データベースＳ 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 システム創成情報工学科
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、サーバにいろいろなコマンドを送り、データベースの操作を実行することができる。サーバとクライアントは一般には別々のコンピュータであるが、同一のコンピュータであっても構わない。

本演習では、Computing Laboratory（学科端末室）の各パソコンがクライアントになる。サーバとしては、学科で管理されている popuradb.ces.kyutech.ac.jp という名前のコンピュータを使用する。
本演習では、クライアントの環境として Windows（の Cygwin 環境）を使用する。
また、サーバには学科内のコンピュータからしかアクセスできないように制限がかけられているので、自宅のパソコンに PostgreSQL のクライアントをインストールしても、そこから学科のサーバにアクセスすることはできない。もし、自宅で作業したい場合は、自宅のパソコンで PostgreSQL サーバも稼動させて、そこにデータベースを作成すること。

2. PostgreSQL のインストール

CL のパソコンには、すでに PostgreSQL がインストールされているので、特に作業は必要ない。

もし、自宅のパソコン（PC-UNIX や Windows）にインストールして使いたい場合は、適当な参考書やウェブサイトなどを参照すること。PostgreSQL はユーザも多く、参考書は豊富にあるので、それほど難しくはない。最近は、Windows 版の PostgreSQL も公開されており、Windows にインストールして使用することもできる。

3. 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。
Cygwin のターミナルから 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 [*dbname*] [-h *hostname*] [-U *username*] [-E *char_code*]
PostgreSQL サーバ上にデータベースを作成する。 オプション *dbname* には、データベース名を指定する。省略すると、ユーザ名と同じ名前のデータベースが作成される。 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 *char_code* には、データベースで使用する日本語文字コードを指定する。本演習では、ユニコード（UTF8）を使用する。

データベースを作成したら、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いろいろな操作を行うことができる。

psql [*dbname*] [-h *hostname*] [-U *username*]
PostgreSQL サーバに接続する。 オプション *dbname* には、これから作業を行うデータベース名を指定する。省略すると、ユーザ名と同じ名前のデータベースが使用される。指定したデータベースが存在しなければ、エラーが出る。 オプション *hostname* 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。

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、あらかじめ履修者全員が PostgreSQL ユーザとして登録されているので、PostgrSQL ユーザ名（*username*）には自分に割り当てられたユーザ名を使用する。各自の九工大アカウント名（端末にログインするときに使用するアカウント名）を、そのまま PostgreSQL ユーザ名として使用する。
また、データベース名（*dbname*）も、他の人と重複しないように、必ず、自分の PostgreSQL ユーザ名と同じものを指定すること。本授業の演習では、*username* ＝ *dbname* とする。

```
username@pcXX ~
# createdb dbname -h popuradb.ces.kyutech.ac.jp -U username -E UTF8
CREATE DATABASE
# psql dbname -h popuradb.ces.kyutech.ac.jp -U username
Welcome to psql 7.3.2, the PostgreSQL interactive terminal.

Type:  copyright for distribution terms
       ¥h for help with SQL commands
       ¥? for help on internal slash commands
       ¥g or terminate with semicolon to execute query
       ¥q to quit

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たい場合は、dropdb プログラムを使用することで削除できる。詳細は末尾の付録を参照。

4. 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。
¥?	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。

5. 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	data	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で示したように、¥i コマンドを

使用する。

```
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。

6. 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‘, ‘007‘, ‘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表に格納することができる。

```
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 ','
```

7. 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。
上記の処理を行いテーブル 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、結果を確認してみよ。

「30 歳以下の従業員の氏名の一覧」

```
dbname =# select name from employee where age < 31;
```

「部門番号 01 の従業員の人数」

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

8. 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 の年齢を 20 に変更」

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

9. 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. psql の実行結果のファイルへの出力方法

Unix シェルのリダイレクションの仕組みを使用することで、psql の実行結果をファイルに書き出したり、psql に行わせる処理をファイルから読み込んだりすることができる。

例えば、以下のようにして psql を起動すると、psql の出力メッセージは、画面ではなく、ファイル result.txt に書き出される。

```
psql dbname -h popuradb.ces.kyutech.ac.jp > result.txt
```

同じように、以下のようにして psql を起動すると、キーボードから入力する文字列の代わりに、command.txt の内容が読み込まれて実行される。

```
psql dbname -h popuradb.ces.kyutech.ac.jp < command.txt
```

上の２つを組み合わせて、下記のように書くこともできる。

```
psql dbname -h popuradb.ces.kyutech.ac.jp < command.txt > result.txt
```

付録 2. 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（CL）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
```

付録 3. CL 端末の使い方

Windows の基本的な使い方については、本授業の範囲外であるため、リテラシーの講義で習った内容を復習したり、基本的なことは自分で調べたりするなどして、解決すること。
どうしても分からないことがあれば、個別に質問に来れば、出来る限り対応する。

Q. テキストファイルを編集しようとすると、文字化けしたり、改行が正しく表示されなかったりします。
A. Windows 標準のメモ帳は、Unix 系の文字コード・改行コードにきちんと対応していないので、テキストファイルの編集には、これらに対応した別のテキストエディタを使う。CL 端末には、TeraPad がインストールされている（デスクトップにショートカットが置かれている）。

Q. createdb や psql などのプログラムが、コマンドプロンプトから実行できません。
A. これらのプログラムは Cygwin 上で動作するものなので、Windows のコマンドプロンプトからは起動できない。デスクトップに Cygwin のショートカットが置かれているので、これをダブルクリックして Cygwin のターミナルを起動し、ここから createdb や psql などのプログラムを実行する。

Q. Cygwin のターミナルにコピー・ペーストができません。
A. Cygwin のウィンドウの左上のアイコンをクリックすると、メニューが表示されるので、ここからコピー・ペースト等の操作を行うことができる。

Q. ¥i などのコマンドを使うと、ファイルが存在しないというエラーが出ます。
A. Cygwin のターミナルを起動した状態では、カレントディレクトリはユーザのホームディレクトリ（Z ドライブ）となっている。ファイルを読み書きするためには、ファイルを Z ドライブに置くか、適当なサブディレクトリに置いた上で、cd コマンドを使用してそのディレクトリまで移動する。

Q. ログインできません／システムがおかしくなりました。
A. システムやアカウントのトラブルについては、学科技術職員室（E526、研究棟 5 階エレベータ裏の部屋）にいる技術職員の人に申し出ること。教員には CL のシステム管理者の権限がないので、教員のところに相談に来ても、システムのトラブルについては対応できない。必ず、再起動してみたり、一通り自分で調べたりした上で、システムのトラブルであると思われるときにのみ、相談に行くこと。

付録 4. 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。
よく出るエラーとしては、下記のようなものがある。

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、テーブル名の情報を追加する。

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。

付録 5. データベースの削除方法

作成したデータベースを削除して最初から作り直したい場合は、dropdb コマンドを使うと削除できる。

```
dropdb dbname -h popuradb.ces.kyutech.ac.jp -U username
```