情報科学演習 資料 9
データベースシステムへの問い合わせの基礎 — SQL と SELECT 文の基本

2020 年 7 月 13 日

目 次

1 SQL とは

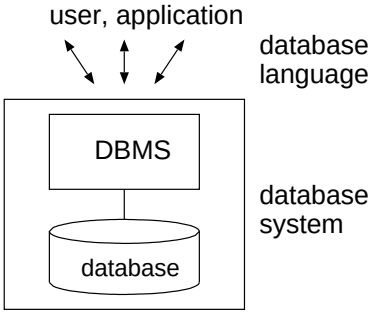
1.1 データベース

データベース 複数のユーザーやプログラムで共有されることを意図したデータの集まりをデータベース (database) といいます。

データベース管理システム (DBMS) データベースとその利用者 (ユーザーやアプリケーション) の仲介をするのがデータベース管理システム (database management system; DBMS) です。データベース管理システムはデータベースを一括して管理し，利用者からのデータの蓄積や検索，更新の要求に応え，その処理を効率良く確実に行います。

データベースシステム データベースとデータベース管理システムを合わせてデータベースシステム (database system) といいます。データベースという用語はデータベースシステムを指して使われることもあります。

データベース言語 利用者はデータベース言語 (database language) を利用して，データベース管理システムを通じてデータベースを操作します。データベース言語は，データベースの統一的な利用法を利用者に提供するものです。代表的なデータベース言語として SQL があります。



1.2 関係データベースと SQL

1.2.1 関係データベース

データベースシステムには幾つかの種類がありますが，この授業では，現在最も標準的に用いられている**関係データベース** *(relational database)* システムを扱います。

関係データベースシステムにおけるデータベースは**テーブル** (表; table) の集まりです。テーブルを関係 (relation) と呼ぶことがあります[1]。

テーブルは**行** *(row)* と**列** *(column)* で構成され，列には必ず名前があります。行をタプル (tupple)，列を属性 (attribute)，列の名前を属性名と呼ぶことがあります[2] 。

[1] 集合論での「関係」に由来します
[2] awk での用語に対応させれば，行はレコードであり，列はフィールドです。

1.2.2 SQL

SQL (structured query language) は関係データベースを扱うためのデータベース言語です。SQL では，C などの手続き型言語とは違い，原則として目的を達成するための手順 (how) ではなく，何をしたいか (what) を記述します。

この授業で主に扱う SQL は， ANSI (米国規格協会) と ISO (国際標準化機構) が 1992 年に制定した SQL2 規格 (通称 SQL-92) です。

SQL2 は 日本語の利用が可能になるよう，1995 年に JIS (日本工業規格) 規格化もされています。

1.3 テキストファイルとデータベースからの検索の実際

1.3.1 UNIX コマンドを使ったテキストファイル内検索

1. less コマンドを使って大きなテキストファイル (全国郵便番号データ; 約 12 万行) を表示して検索する

   ```
   less /pub/eis/postcode/ken_all
   ```

   (a) less の / コマンドで 01202 を検索する

   (b) / で 47201 を検索（01202 のときに比べ，ちょっとだけ時間がかかります）

   less は q をタイプして終了する。

2. grep コマンドで同じファイルを検索

   ```
   grep 47201 /pub/eis/postcode/ken_all |less
   ```

1.3.2 データベースシステムを使った検索例

テキストファイル ken_all と同等のデータを含むデータベースを用意しています。データベース管理システムを通じて，そこから検索を行ってみます。

1. psql コマンドを実行してデータベースに接続する

   `psql db_a` (データベース db_a に接続)

2. 検索: プロンプト `db_a=>` に対して次をタイプして `<ENTER>` を押す。

   ```
   select * from postcode where pub_auth_code = '47201';
   ```

   検索結果が一画面に収まらないために，less を通じて検索結果が表示されるので，`<SPACE>` を押して最後のページまで表示し終えて less を終了するか，`q` を押して検索結果の表示を終了する。操作の途中でわからなくなったら，CTRL-c を押してみるとよい。

3. psql を終了してデータベースシステムとの接続を絶ち，UNIX のシェルに戻る

   `\q` (psql の終了)

1.4 SQL 文の記述に関する注意

SQL 文

```
select * from postcode where new_post_code = '0400044';
```

は，テーブル postcode から, 列 (属性) new_post_code の値が 0400044 である行を表示するものです。この文における select や from，where は，SQL において特別の意味を持つ語であり，これらを SQL のキーワードと呼びます。

SQL 文の記述では，キーワードやテーブル名および属性名に大文字／小文字の区別はありませんので，どちらで書いても構いません。ただし，書籍等では文を見やすくするために，キーワードを大文字で表記し，テーブル名や属性名を小文字にするのが一般的です。この資料でも，これ以降は原則として，キーワードを大文字で記しますが，psql で実行する際には小文字でタイプして構いません。

一方，04000044 はデータベースからの検索の際に，検索条件として使う値 (文字) です。値の大文字/小文字は区別されます。また，これから当面は，値を SQL 文に記述する場合には，それを' (単一引用符 ; シングルクォート) で囲みます。ただし，値を単一引用符で囲むか否かは列の「データ型」に依存します。データ型については後の回で説明します。なお，値として ' 自身を表現したければ，'' とします。

語と語との区切りには連続する空白文字や改行を使うことができます。したがって，上記の SQL 文は

```
SELECT *
FROM postcode
WHERE new_post_code = '0400044';
```

と書くこともできますし，逆に複数行の SQL 文を一行で書いても構いません。

2 psql コマンドの使い方

この授業では，データベースシステムとして PostgreSQL というソフトウェアを利用します。psql は PostgreSQL に付属するデータベース操作のための対話型コマンドです[3]。この授業では psql コマンドを使って SQL 文を実行します。

2.1 起動方法

psql の基本的な起動の仕方は

```
psql データベース名
```

です。

psql コマンドの引数には，オプションや接続するデータベース名を指定しますが，実際に何を指定するかについては，授業での指示に従ってください。なお，psql コマンドの一般的な実行の書式は `psql --help` でわかります。

[3] psql は標準的な UNIX コマンドではないので，PostgreSQL がインストールされていないシステムでは使えません。

2.2 起動後の操作法

1. psql を起動すると，次の表示が表れます。

```
psql (8.4.20)
"help" でヘルプを表示します.

db_a=>
```

   最後の db_a=> は psql におけるプロンプトです。これに続き SQL 文などをタイプし <ENTER> を押して実行します。プロンプトにおける db_a は接続中のデータベース名です。

2. psql の終了は \q です。

3. SQL の命令は文を単位として行います。psql を利用している場合，SQL 文 (\ 以外で始まる命令) はセミコロン (;) で終わります。

   セミコロンを打たずに <ENTER> を押すと，SQL は実行されずに db_a-> のようなプロンプトが現れます (=> ではなく -> である点に注意) 。SQL 文では，改行は無視されますので，続いてセミコロンを打ってから<ENTER> を押せば，元の SQL を実行できます。

   SQL を実行せずに，元のプロンプト (=>) に戻すには，CTRL-c を打ちます。

4. SQL 文の入力時に tcsh のコマンドラインとほぼ同様の履歴 (ヒストリ) 機能や補完機能が使えます。なお，補完は SQL キーワードやテーブル名，列名などが対象ですが，補完が効かない場合もあります[4]。

2.3 練習

psql コマンドでデータベース db_a に接続し，以下の SQL 文を実行してください。

1. 次の SQL 文による検索（問い合わせ）を行ってください。すべて小文字でタイプして構いません。

```
SELECT * FROM postcode WHERE pub_auth_code = '01202';
```

2. 次の問い合わせを実行してください。ただし，psql の履歴（ヒストリ）機能を活用してください。

```
SELECT * FROM postcode WHERE pub_auth_code = '13101';
```

3. 再度，同じ問い合わせを実行します。ただし，今回は 3 行に分けてタイプしてください。

```
select * from postcode
where pub_auth_code = '13101'
;
```

4. 次の SQL 文による問い合わせを実行して，郵便番号から住所を検索してください。

[4]例えば，SELECT の直後の列名は補完できません。テーブル名を入力する前では，列名の候補が定まらないからです。

```
SELECT * FROM postcode WHERE new_post_code = '0400044';
```

5. 次の SQL 文も試してください。

```
SELECT * FROM postcode WHERE new_post_code = '0400000';
```

6. 自分の家の郵便番号などを使って検索してみてください。

3 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。

3.1 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。

3.1.1 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)
```

3.1.2 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;
```

3.2 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。

3.3 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[5]。

3.3.1 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)
```

[5] 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 4 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)
```

3.3.2 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[6]。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)
```

[6]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。

4 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。

4.1 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。

4.2 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。

5 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。重複する表示は無くすこと。
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。