データベース 演習資料

第1回 PostgreSQL によるデータベース実践演習

九州工業大学 情報工学部
講義担当：尾下真樹

1. 演習環境

現在、リレーショナルデータベースシステムとして、商用のものからフリーのものまで、多くのシステムが利用可能である。本演習では、自由に利用可能なシステムとして広く使われている、PostgreSQL（ぽすとぐれす、ぽすとぐれすきゅーえる、などと読む）を使用する。
PostgreSQL はサーバクライアント型のシステムである。サーバとなるコンピュータでは、PostgreSQL のプログラムが常に動いており、外部からコマンドが送られてくるのを待っている。データベースのデータは全てクライアント側ではなくサーバ側のハードディスクに記録される。クライアントのコンピュータから、サーバにさまざまなコマンドを送ることで、データベースの操作を実行できる。
本演習では、**db.tom.ai.kyutech.ac.jp** という名前のコンピュータを、PostgreSQL データベースサーバとして使用する。なお、データベースサーバには、学内のコンピュータからしか接続することができない設定となっている。そのため、データベースサーバを利用する場合は、飯塚キャンパスのネットワークから接続するか、大学外から飯塚キャンパスの VPN を経由して接続する必要がある。
演習を行うクライアント側のコンピュータとしては、BYOD 端末の Ubuntu 仮想環境を用いる。

1.1. BYOD 端末の仮想環境（Ubuntu）での演習

情報基盤センターから配布されている VirtualBox 用または wsl 用の Ubuntu 仮想環境を使用することで、各自の BYD 端末をクライアント側のコンピュータとして使用して、演習を行うことができる。
配布されている Ubuntu 仮想環境には、あらかじめ PostgreSQL クライアントのソフトウェアが導入されているため、特に追加のソフトウェアをインストールすることなく、演習を行うことができる。
デスクトップのメニューバーに配置されているアイコンからターミナル（端末）を起動して、ターミナルからコマンドを入力することで、PostgreSQL のプログラムを実行できる。

BYOD 端末を使って演習を行う前に、以下の準備を行うこと。

1. 情報基盤センターから配布されている VirtualBox 用または wsl 用の Ubuntu 仮想環境を利用できるようにする。
2. Moodle 上で配布する各自の PostgreSQL ユーザのパスワードを取得する。
3. 飯塚キャンパス外から演習を行うときには、飯塚キャンパスの VPN サーバに接続する。
   戸畑キャンパスではなく、飯塚キャンパスの VPN サーバに接続していることを確認する（図 1）。

上記 1･3 の設定方法については、情報基盤センターの ISC オンラインガイド（http://onlineguide.isc.kyutech.ac.jp/）を参照すること。また、これらの設定に関して問題や不明な点があれば、情報基盤センターの相談窓口で質問すること。

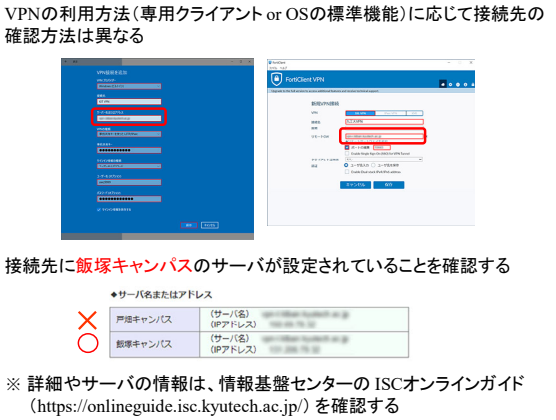

図 1 VPN の接続先の設定の確認

2. データベースの作成と psql の起動

最初に、テーブルやデータを記録するためのデータベースを作成する必要がある。
ターミナル（端末）から createdb というコマンドを使用してデータベースを作成する。createdb を実行するとデータベースサーバ上にデータベースが作成される。
createdb の使い方は下記の通り。

createdb [-h *hostname*] [-U *username*] [*dbname*]
PostgreSQL サーバ上にデータベースを作成する。 オプション ***hostname*** には、サーバとなるコンピュータの IP アドレスもしくは名前を指定する。 オプション ***username*** には、PostgreSQL ユーザ名を指定する。 オプション ***dbname*** には、作成するデータベース名を指定する。

データベースを作成したら、ターミナルから psql というプログラムを起動してサーバに接続する。psql は対話型のプログラムで、psql 上でコマンドを入力することで、データベースに対してさまざまな操作を行うことができる。

psql [-h *hostname*] [-U *username*] [*dbname*]
PostgreSQL サーバに接続する。 オプション ***dbname*** にの役割は、上記の createdb と同じ。 データベース名を省略すると、ユーザ名と同じ名前のデータベースに接続する。

ここで、実際にデータベースを作成して psql を起動してみる。下記のように、ターミナルから createdb と psql を実行する。
本演習では、履修者の九工大 ID と同じ名前で PostgreSQL ユーザが作成されているので、PostgrSQL ユーザ名（*username*）には、各自の九工大 ID を記述する。
データベース名（*dbname*）は、他の利用者と重複しない名前を指定する必要がある。本演習では、各自の PosgreSQL ユーザ名をデータベース名として使用することにしているので、データベース名（*dbname*）にも、自分の九工大 ID 名を記述する。（*username* と *dbname* の両方に九工大 ID を記述する。）
コマンドを実行後、パスワードの入力が求められたら、Moodle 上で配布する各自の PostgreSQL ユーザのパスワードを入力する。（九工大 ID のパスワードとは異なるので、注意する。）

下記の実行例で、行頭の $ は入力受付記号を表す。環境によっては、$ の前にローカルユーザ名・OS・カレントディレクトリなどの情報が表示されるが、以下の実行例では省略している。また、表示される psql のバージョン等は、環境によって変化する可能性がある。
なお、パスワード入力時は、他の多くのプログラムと同様、セキュリティのため、キー入力の内容は画面に表示されないようになっているが、入力は行われているので、パスワードとエンターキーを入力する。

```
$ createdb -h db.tom.ai.kyutech.ac.jp -U username dbname

$ psql -h db.tom.ai.kyutech.ac.jp -U username dbname
Password for user username: ??????
psql (12.5 (Ubuntu xxx)  server 10.17)
Type "help" for help.

dbname=>
```

psql を起動すると上記のように入力待ちの画面になる。この状態で、SQL や psql 専用のコマンドを入力することで、さまざまな操作を実行できる。
なお、createdb によるデータベース作成は最初に一度だけ行えば良いので、二回目以降にデータベースを利用するときには、createdb を行う必要はなく、psql を起動するだけで利用できる。
もし、作成したデータベースを削除して作り直したい場合は、下記のように dropdb プログラムを使用することで削除できる。このとき、作成したテーブルやデータは全て削除されるので、注意して実行すること。

```
dropdb -h db.tom.ai.kyutech.ac.jp -U username dbname
```

先述の通り、データベースサーバに接続するためには、飯塚キャンパスのネットワークから接続するか、大学外から飯塚キャンパスの VPN を経由して接続する必要がある。大学外から直接接続しようとすると、createdb や psql のプログラムを起動しても、データベースサーバに接続できず、処理が進まない状態になる。そのような場合は、プログラムを一度終了して（Ctrl＋C キー）から、VPN 接続を行ってやり直すこと。

3. psql の基本コマンド

psql には、SQL とは別に基本的な内部コマンドが用意されている。内部コマンドは、\ で始まる（環境によっては、\ は ¥ と表示されるので注意）。以下に、主なコマンドを紹介する。

コマンド	機能
¥q	psql を終了。
¥l	データベース一覧の表示。
¥dt	テーブル一覧の表示。
¥i *filename*	オプション *filename* で指定したファイルを読み込み、そのコマンドを実行する。
¥copy	ファイルとテーブルの間でデータを読み書きする。詳しい使い方は後述。
¥o *filename*	SQL の実行結果を画面に表示する代わりにファイルに出力する。
¥?	psql で使用可能なコマンドの一覧を表示。

psql が起動した状態で、¥l と入力してリターンキーを押すと、他の人が作成したデータベースも含めて、サーバ内にある全てのデータベースの一覧が表示される。
また、¥dt と入力すると、データベース内に作成されているテーブルの一覧が表示される。データベースを作成してすぐの状態では、テーブルは存在しないので、何も表示されない。

4. テーブルの作成

SQL の CREATE TABLE 文を実行することで新しいテーブルを作成できる。

create table *tname* (*atribute1 type1* [*constraints1*], …, [*other_onstraints*]);
新しくテーブルを作成する。 *tname* には、テーブルの名前を指定する。 *atribute* と *type* には、それぞれ属性名と属性の型を指定する。属性と型の組み合わせは、属性の数だけいくつでも記述できる。 *constraints1* には、属性についての制約を記述できる。空値を許さない属性については、型名の後に not null を付け加える。not null 指定された属性については、その属性が空値であるようなデータの挿入ができなくなる。同じく、unique 制約を加えると、既存のデータと同一の属性値を持つデータは、挿入できなくなる。primary key 制約を加えると、その属性がテーブルの主キーになる。foreign key 制約を加えると、その属性が外部キーとなる。外部キー（参照整合性制約）については、後述する。 また、*other_onstraints* には、テーブルや属性に関する制約を記述する。制約は、制約の種類（属性, …）の形式で記述する。括弧内に複数の属性を記述すると、複数の属性に関する制約となる。一つの属性のみを記述すると、その属性に関する制約となる。一つの属性に関する制約は、属性の直後に記述しても、最後に分けて記述しても、どちらでも構わない。複数の属性に関する制約は、最後に分けて記述することになる。 その他、属性値の範囲などの一貫性制約も指定できるが、本資料では詳しい説明は省略する。

属性の型としては、主に以下のものが指定できる。

分類	属性	意味
数値型	int2	整数（2 バイト、-32378～+32767）
	int / int4	整数（4 バイト、±約 21 億）
	int8	整数（8 バイト、±約 18 桁）
	real / float4	浮動小数点表現による実数（4 バイト）
	float8	浮動小数点表現による実数（8 バイト）
文字列型	text	可変長文字列（長さ制限なし）（PostgreSQL 独自）
	varchar(n)	可変長文字列（最大の n 文字）
	char(n)	固定長文字列（常に n 文字、自動的に空白が追加される）
日付時刻型	date	日付
	time	時間
	timestamp	日付と時間
	interval	日付時間の間隔
他	boolean	真偽値（TRUE, FALSE, NULL のどれかをとる）

例えば、職員（職員番号、部門番号、氏名、年齢）のようなテーブルを作成する場合は、psql のコマンドラインから以下のように入力する。

```
dbname=# create table employee( id varchar(4) not null unique, dept_no varchar(2), name varchar(12) not null,age int2, primary key( id ) );
```

行の最後にセミコロン（;）をつけるのを忘れないこと。psql は、セミコロンまでをひとつのコマンドとして認識する。
別のやり方として、あらかじめコマンドを記述したテキストファイルを作成しておくことで、psql からそのテキストファイルを読み込んでコマンドを実行することができる。コマンドラインからでは、長いコマンドを入力するのが面倒なことと、もし１つでも間違えてエラーが出たらまた最初から入力しなおさないといけないので、複雑なコマンドを実行する場合は、テキストファイルを作成してから実行した方が良い。
例えば、テキストエディタを起動し、以下のように入力して、employee_table.txt という名前で保存する。

employee_table.txt

```
create table employee(
    id varchar(4) not null unique,
    dept_no varchar(2),
    name varchar(12) not null,
    age int2,
    primary key( id )
);
```

ファイルからコマンドを読み込んで実行するためには、上記のコマンド一覧にある ¥i コマンドを使用する。このとき、読み込むファイルは、psql を起動した時点でのカレントディレクトリに存在する必要があることに注意する。

```
dbname =# ¥i employee_table.txt
```

作成したテーブルは、drop table コマンドを使用して削除できる。テーブルを削除すると、テーブルに入力した全てのデータも同時に削除されるので注意すること。

drop table *tname*;
既存のテーブルを削除する。*tname* には、テーブルの名前を指定。

なお、一度作成したテーブルの名前や属性名は、alter コマンドを使用して変更できる。

alter table *old_name* rename to *new_name*;
テーブルの名前を変更する。*old_name* には変更前の名前、*new_name* には変更後の名前を指定する。

alter table *tname* rename column *old_name* to *new_name*;
テーブルの属性（列）の名前を変更する。*tname* にはテーブルの名前を指定。*old_name* には変更前の属性の名前、*new_name* には変更後の属性の名前を指定する。

その他、テーブルに列や制約を追加することもできるが、本資料では詳しい説明は省略する。

5. データの追加

テーブルへのデータ（行）の挿入は、SQL の INSERT 文を実行することで行える。

insert into *tname*(*atribute1*, …, *atributeN*) values(*value1*, …, *valueN*);
テーブルに新しいデータ（行）を追加する。 *tname* には、テーブルの名前を指定する。*atribute1*～*atributeN* には属性名（列）を、*value1*～*valueN* にはそれぞれの属性値を指定する。not null 指定されていない属性は省略できる。

以下に、具体例を示す。

```
dbname =# insert into employee( id, dept_no, name, age ) values( '0001', '01', 'taro',20 );
INSERT 25177 1
```

ここで、INSERT の後ろに表示されるメッセージは、OID が 25177 のデータが 1 つ追加された、ということを意味する。PostgreSQL では、全てのデータ（行）がユニークな OID を内部に持つ。通常は、OID は気にする必要はない。
入力されたデータを確認するためには、下記のような、テーブルの全データの全属性値を表示させる SQL を実行してみると良い。

```
dbname =# select * from employee;
```

沢山のデータを挿入するとき、各データごとにいちいち INSERT 文を使って記述するのは面倒である。そこで、別の方法として、psql の copy コマンドを使用すればテキストファイルから一度に複数のデータを挿入できる。

employee_data.txt

```
0001,01,織田 信長,48
0002,02,豊臣 秀吉,45
0003,03,徳川 家康,39
0004,01,柴田 勝家,60
0005,02,伊達 政宗,15
0006,03,上杉 景勝,26
0007,01,島津 家久,35
```

例えば、上のようなテキストファイルを作成しておき、copy コマンドを実行すれば、テキストファイルのデータをテーブルにまとめて格納できる。

```
dbname =# ¥COPY employee FROM 'employee_data.txt' USING DELIMITERS ','
```

逆に、copy コマンドを利用して、下記のように、テーブルの全データをファイルに書き出すこともできる。

```
dbname =# ¥COPY employee TO 'employee_data_output.txt' USING DELIMITERS ','
```

6. SQL による問い合わせ

SQL を使って問い合わせを記述することで、データを検索して表示させることができる。
SQL については講義で詳しく学習しているので、ここでは書き方の説明は省略して、具体例のみ示す。
上記の処理を行って employee にデータが挿入された状態で、下記の問い合わせを実行して確認してみよ。

「40 歳以下の従業員の氏名の一覧」

```
dbname =# select name from employee where age < 41;
```

「部門番号 01 の従業員の人数」

```
dbname =# select count(*) from employee where dept_no = '01';
```

7. データの更新と削除

SQL の DELETE 構文や UPDATA 構文を使用することで、データの削除や変更もできる。

「従業員番号 0001 の従業員を削除」

```
dbname =# delete from employee where id = '0001';
```

「従業員番号 0002 の従業員の年齢を 20 に変更」

```
dbname =# update employee set age = 20 where id = '0002';
```

8. 複数のテーブルの作成

同様に、別のテーブルとして、部門（部門番号, 部門名）のテーブルを作成してみる。

department.txt

```
create table department(
        dept_no varchar(2) not null unique,
        name varchar(12) not null,
        primary key( dept_no )
);
insert into department values( '01', '開発' );
insert into department values( '02', '営業' );
insert into department values( '03', '総務' );
```

上記のように、テーブル作成とデータ挿入の SQL をまとめてファイルに記述して、一度に実行することもできる。

```
dbname =# ¥i department.txt
```

これで、部門のテーブルが作成された。
次に、2 つのテーブルを使って、従業員の部門番号から、部門の部門番号へ、外部参照整合性制約を追加してみる。外部参照整合性制約を追加することで、部門のテーブルにない部門番号を、従業員の部門番号の属性値として使えなくなり、不整合なデータが格納される可能性を減らすことができる。
上で紹介した alter table コマンドを使って、テーブルに制約を追加することができる。

alter table *teble_name* constraint *constraint_name constraint*;
テーブルに制約を追加する。主キー制約、Unique 制約、NUL 制約、外部参照整合性制約、などを追加することができる。*constraint_name* は、適当な制約の名前を設定する。制約名は、後で制約のみを削除するときのためのものである。*constraint* には、制約の内容を記述する。

外部参照整合性制約を追加するときには、制約の内容として、下記のように、外部キーとなる属性と、参照先のテーブル・属性名を指定する。

```
dbname =# alter table employee add constraint employee_dept_key foreign key (dept_no)
references department (dept_no);
```

上の制約を追加した後で、以下のようなデータ（部門番号が 04 のデータ）をテーブルに追加するコマンドを実行してみて、データの追加ができないことを確認せよ。

```
dbname =# insert into employee values( '0020', '04', 'Jack', 20 );
```

付録 1. テキストファイルの文字コードを変換する方法

日本語の文字コードとして、Windows ではシフト JIS を使用し、Linux では EUC が使用されている。最近は、どちらの環境もユニコードに対応しているが、環境やソフトウェアによっては、シフト JIS や EUC が使われることも多い。
そのため、異なる文字コードで作成したファイルを別の環境に持って行くと、日本語が文字化けする。
また、Windows と Linux では、改行コードも異なる。
Linux にインストールされている nkf コマンドなどを使うことで、文字コードや改行コードを変換できる。
詳しい使い方は、nkf --help を実行し、表示される使い方の説明を参照すること。

テキストファイル(result.txt)を Windows 環境用のシフト JIS 形式のファイル(result_sjis.txt)に変換

```
nkf -s -Lw result.txt > result_sjis.txt
```

テキストファイル(result.txt)を Linux 環境用のユニコード形式のファイル(result_utf.txt)に変換

```
nkf -u -Lu result.txt > result_sjis.txt
```

テキストファイル(result.txt)を Linux 環境用の EUC 形式のファイル(result_euc.txt)に変換

```
nkf -e -Lu result_sjis.txt > result_euc.txt
```

テキストファイル（result.txt）の文字コードを判別して表示する

```
nkf --guess result_euc.txt
```

付録 2. PostgreSQL のエラーへの対処

createdb や psql などのプログラムを実行したときに表示されるエラーと対処方法は、下記の通り。

psql: error: could not connect to server: 接続がタイムアウトしました
飯塚キャンパスのネットワークや、飯塚キャンパスの VPN サーバに接続されていない。
学外から演習を行うときには、飯塚キャンパスの VPN サーバに接続する必要がある。

VPN 接続の設定（確認）方法は、1.1 節（図 1）の説明を参照すること。
自分がどの VPN サーバに接続しているかを把握しておらず、飯塚キャンパスの VPN サーバに接続しているつもりで、実際には戸畑キャンパスの VPN サーバに接続しており、正しく演習を行えない場合が多くある。
このような問題が生じた場合は、飯塚キャンパスのネットワークに接続した状態で演習を行うか、上記の説明を参照して、飯塚キャンパスの VPN サーバに接続していることを確認すること。

psql: error: FATAL: password authentication failed for user "????????"
ユーザ名、または、パスワードが間違っている。指定された通りのユーザ名・パスワード（九工大 ID と同一のユーザ名＋Moodle で配布されるパスワード）を入力しているか、確認する。

指定された期限までに履修登録と九工大 ID の申告が行われていなければ、ユーザの作成とパスワードの配布は行われないため、データベースサーバは利用できない。正しいユーザ名・パスワードを入力してもエラーが表示される場合は、九工大 ID の申告が間違っていたなどの理由で正しいユーザが作成されていない可能性があるため、授業担当教員宛に電子メールで、学生番号・氏名・九工大 ID の情報を明記した上で連絡すること。

psql: error: too many command-line arguments (first is "????????")

psql: warning: extra command-line argument "????????" ignored
psql のオプションが正しく入力されていない。本資料の通りにコマンドを入力していることを確認する。
特に、－（マイナス）記号やスペースなどの文字が、正しく半角文字で入力されていることを確認する。
資料の文字列をコピーして貼り付けると、フォント等の関係で正しい文字が入力されない場合があるため、必ず、キーボードから入力を行うこと。

付録 3. SQL のエラーへの対処

SQL の書き方を間違えると、エラーメッセージが出力されて、SQL が実行されない。
エラーが発生したときには、SQL 中のどこでエラーが発生したかという情報と、エラーが発生した理由が表示されるので、これらをよく読んで、エラーを修正する。
よく出るエラーと対処方法としては、下記のようなものがある。

ERROR: syntax error at or near “???”

SQL の入力が間違っている。どの単語の前後で間違いが生じているかが表示されているので、その情報を参考に間違いを修正する。

前の入力文の最後に ； を付けていない場合、前の入力文と合わせて実行しようとしてエラーが生じている可能性があるので、その場合は、； を入力して前の入力文を終了させてから、SQL を入力する。

ERROR: ??? does not exist

SQL で指定したテーブルや属性名が存在していない。正しいテーブル名や属性名を入力していることを確認する。

ERROR: relation “???” already exists

作成しようとしたテーブルと同じ名前のテーブルはすでに作成されているため、作成できない。テーブルの名前を別の名前にするか、既存のテーブルを削除してから作成する。

ERROR: duplicate key value violates unique constraint "???"

主キーの値が重複するデータをテーブルに挿入することはできない。主キーを別の値にして挿入するか、値が重複する既存のデータを削除してから挿入する。

ERROR: value too long for type character varying (???)

文字列型の属性に格納しようとしている文字列が、テーブル作成時に指定した文字数（最大文字数）を超えているため、格納できない。

ERROR: column reference “???” is ambiguous

??? という名前の属性が複数のテーブルにあるので、どのテーブルの属性を使うのか決定できない。
テーブル名.属性名 のように、属性の前にテーブル名を明記する。

ERROR: column ??? must appear in the GROUP BY clause or be used in an aggregate function

GROUP BY を使うときには、SELECT 節に記述する属性は、GROUP BY に使った属性か、集約関数でなければならない。

ERROR: No operator matches the given name and argument type(s). You might need to add explicit type casts.

属性に対して代入や比較や演算を行おうとしている値の型が異なっている。例えば、文字列型の属性の値に対して数値と比較を行っているなど。代入や比較や演算を行うときには、属性や値の型を合わせる必要がある。