合といいます。area_code と n_hospital の自然結合は

```
SELECT *
FROM area_code NATURAL JOIN n_hospital;
```

で得られます。

2.3.4 列と行の限定

等結合の結果から，特定の列を指定した順序で得たい場合には，これまでと同様に SELECT の後に列名を明示します。

```
SELECT area_code.code, area_code.name, n_hospital.num
FROM area_code, n_hospital
WHERE area_code.code = n_hospital.code;

 code  |  name  | num
-------+--------+-----
 01202 | 函館市 | 284
 01337 | 七飯町 |  19
 01236 | 北斗市 |  27
 01343 | 鹿部町 |   3
 01345 | 森町   |  14
(5 rows)
```

さらに検索条件を加えることにより，取り出す行を限定できます。

```
SELECT area_code.code, area_code.name, n_hospital.num
FROM area_code, n_hospital
WHERE area_code.code = n_hospital.code
  AND area_code.code = '01202';

 code  |  name  | num
-------+--------+-----
 01202 | 函館市 | 284
(1 row)
```

さらに表示する列と行を限定してみます。

```
SELECT area_code.name, n_hospital.num
FROM area_code, n_hospital
WHERE area_code.code = n_hospital.code
  AND (area_code.code = '01202' OR area_code.code = '01337');

  name  | num
--------+-----
 函館市 | 284
 七飯町 |  19
(2 rows)
```

この SQL 文においては，次の点に注意してください。

1. 表示しない列 (code) も検索条件に含めることが可能
2. AND は OR より評価の優先順位が高いので，必要に応じて括弧を使う

3 別名の利用

3.1 列の別名

テーブルを結合すると，表示した列名の意味がわからなくなることがあります。例えば，先の例では，name や num が何の名前や数なのか，結果を見ただけではわかりません。SELECT に続く列名の指定において，SQL のキーワード AS を使って列に別名を与えれば，列の意味を明確にできます。

```
SELECT area_code.name AS area_name, n_hospital.num AS hospital_num
FROM area_code, n_hospital
WHERE area_code.code = n_hospital.code AND area_code.code = '01202';

 area_name | hospital_num
-----------+--------------
 函館市    |          284
(1 row)
```

3.2 参考 — 相関名 (テーブルの別名)

テーブル名に，より短い別名 (相関名) を与えて，複数のテーブルを扱う SQL 文を短く書くことができます[2]。

相関名は FROM に続くテーブル名の後で AS に続き定義します[3]。定義した相関名は，SELECT に続く列名や WHERE 検索条件で使用できます。

[2]SELECT 文では同一テーブル同士に結合（自己結合）を施すこともできます。その場合，FROM の後ろのテーブル名に重複は許されないので，テーブルに相関名を定義して，異なるテーブル名で結合する必要があります。

[3]相関名のための AS は省略可能。

次の例ではテーブル area_code に相関名 a を，テーブル n_hospital には相関名 h を与えています。

```
SELECT a.name, h.num
FROM area_code AS a, n_hospital AS h
WHERE a.code = h.code AND a.code = '01202';

  name  | num
--------+-----
 函館市 | 284
(1 row)
```

4 演習問題

1. 次の各テーブルについて，各列のデータ型を確認しなさい。また，テーブルに含まれる列と行をすべて表示しなさい。

   (a) テーブル population
   列: name 地域名 ，popul 人口，house 世帯数

   (b) テーブル spring
   列: name 温泉名，area 地域コード

2. 次の出力を得るための SQL 文を実行しなさい。

```
  name  | popul
--------+--------
 函館市 | 294264
(1 row)
```

3. 次の出力を得るための SQL 文を実行しなさい。表示される行の順番は問わない。(以下の設問も同様)

```
  name  | code  |  name  | popul  | house
--------+-------+--------+--------+--------
 函館市 | 01202 | 函館市 | 294264 | 128411
 七飯町 | 01337 | 七飯町 |  28424 |  10363
 北斗市 | 01236 | 北斗市 |  48056 |  17779
 鹿部町 | 01343 | 鹿部町 |   4919 |   1646
 森町   | 01345 | 森町   |  19149 |   7363
(5 rows)
```

4. テーブル spring と area_code から，次の出力を得るための SQL 文を実行しなさい。

```
     name     | location
--------------+----------
```

```
 谷地頭温泉     | 函館市
 湯の川温泉街 | 函館市
 東大沼温泉郷 | 七飯町
 川汲温泉郷     | 函館市
 戸井温泉       | 函館市
 せせらぎ温泉 | 北斗市
 鹿部温泉郷     | 鹿部町
 濁川温泉郷     | 森町
 仁山温泉       | 七飯町
(9 rows)
```

5. テーブル area_code と population を使って，人口 (列 popul) が 30000 人以上の地域のコードと人口のみを表示しなさい。

6. 次の出力を得るための SQL 文を実行しなさい。area_code, population, n_hospital の三つのテーブルを用いる。

```
  name   | code  |  name   | popul  | house  | code  | num
---------+-------+---------+--------+--------+-------+-----
 函館市 | 01202 | 函館市 | 294264 | 128411 | 01202 | 284
 七飯町 | 01337 | 七飯町 |  28424 |  10363 | 01337 |  19
 北斗市 | 01236 | 北斗市 |  48056 |  17779 | 01236 |  27
 鹿部町 | 01343 | 鹿部町 |   4919 |   1646 | 01343 |   3
 森町    | 01345 | 森町    |  19149 |   7363 | 01345 |  14
(5 rows)
```

7. 次の出力を得るための SQL 文を実行しなさい。

```
  name   | code  | popul  | house  | hospital
---------+-------+--------+--------+----------
 函館市 | 01202 | 294264 | 128411 |      284
 七飯町 | 01337 |  28424 |  10363 |       19
 北斗市 | 01236 |  48056 |  17779 |       27
 鹿部町 | 01343 |   4919 |   1646 |        3
 森町    | 01345 |  19149 |   7363 |       14
(5 rows)
```