情報科学演習／データベース入門 A 資料 10
問い合わせの基礎 — SELECT

2017 年 6 月 26 日

目 次

1 準備

SQL 文を効率よく打ち込み実行できるように，資料 9-2. DBMS と SQL のうち，特に

- 第 2 章 psql コマンドの使い方
- 第 1 章第 4 節 SQL 文の記述に関する注意

を復習してください。

2 SELECT 文の基本的な使い方

SQL でデータベースへの問い合わせを行ってデータを取り出すには SELECT 文を使います。

2.1 列の選択

SELECT 文の最も基本的な使い方は

```
SELECT 列名 FROM テーブル名;
```

です。この書式ではテーブルからすべての行を取り出します。表示する列は，列名をカンマ区切りで指定して特定します。ただし，列名部分に “列名” 以外のものを書くこともあります。

2.1.1 すべての列を表示

列名として * を指定すれば，すべての列を表示します。

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

SELECT * FROM postcode;
(実行結果は省略。一画面を越える結果はページャ (more や less) で表示されるので，
表示を終了するには，ページャの終了命令である q を打つ)
```

2.1.2 特定の列のみを表示

SELECT に続く列名の箇所に，列名あるいはカンマ区切りの列名リストを指定します。

```
SELECT name FROM area_code;
SELECT new_post_code, ken_kana FROM postcode;
```

2.2 重複行の除去: DISTINCT

表示結果から，重複する行を除外するには DISTINCT キーワードを使って

```
SELECT DISTINCT 列名 FROM テーブル名;
```

とします。

テーブル spring の属性名 (列名) と, テーブル内のデータ（インスタンス）は次のとおりです。

```
     name      | area
---------------+-------
 谷地頭温泉    | 01202
 湯の川温泉街  | 01202
 東大沼温泉郷  | 01337
 川汲温泉郷    | 01202
 戸井温泉      | 01202
 せせらぎ温泉  | 01236
 鹿部温泉郷    | 01343
 濁川温泉郷    | 01345
 仁山温泉      | 01337
(9 rows)
```

これに対して

```
SELECT area FROM spring;
```

を実行すると，

```
 area
-------
 01202
 01202
 01337
 01202
 01202
 01236
 01343
 01345
 01337
(9 rows)
```

となり，複数の同じ値が出力されますが，

```
SELECT DISTINCT area FROM spring;
```

とすれば，重複する行は表示されません。

2.3 行の選択: WHERE 句の基本

SELECT 文で検索条件を指定して，特定の行のみを取り出すには，これまでの SELECT 文に続いて

WHERE 検索条件

を追加して記述します。ここで検索条件の基本形は

列名 比較演算子 値

または，これを論理演算子 AND や OR で結んだものです[1]。

2.3.1 比較演算子

まず SQL における比較演算子をまとめます。

=	等しい
<>	等しくない
<	小なり
<=	以下
>	大なり
>=	以上

大小を比較する < や > 等の比較演算子は，数値以外にも，順序を持つ値一般に対して使用できます。例えば，計算機内での文字の表現は文字コードであり，文字コード同士には順序がありますから，文字や文字列の大小を比較することが可能です。

比較演算子は次のように使います。

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

SELECT * FROM area_code WHERE code <> '01202';
  name   | code
--------+-------
 七飯町 | 01337
 北斗市 | 01236
 鹿部町 | 01343
 森町   | 01345
(4 rows)
```

[1] 一般には，WHERE の 検索条件には真や偽（および不定）の値をとる式を記述することができます。「列名 比較演算子 値」はそのような式の代表例であり, 第 3 章の LIKE を使った式もその一つです。

```
SELECT * FROM area_code WHERE code > '01335';
  name   | code
--------+-------
 七飯町 | 01337
 鹿部町 | 01343
 森町   | 01345
(3 rows)
```

2.3.2 論理演算子を使った検索条件の結合

1. 複数の検索条件を論理演算子 「AND (かつ)」 や「OR (または)」で結ぶことができます[2]。これを SELECT 文の WHERE 句に用いれば，検索条件全体を真とする行のみが出力されます。

```
SELECT * FROM area_code
WHERE code = '01202' OR code = '01337';
  name   | code
--------+-------
 函館市 | 01202
 七飯町 | 01337
(2 rows)

SELECT * FROM area_code
WHERE code >= '01202' AND code < '01337';
  name   | code
--------+-------
 函館市 | 01202
 北斗市 | 01236
(2 rows)
```

2. 3 つ以上の検索条件を論理演算子で結ぶ場合，AND と OR の優先順位を考慮する必要があります。AND は OR より優先順位が高いので，必要に応じて ( ) を使います。

   次の二つの実行例の違いに注意してください。

```
SELECT * FROM area_code
WHERE code = '01202' OR code <= '01343' AND code >= '01337';
  name   | code
--------+-------
 函館市 | 01202
 七飯町 | 01337
 鹿部町 | 01343
(3 rows)
```

[2]論理演算子として，検索条件の真偽を反転する「NOT（否定；でない)」もあります。NOT は `NOT code = '01337'` のように検索条件の前に置いて使います。

```
SELECT * FROM area_code
WHERE (code = '01202' OR code <= '01343') AND code >= '01337';
  name  | code
--------+-------
 七飯町 | 01337
 鹿部町 | 01343
(2 rows)
```

なお，上記の例では全て同一列 (code 列) に関する検索条件を論理演算子で結びましたが，相異なる列に関する検索条件を AND や OR で結ぶことも可能です。

3 文字列パターンによる行の選択: LIKE

通常，LIKE は記号

- % 任意の文字列 (長さ 0 の文字列も含む)
- _ 任意の 1 文字

と共に使います。

3.1 任意の文字列: %

LIKE と % を使って，テーブル area_code から列 code に文字列 133 を含む行のみを取り出すには，次のようにします。

```
SELECT * FROM area_code WHERE code LIKE '%133%';
  name  | code
--------+-------
 七飯町 | 01337
(1 row)
```

% は任意の文字列を表す記号ですから，UNIX シェルのメタキャラクタにおける * に相当するものです。従って，

```
WHERE code LIKE '013%'
```

は列 code が 013 で始まる行を指定し，

```
WHERE code LIKE '%02'
```

は列 code が 02 で終わる行を指定しています。

3.2 任意の 1 文字: _

次に示すのは，任意の 1 文字を意味する _ の利用例です。_ はシェルのメタキャラクタ ? に相当します。最初の 4 文字は何でも構わなくて，5 文字目が 5 である行が取り出されます。

```
SELECT * FROM area_code WHERE code LIKE '____5';
 name | code
------+-------
 森町 | 01345
(1 row)
```

LIKE の後ろには % と _ の両方が含まれていても構いません。

4 行の並び順指定: ORDER BY

これまでの形式で SELECT 文を実行した場合，出力される行の順序は保証されません。行の順序を指定するには ORDER BY を使います[3]。

4.1 昇順指定

ORDER BY 句の最も単純な形式は次のとおりです。

```
ORDER BY 列名
```

この形式では, 出力行が指定した列名で昇順に並びます。昇順 (ascend) 並びであることを明示して

```
ORDER BY 列名 ASC
```

とすることもできます。

```
SELECT * FROM spring
ORDER BY area;
     name      | area
---------------+-------
 谷地頭温泉    | 01202
 湯の川温泉街  | 01202
 川汲温泉郷    | 01202
 戸井温泉      | 01202
 せせらぎ温泉  | 01236
 東大沼温泉郷  | 01337
 仁山温泉      | 01337
 鹿部温泉郷    | 01343
 濁川温泉郷    | 01345
(9 rows)
```

[3]関係データベースの行は，理論上は数学における集合の要素に対応します。集合では要素の並び順に意味はありませんので，理論上は関係データベースの行の並び順も無意味です。ただし，実用上はそれでは困りますので，ORDER BY で行の表示順序を指定できるようになっています。

4.2 逆順指定

行を降順 (descend) に出力するには

```
ORDER BY 列名 DESC
```

とします。

4.3 複数列による並び順指定

列名を ,（カンマ）区切りで複数指定して

```
ORDER BY 列名 1, 列名 2, ...
```

とすれば, 列名 1 の値が等しい行が, さらに列名 2 で並べ替えられます。各列名に続き ASC や DESC も指定できます。

```
SELECT * FROM spring
ORDER BY area DESC, name;
     name      | area
---------------+-------
 濁川温泉郷    | 01345
 鹿部温泉郷    | 01343
 仁山温泉      | 01337
 東大沼温泉郷  | 01337
 せせらぎ温泉  | 01236
 戸井温泉      | 01202
 川汲温泉郷    | 01202
 谷地頭温泉    | 01202
 湯の川温泉街  | 01202
(9 rows)
```

ここで，name 列の並べ替えは，日本語の文字コードに基づいて行われます。

4.4 WHERE 句を含む SELECT 文での ORDER BY

SELECT 文に「WHERE 検索条件」を含む場合，ORDER BY はその後ろに書く必要があります。

```
SELECT * FROM area_code
WHERE code = '01236'
  OR code = '01202'
  OR code = '01337'
ORDER BY code;
```

```
  name   | code
---------+-------
 函館市 | 01202
 北斗市 | 01236
 七飯町 | 01337
(3 rows)
```

5 演習問題

1. テーブル postcode （郵便番号簿）に含まれる，列 ken_kanji（県名）の内容のみをすべて表示しなさい。この結果が得られたら, 重複する行の表示を無くしなさい。

2. テーブル postcode から，郵便番号 (列名: new_post_code) が 0400054 である県名 (列名: ken_kana)，市名（列名:town_kana)，町名（列名: zone_kana) を表示しなさい。

3. テーブル postcode から，自治体コード（列名: pub_auth_code）が 01202 と 01204 である行のみを表示させなさい。ただし，表示する列は自治体コードと市の名前 (列名: town_kana) のみとする。重複する行の表示は省くこと。

4. テーブル postcode から，列 pub_auth_code の左から 3 つ目の数字が 1 である行のみを表示しなさい。ただし，表示する列は pub_auth_code，ken_kana, town_kana の 3 列のみとし，結果が重複する行は出力しないこととする。

5. 前の問題の出力を, 都道府県名 (ken_kana 列) で昇順に並べなさい。同じ都道府県名の行については, さらに市区名 (town_kana 列) で昇順に並べなさい。