はじめに（対象者について）

本教材は、PostgreSQL の利用を検討している方や、これから PostgreSQL をはじめて学習する方が、短期間で PostgreSQL のポイントを習得することを目的としています。そのため、1 つ 1 つの機能詳細を深く掘り下げるのではなく、実習を中心に浅く広く学習していただくように構成しました。およそ 3 時間から 5 時間で、一通りの学習を終えることができます。
本教材の学習後には、公開されているマニュアルなどを活用して、さらに理解を深めていってください。本教材を終えた後であれば、最初は少し難しく感じたかも知れないマニュアルも、はるかに内容が理解できるようになっているはずです。
本教材が、PostgreSQL を利用するうえでのファーストステップになれば幸いです。

前提必要知識

本教材の学習は、以下の前提知識が必要です。※基本が理解できていれば問題ございません。

- RDBMS の基本機能を理解している（基本的な SQL、トランザクション、ロック、など）。
  ※本教材は、一般的な「データベースの基礎」を学習するためのものではございません。
- 基本的な Linux コマンドを理解している（ls、vi エディタの使用方法、など）。

表記について

参照 ：参照ページ

Tips ：補足事項。知っておくと便利な知識を紹介しています。

本教材での PostgreSQL 使用バージョン

本教材は、PostgreSQL バージョン 9.4 で作成しています。また、例題もすべて同バージョンでの内容になっています。

PostgreSQL のマニュアル

PostgreSQL のマニュアルは以下のサイトからダウンロードできます。
本教材で PostgreSQL の重要ポイントを幅広く学習したら、今度はマニュアルを使って理解を更に深めましょう。

▼日本 PostgreSQL ユーザ会
URL：http://www.postgresql.jp/document/
※用語や機能などを調査したい場合、「目次」ページの検索が便利です。

CONTENTS

実習環境の準備

１章 ：PostgreSQL 概要

２章 ：PostgreSQL の基本を理解する

３章 ：データベースを管理する

４章 :バックアップ・リカバリを体験する

付録

３.データベースの作成

PostgreSQL では、１つのデータベースクラスタに複数のデータベースを作成できます。

１.データベースを作成する

データベースは、create database 文によって作成します。

■例題３－３： create database 文によって、データベースを作成する。

create database 文を使ったデータベースの作成、作成したデータベースの確認、接続を行います。

※例題 3-2 で postgres プロセスを停止している場合、起動してから以下の例題を行ってください。

①psql を起動します。

```
[postgres@post-srv ~]$ psql
psql (9.4.0)
Type "help" for help.

postgres=#
```

※データベースやユーザ名を指定しないと、postgres データベースに PostgreSQL のスーパユーザで接続されます。

②メタコマンド「\l」で、既に作成されているデータベースを確認できます。

※postgres データベースは、initdb コマンドでデータベースクラスタを作成すると自動的に作成される、デフォルトデータベースです。それ以外に、新規データベースのテンプレートとして template0、template1 データベースが存在します。

```
postgres=# \l
                                List of databases
   Name    |  Owner   | Encoding | Collate | Ctype |   Access privileges
-----------+----------+----------+---------+-------+-----------------------
 postgres  | postgres | UTF8     | C       | C     |
 template0 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
 template1 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
(3 rows)
```

③この例では、create database 文で「postdb」データベースの作成を行います。

※createdb コマンドを実行すると、内部的に postgres データベースに接続し、create database 文を実行してデータベースを作成しています。

```
postgres=# create database postdb;
CREATE DATABASE
```

④メタコマンド「\l」で、postdb データベースが作成されたことを確認します。

```
postgres=# \l
                                 List of databases
   Name    |  Owner   | Encoding | Collate | Ctype |   Access privileges
-----------+----------+----------+---------+-------+-----------------------
 postdb    | postgres | UTF8     | C       | C     |
 postgres  | postgres | UTF8     | C       | C     |
 template0 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
 template1 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
(4 rows)
```

⑤メタコマンド「\c」で、作成した postdb データベースに接続します。接続先の postdb データベースを指定します。

```
postgres=# \c postdb
You are now connected to database "postdb" as user "postgres".
```

※ユーザ名を指定していないため、スーパユーザ（postgres）でログインしています。

⑥接続後、接続しているデータベースを確認します。

```
postdb=# select current_database();
 current_database
------------------
 postdb
(1 row)
```

４.パラメータの設定

PostgreSQL には、データベースの動作に影響を与える数多くのパラメータが存在します。
この項目では、パラメータの設定方法や設定されている値の確認方法を解説します。

１.postgresql.conf ファイル

postgresql.conf ファイルは、PostgreSQL のパラメータ設定ファイルです。
このファイルは、デフォルトではデータベースクラスタのルートディレクトリ（環境変数 PGDATA）に配置されています。

■例題３－４： postgresql.conf ファイルでパラメータを変更する
postgresql.conf ファイル内のパラメータを１つ変更し、変更が反映されていることを確認します。
※例題 3-3 終了後、データベースに接続したままの場合は、psql を「\q」で終了してから例題を行ってください。

①データベースクラスタのルートディレクトリに、postgresql.conf ファイルが存在することを確認します。

```
[postgres@post-srv ~]$ ls $PGDATA
PG_VERSION  pg_dynshmem    pg_multixact  pg_snapshots  pg_tblspc             postgresql.conf
base        pg_hba.conf    pg_notify     pg_stat       pg_twophase           postmaster.opts
global      pg_ident.conf  pg_replslot   pg_stat_tmp   pg_xlog               postmaster.pid
pg_clog     pg_logical     pg_serial     pg_subtrans   postgresql.auto.conf
```

②vi エディタを使用して postgresql.conf ファイルを開きます。

```
[postgres@post-srv ~]$ vi $PGDATA/postgresql.conf
```

③ここでは、データベースへの最大接続数を設定する max_connectionsパラメータを変更します。デフォルトでは100に設定されていますが、ここでは120に増加します。postgresql.conf ファイルの該当箇所を変更し、ファイルを保存します。

※以下は、vi エディタによる postgresql.conf ファイルの編集例です。

```
・・・略・・・

#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
#port = 5432                            # (change requires restart)
max_connections = 100                   # (change requires restart)
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/tmp'       # comma-separated list of directories
                                        # (change requires restart)
#unix_socket_group = ''                 # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
                                        # (change requires restart)
```

max_connections パラメータの右側に(change requires restart)とあるため、このパラメータは変更後、postgres プロセスを再起動する必要があります。

④変更した max_connections パラメータの値を有効化するため、postgres プロセスを再起動します。pg_ctl で restartを実行します。

※-wは、起動・停止・再起動のコマンドで使用できる、処理の完了を待機するオプションです。

```
[postgres@post-srv ~]$ pg_ctl -w restart
LOG:  received smart shutdown request
LOG:  autovacuum launcher shutting down
waiting for server to shut down....LOG:  shutting down
LOG:  database system is shut down
 done
server stopped
waiting for server to start....LOG:  database system was shut down at 2016-04-06 16:05:34 JST
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
 done
server started
```

２.パラメータ設定の確認

データベースで有効化されているパラメータは、psql の show コマンドで確認できます。

■例題３－５： データベースのパラメータ設定を確認する。
例題 3-4 で設定した max_connections パラメータの値が有効化されている（120 になっている）ことを確認します。

①psql で postdb データベースにログインします。

```
[postgres@post-srv ~]$ psql -d postdb
psql (9.4.0)
Type "help" for help.

postdb=#
```

②show コマンドで、max_connections パラメータの値を確認します。

```
postdb=# show max_connections;
 max_connections
-----------------
 120
(1 row)
```

※show all を実行すると、設定されている全てのパラメータを一覧できます。

例題 3-4 で設定した値が有効になっていることが確認できました。

③設定されているパラメータ値は、current_setting システム関数でも確認できます。

```
postdb=# select current_setting('max_connections');
 current_setting
-----------------
 120
(1 row)
```

３.データベース稼動中にパラメータを変更する

パラメータには、静的なパラメータ（変更を有効にするには postgres プロセスの再起動が必要）と動的なパラメータ（データベース稼動中に変更できる）があります。
動的パラメータの変更は、データベースクラスタ全体で変更を有効にする alter system 文、データベース単位で有効にする alter database 文、特定ユーザの接続にだけ影響する alter role 文による方法があります。また、特定セッションのみ値を変更したい場合は、set コマンドによる変更も行えます。

■例題３－６： 接続している特定セッションのパラメータを変更する
動的パラメータの１つである enable_seqscan パラメータを、コマンドを実行する特定セッションだけ無効に変更します。

①show コマンドで、現在有効になっている enable_seqscan パラメータの値を確認します。
※postdb データベースに接続している状態からスタートしています。

```
postdb=# show enable_seqscan;
 enable_seqscan
----------------
 on
(1 row)
```

②set コマンドで enable_seqscan パラメータの値を off に変更します。

```
postdb=# set enable_seqscan to off;
SET
```

③off に変更されていることを確認します。

```
postdb=# show enable_seqscan;
 enable_seqscan
----------------
 off
(1 row)
```

enable_seqscan パラメータの値を、データベースを起動した状態で変更できたことが確認できました。