2008年6月6日（金） 12:40〜15:30
@PostgreSQL Conference 2008

MySQLユーザーのための

PostgreSQL入門

リナックスアカデミー 学校長
濱野 賢一朗

1

講師紹介

濱野 賢一朗（はまの けんいちろう）

- リナックスアカデミー　学校長
- 情報処理推進機構（IPA）オープンソフトウェアセンター 人材育成WG
- 情報処理推進機構（IPA）情報処理技術者試験 試験委員
- 日本LDAPユーザ会／日本Sambaユーザ会 発起人（の一人）
- 日本Apacheユーザ会／日本PHPユーザー会　スタッフ
- 「Oracle for Linux メーリングリスト」「WebDAV-JP ML」 管理人
- 著書に「オープンソースソフトウェアの本当の使い方」「Linux教科書 LPIC」「PHP プログラミング Black Book」「qmailで作る快適メール サーバー」、翻訳書に「PHPデスクトップリファレンス」などがある。
- 最近は、Linuxやオープンソースソフトウェアを理解した技術者を養成するための体系的な仕組みを模索中。

2

■概要

PostgreSQL のインストールから基本的な設定方法、利用方法を 160 分にぎゅーっと凝縮して解説する欲張りなチュートリアルです。一人で PostgreSQL を導入・利用・運用できることを目指します。

■対象者

PostgreSQL をきちんと導入・利用・管理できるようになりたい方

これから PostgreSQL を使おうと考えている方

PostgreSQL を体系的に学びなおしたい方

※ 本チュートリアルでは、SELECT 文や CREATE TABLE 文など基本的 なSQL 文を知っていることを前提します。

PostgreSQLの概要と特徴

■ オープンソースの本格的なORDBMS
- オブジェクトリレーショナルデータベース管理システム
- PostgreSQL Global Development Groupが開発
- カリフォルニア大学バークレイ校で開発されたPostgresをベースとして10年以上継続して開発されている
- SQL標準、トランザクションなど必要な機能をほぼ網羅している

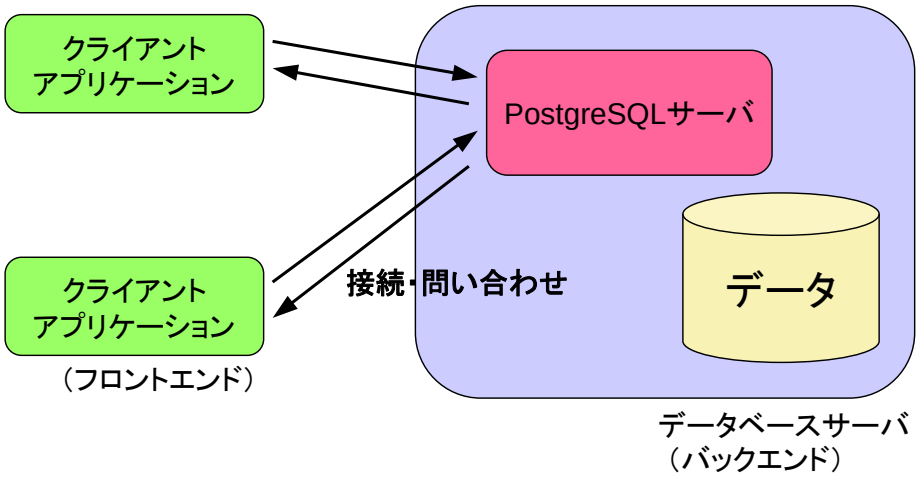
■ クライアント・サーバ型を採用している
- ネットワークを通じた問い合わせを行う

■ BSDライセンスで提供
- 誰でも自由に無償で利用・変更・配布することができる

4

PostgreSQLの基本構成

■ クライアント・サーバ構成



5

PostgreSQLの特徴

- SQL標準をサポート
  - SQL 2003の多くを実装している
  - 多少の構文や機能が異なる場合があるが、SQL標準に基づいた問い合わせはほぼサポートされる
- 強力なトランザクション機能・同時実行制御
  - 行レベルロック、読み取り一貫性などが保証される
- 豊富なプラットホームで動作
  - Linux/UNIX、MacOS X、Windowsなど多くの環境で動作する
- 国内での豊富な導入実績
  - 日本語を含むマルチバイト対応が古くより実現されている
  - 既に大規模システムへの適用が行われている

6

PostgreSQLのバージョン

- 最新バージョンは 8.3.1。
- メジャーバージョン（8.3）＋マイナーバージョン（1）
  - バージョン番号の上2つの数字がメジャーバージョンを表し、3つ目の数字がマイナーバージョンを表す
  - バグ修正やセキュリティ対応はマイナーバージョンアップ
  - 機能追加や仕様変更はメジャーバーションアップ
  - メジャーバージョンが変わるとデータベースの互換性がなくなる可能性があるため、バージョンアップに注意が必要
- 新メジャーバージョンはおよそ1年おきにリリース
  - 年末～年始にかけてリリースされる？
  - 8.3　2008年2月 ／ 8.2　2006年12月 ／ 8.1　2005年11月
    8.0　2005年1月 ／ 7.4　2003年11月 ／ 7.3　2002年11月

7

PostgreSQLのインストール（Linux）

- 1. アカウントの作成
- 2. アーカイブファイルなどの入手
- 3. Makefileの生成（configureの実行）
- 4. コンパイル・インストール
- 5. 環境設定
- 6. データベースクラスタの作成
- 7. データベースサーバの起動

8

アカウントの作成

■ PostgreSQLの管理者ユーザー

- セキュリティ上の配慮から、rootとは別に作成する
- 一般的に postgres を利用することが多い

9

■ユーザー postgres の作成例

```
# getent passwd          ← ユーザーpostgresが作成されていないことを確認
# useradd postgres
# passwd postgres        ← パスワードを設定（2回入力）
```

アーカイブファイルなどの入手

- インストール先ディレクトリを作成する
  - 一般的には /usr/local/pgsql が用いられる
  - 所有者をPostgreSQLの管理者ユーザーとする
- PostgreSQLのソースコードなどを入手
  - ファイル名は、postgresql-バージョン.tar.gzなど
    - 最新版は postgresql-8.3.1.tar.gz
  - 次のWebサイト等を辿ることでダウンロードできる
    - http://www.postgresql.org/
    - http://www.postgresql.jp/
- アーカイブファイルを展開して、INSTALLなどを確認
  - インストールに必要なツールやライブラリを確認する
  - Makeコマンドや Cコンパイラ、GNU Readlineライブラリなどを必要とする

10

■インストール先ディレクトリの準備

```
# mkdir /usr/loca/pgsql
# chown postgres.postgres /usr/local/pgsql
```

■アーカイブファイルのダウンロード・展開例

```
# su - postgres
$ cd /tmp
$ wget http://www.ring.gr.jp/pub/misc/db/postgresql/v8.3.1/postgresql-8.3.1.tar.gz
$ tar xzvf postgresql-8.3.1.tar.gz
$ cd postgresql-8.3.1
```

（ディレクトリ中のREADMEやINSTALLなどのドキュメントを確認）

Makefileの生成（configureの実行）

■ configureコマンドを用いて、環境を応じたコンパイル・インストール手順を生成する

■ 必要に応じてオプションを指定する

- インストール先ディレクトリの変更
- インストール機能の追加・削除 など

11

■configureのオプションの表示

```
$ configure --help
```

■configureの実行

```
$ configure
```

■configureの実行（インストール先ディレクトリを変更）

```
$ configure –prefix=/usr/local/postgresql-8.3.1
```

コンパイル・インストール

■ コンパイル
- 各種プログラムがコンパイルされる
- コマンド「make all」を実行

■ テスト
- 正常にコンパイルされて、動作することを確認する
- コマンド「make check」を実行
- 結果として「All 114 tests passed.」と表示されればOK

■ インストール
- 各ファイルを適切なディレクトリに配置する
- コマンド「make install」を実行

12

■実行例

```
$ make all
 :
All of PostgreSQL successfully made. Read to install.
$ make check
 :
======================
All 114 tests passed.
======================
 :
$ make install
 :
PostgreSQL installation complete.
```

■インストールされるディレクトリ・ファイル

インストールが完了すると/usr/local/pgsql（もしくはconfigure時に--prefixオプションで指定したディレクトリ）に各種ディレクトリ・ファイルが配置される。主要なディレクトリとしては次のものが挙げられる。

bin/	実行ファイル（postgres、psqlなど）
doc/	ドキュメントファイル
include/	C言語用のincludeファイル
lib/	C言語用のライブラリファイル
man/	オンラインマニュアルファイル

- $ configure
- $ ./configure

13

環境設定

■ 環境変数を指定する

- PGDATA　　データベースクラスタのパス
- PGLIB　　ライブラリのパス
- PATH　　コマンドのサーチパス
- さらにMANPATHやLD_LIBRARY_PATHを指定してもよい

■ bashの設定ファイルに指定する

- ユーザーpostgresのホームディレクトリにある .bashrcに設定を追記する
- 追記したら、sourceコマンドなどで指定した値を有効にする

14

■実行例

```
$ vi /home/postgres/.bashrc
  (ファイル末尾に次を追記する)
   PG=/usr/local/pgsql
   export PGDATA=$PG/data
   export PGLIB=$PG/lib
   export PATH=$PATH:$PG/bin
$ source /home/postgres/.bashrc
$ env | grep PG        ← 正しく PGDATA、PGLIB が指定されていることを確認
```

データベースクラスタの作成

■ データベースクラスタ

- PostgreSQLが管理するデータ領域
- 複数のデータベースのデータや設定ファイルなどで構成

■ データベースクラスタの作成

- initdbコマンドで作成する
- 環境変更PGDATAで指定したディレクトリに作成される

15

■initdbコマンドによるデータベースクラスタの作成例

```
$ initdb –encoding=UTF-8 –no-locale
```

■initdbコマンドの主なオプション

--encoding=文字エンコーディング	PostgreSQLの標準で用いる文字エンコーディング
--no-locale	ロケールを使用しないためのオプション

■データベースクラスタ内に作成される主なディレクトリ・ファイル

global/	データベース共通のデータが格納される
base/	データベースごとのデータが格納される
postgresql.conf	PostgreSQLの設定ファイル
pg_hba.conf	クライアント認証の設定ファイル

PostgreSQLの起動・停止

■ 起動・停止は、pg_ctl コマンドを用いて行える

■ PostgreSQLの起動
- `$ pg_ctl start`

■ PostgreSQLの停止
- `$ pg_ctl stop`
- 接続中のユーザーがセッションを切断するまで待って、PostgreSQLを停止する

16

RPMパッケージを用いたインストール

■ PostgreSQL関連パッケージをインストール

- postgresql
- postgresql-libs
- postgresql-server

■ 起動スクリプトを用いた起動・停止が可能

- /etc/init.d/postgresql start
- /etc/init.d/postgresql stop

■ テータベースクラスタは /var/lib/pgsql/data 以下に作成される

17

■PostgreSQL関連パッケージを用いたインストール（yum利用の場合）

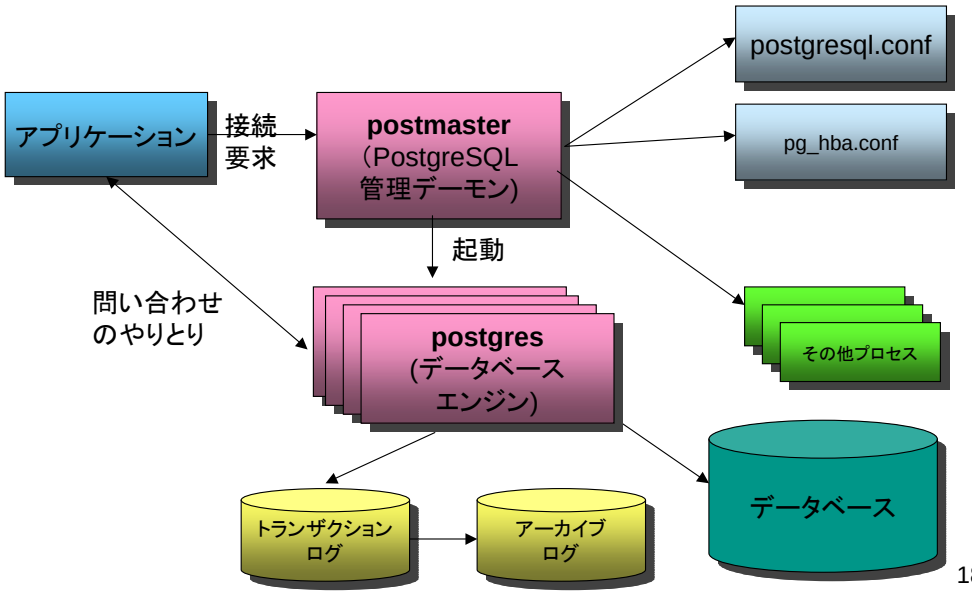
```
# yum -y install postgresql-server
```

■PostgreSQL RPM Building Project

Red Hat Enterprise Linux（CentOS）、Fedora Coreに対応したPostgreSQL 7.3～8.3のRPMパッケージは、PostgreSQL RPM Building Projectのyumリポジトリから入手することができる。ただし、ディストリビューション単位でのサポートを受けている場合には、サポート対象外となる可能性があるので注意が必要です。

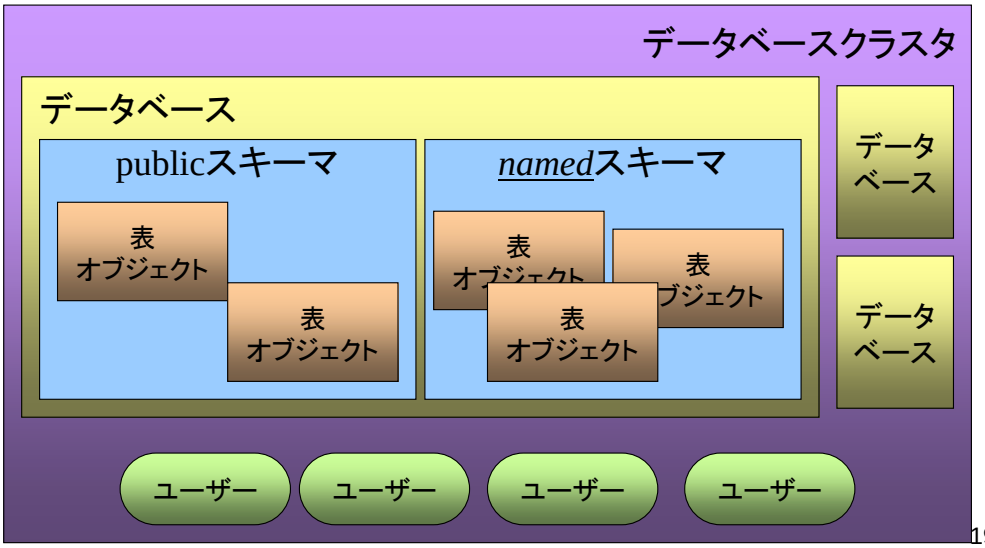
http://yum.pgsqlrpms.org/

PostgreSQLのプロセス構造



18

データベースクラスタの全体感（論理構造）



19

日本語マニュアル

■ PostgreSQL付属ドキュメントを日本語に翻訳したものが、オンラインで提供されている

- http://www.postgresql.jp/document/


PostgreSQL 8.3.1文書

PostgreSQL グローバル開発グループ

著作 著作 © 1996-2008 PostgreSQL グローバル開発グループ

目次

はじめに

PostgreSQL とは?
PostgreSQL の歴史
表記法
より進んだ情報
パフォーマンス・ガイドライン

I チュートリアル

1章 さあ始めましょう
2章 SQL 言語
3章 高度な機能

II SQL 言語

4章 SQL の構文
5章 データ定義

シンプルではあるが、必要な情報をよくまとまっているので、非常に便利。

調べたいことがあれば、まずはこのドキュメントを読んでみるとよい。

新バージョンの翻訳も素早く対応されている（感謝！）。

20

コマンド psql

■ PostgreSQLの対話ターミナル

- 実質的なPostgreSQLの管理用インターフェイス
- SQLの入力・問い合わせ結果表示だけでなく、psqlコマンド（バックスラッシュコマンド、メタコマンド）の実行ができる

■ psqlの起動

- psql データベース名
- psql データベース名 ユーザー名
- psql –h ホスト名 データベース名 ユーザー名　など

■ psqlの終了

- psqlコマンド ¥q を入力する

21

■データベースの一覧表示（-l オプション）

```
$ psql -l
            List of databases
     Name      |     Owner     | Encoding
---------------+---------------+----------
 postgres      | postgres      | UTF8
 template0     | postgres      | UTF8
 template1     | postgres      | UTF8
(4 rows)
```

データベースクラスタ作成時に、初期データベースとして template0 , template1 , postgres が作成されている。

■データベース template1に接続して、SELECT文で問い合わせを行い、psqlを終了する。

```
$ psql template1
Welcome to psql 8.3.1, the PostgreSQL interactive terminal.
  :
template1=# SELECT * from pg_user ;
  :
template1=# ¥q
```

SQL文の最後、SQL文とSQLの区切りは「;」で表現する。

psqlでは改行を行うとそのまま次の行にSQL文が繰り越されていると解釈される。

psqlコマンド（バックスラッシュコマンド、メタコマンド）

■ psql内で利用できるコマンド

- 文末のセミコロンは不要

■ 主なpsqlコマンド

コマンド	説　明
¥q	Psqlの終了
¥?	Psqlコマンドの一覧
¥h [SQL文]	SQL文のヘルプ
¥d	テーブル、インデックスなどの一覧
¥d [オブジェクト名]	データベースオブジェクトの情報表示

22

データベースの作成・削除

■ データベースの作成

- createdbコマンド
- SQL文「CREATE DATABASE」

■ データベースの削除

- dropdbコマンド
- SQL文「DROP DATABASE」

23

■データベースの作成・削除例

```
$ createdb sample1
$ createdb –encoding=EUC_JP sample2
$ psql –l
 List of databases
     Name      |     Owner     | Encoding
---------------+---------------+----------
 postgres      | postgres      | UTF8
 sample1       | postgres      | UTF8
 sample2       | postgres      | EUC_JP
 template0     | postgres      | UTF8
 template1     | postgres      | UTF8
(5 rows)
$ dropdb sample2
$ psql –l
 List of databases
     Name      |     Owner     | Encoding
---------------+---------------+----------
 postgres      | postgres      | UTF8
 sample1       | postgres      | UTF8
 template0     | postgres      | UTF8
 template1     | postgres      | UTF8
(4 rows)
```

テーブルの作成・削除

■ テーブルの作成

- CREATE TABLE文

```
CREATE TABLE テーブル名 (
    列名  データ型 [制約] , ・・・
)
```

■ テーブルの削除

- DROP TABLE文

24

■テーブル作成例

```
$ psql sample1
     :
sample1=# CREATE TABLE member (
sample1-#    code INTEGER PRIMARY KEY,
sample1-#    name TEXT NOT NULL,
sample1-#    phone TEXT         );
CREATE TABLE
sample1=# SELECT * FROM member ;
 code | name | phone
------+------+-------
(0 rows)
sample1=# INSERT INTO member(code,name,phone)
sample1-# VALUES (10,'hamano','03-1111-2222');
INSERT 0 1
sample1=# INSERT INTO member(code,name) VALUES (11,'sato');
INSERT 0 1
sample1=# SELECT * FROM member ;
 code |  name  |    phone
------+--------+--------------
   10 | hamano | 03-1111-2222
   11 | sato   |
(2 rows)
sample1=# ¥q
```

主なデータ型

■ PostgreSQLは多くのデータ型をサポートする

データ型	説　明
INTEGER	4バイト整数型
BIGINT	8バイト整数型
SERIAL	シーケンスを自動作成する整数型
VARCHAR(n)	可変長文字列
TEXT	文字数制限なし可変長文字列型
DATE	日付
TIME	時刻
TIMESTAMP	日付＋時刻

25

■データ型の詳細情報

データ型の仕様および型変換については、付属ドキュメントを参照してください。

第8章 データ型

http://www.postgresql.jp/document/pg831doc/html/datatype.html

■制約とデフォルト値

CREATE TABLE文では、次の制約を指定することができます。

NOT NULL

UNIQUE

PRIMARY KEY（主キー）

FOREIGN KEY / REFERNCES（外部参照キー）

CHECK

DEFAULT

データの操作

■ 次の文で操作する

- データの検索　SELECT文
- データの挿入　INSERT文
- データの削除　DELETE文
- データの更新　UPDATE文

■ SQL標準をほぼカバーしている

- PostgreSQL独自の記述方法もあり、知っていると便利な時がある。LIMITやOFFSETなど
- 当然、複雑な結合や副問い合わせなどもカバーしている。

26

トランザクション

- 複数の処理を単一の処理としてまとめる仕組み
  - 意味的に不可分や処理をまとめて、「すべて成功」か「すべて失敗」のいずれかであること を保障する。
- 明示しない場合、SQL 1文が1つのトランザクションとなる
  - SQL 1文ごとに、暗黙的にトランザクションが開始・完了する。
- 複数のSQL文から構成されるトランザクションの場合、明示的に指定する必要がある
  - トランザクションの開始　　BEGIN
  - トランザクションの正常終了　　COMMIT / END
  - トランザクションの異常終了　　ROLLBACK / ABORT
- 行レベルロックやデッドロックの回避など様々な同時実行制御の仕組みが採用されている

27

■トランザクションの利用例（大した例ではないが・・・）

```
sample1=# SELECT * FROM member ;
 code |  name  |    phone
------+--------+--------------
   10 | hamano | 03-1111-2222
   11 | sato   |
(2 rows)
sample1=# BEGIN ;
sample1=# INSERT INTO member(code,name)
sample1-# VALUES (12,'tanaka');
sample1=# UPDATE member
sample1-# SET phone='090-2222-3333'
sample1-# WHERE code=11;
sample1=# COMMIT;
sample1=# SELECT * FROM member ;
 code |  name  |     phone
------+--------+---------------
   10 | hamano | 03-1111-2222
   12 | tanaka |
   11 | sato   | 090-2222-3333
(3 rows)
```

```
sample1=# BEGIN ;
sample1=# INSERT INTO
member(code,name)
sample1-# VALUES (13,'suzuki');
sample1=# SELECT * FROM member ;
 code |  name  |     phone
------+--------+---------------
   10 | hamano | 03-1111-2222
   12 | tanaka |
   11 | sato   | 090-2222-3333
   13 | suzuki |
(4 rows)
sample1=# ROLLBACK ;
sample1=# SELECT * FROM member ;
 code |  name  |     phone
------+--------+---------------
   10 | hamano | 03-1111-2222
   12 | tanaka |
   11 | sato   | 090-2222-3333
(3 rows)
```

様々なデータベースオブジェクト

■ PostgreSQLは多くのデータベースオブジェクトの作成できる

- ビュー（別表）
- インデックス（索引）
- シーケンス（順序）
- ドメイン
- ルール
- トリガー
- スキーマ ・・・

28

設定ファイル postgresql.conf

■ PostgreSQLの設定ファイル

- 実質的にすべてこのファイルで設定する
- 「設定項目名 = 設定値」のスタイルで記述する

■ 主な設定項目

設定項目	説　明
listen_addresses	ネットワーク接続を受け付けるローカルアドレス
max_connections	データベースサーバの最大同時接続数
shared_buffers	共有メモリバッファのサイズ
log_destination	ログの出力先指定（標準出力／syslog）

29

■postgresql.confの設定例

```
listen_addresses = 192.168.0.10, 127.0.0.1
max_connections = 200
shared_buffers = 96M
log_destination = syslog
```

■変更後の設定の有効化

設定ファイルpostgresql.confを変更した後、変更した内容を適用させるためにはpg_ctlコマンドなどを用いて設定ファイルを再読込させます。

```
$ pg_ctl reload
```

クライアント認証の設定

■ 設定ファイル pg_hba.conf で設定する

- 1つの設定を1行で記述する
- 上から順番に評価される
- 条件を満たさないアクセスは拒否される

接続タイプ	データベース名	ユーザー名	ネットワークアドレス	認証方式

30

■pg_hba.confの設定例

サーバ内からのUNIXドメインソケットによるアクセスは無条件に許可する

```
local   all       all                             trust
```

サーバ内からのループバックインターフェイス経由でのアクセスは無条件に許可する

```
host    all       all     127.0.0.1/32            trust
```

ネットワーク 192.168.100.0/24 からのアクセスは無条件に許可する

```
host    all       all     192.168.100.0/24    trust
```

データベース sample1 にユーザーpenguinがネットワーク172.25.0.0/16 からのアクセスは、MD5でハッシュしたパスワード情報を用いて認証し、成功した場合のみアクセスを許可する

```
host    sample1   penguin 172.25.0.0/16       md5
```

■詳細な記述方法

付属ドキュメントを参照してください。

第21章 クライアント認証

http://www.postgresql.jp/document/pg831doc/html/client-authentication.html

ユーザーの作成・削除

■ PostgreSQLでは、ユーザーなどの概念をロールとして管理している

■ ユーザーの作成
- createuserコマンド
- CREATE ROLE文（CREATE USER文）

■ ユーザーの削除
- dropuserコマンド
- DROP ROLEコマンド（DROP USER文）

31

■ユーザー（ロール）の作成例

```
$ createuser -P penguin
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) n

$ psql sample1
sample1=# CREATE ROLE  hamano PASSWORD 'himitu';
sample1=# SELECT rolname, rolsuper, rolinherit, rolcreaterole, rolcreatedb
sample1-# FROM pg_roles ;
    rolname    | rolsuper | rolinherit | rolcreaterole | rolcreatedb
---------------+----------+------------+---------------+-------------
 postgres      | t        | t          | t             | t
 penguin       | f        | t          | f             | t
 hamano        | f        | t          | f             | f
(3 rows)
sample1=# ¥q

$ psql template1 penguin
template1=> ¥q
```

PostgreSQLをRPMパッケージで導入したした時に、pg_hba.confにて認証方式が「ident sameuser」と設定されており、Linuxのログインユーザーと同一名のデータベースユーザーでなければ認証を通さない場合があるため、注意してください。ログインユーザー名と異なるデータベースユーザー名で接続させるには、pg_hba.confの設定を変更する必要があります。

アクセス権の付与・剥奪

■ テーブルなどデータベースオブジェクト単位でのアクセス権を管理できる
- Psqlコマンド「¥z」で付与状況を把握できる

■ アクセス権の付与
- GRANT文

GRANT アクセス権 ON テーブル名 TO 対象

■ アクセス権の剥奪
- REVOKE文

REVOKE アクセス権 ON テーブル名 FROM 対象

32

■アクセス権の付与例

```
$ psql sample1
sample1=# GRANT ALL ON member TO penguin ;
sample1=# GRANT select,insert,update ON member TO hamano ;
sample1=# ¥z
                          Access privileges for database "sample1"
 Schema |  Name  | Type  |                         Access privileges
--------+--------+-------+-------------------------------------------------------------------------
 public | member | table | {postgres=arwdxt/postgres,penguin=arwdxt/postgres,hamano=arw/postgres}
(1 row)
sample1=# ¥q
```

VACUUM

- PostgreSQLは追記型のデータベースのため、不要となった領域を削除（再利用できるように）する必要がある
- 削除しないままだと、パフォーマンスの低下やディスクの浪費などがおこってしまう
  - VACUUM中もデータベースにはアクセス可能
- 不要領域を削除を行うのが VACUUM
  - SQL文として「VACUUM」を実行
  - シェル上で「vacuumdb データベース名」を実行

33

■全てのデータベースに対してVACUUMを行う

```
$ vacuumdb –a
```

■データベースsample1に対してVACUUMを行う

```
$ psql sample1
sample1=# VACUUM ;
sample1=# ¥q
```

ANALYZE

- 統計情報の更新を行うにはANALYZEを実行する
- インデックスを利用するかどうかは統計情報によって決まるため、ANALYZEを実行しないと最適な検索方法が利用されない可能性がある
- ANALYZEは、vacuumdb -z でも行える。
- 定期的に実行するほか、テーブルの内容が大きく変わった後にANALYZEを実行するのが望ましい。

34

■全てのデータベースに対してVACUUMとANALYZEを行う

```
$ vacuumdb -az
```

■データベースsample1に対してVACUUMとANALYZEを行う

```
$ psql sample1
sample1=# VACUUM ANALYZE ;
sample1=# ¥q
```

■cronを用いて定期的にVACUUMとANALYZEを行う

```
$ crontab -e
```

（viなどのエディタが起動する。次の設定を追記して保存する）

```
10 * * * * /usr/local/pgsql/bin/vacuumdb -az
$ crontab -l
```

※参考
VACUUMやANALYZEのほかに、REINDEX文を実行した方がよい状況もある。

バックアップ・リストア（データベース単位）

■ データベース単位でのバックアップ

- pg_dumpコマンドを用いる
- データベース稼働中にバックアップを取得できる
- バックアップデータは、（原則的に）psqlが処理できるSQL文・psqlコマンドの組み合わせで構成されている。

■ バックアップデータのリストア

- psqlコマンドを用いて、バックアップデータをリストアする。

35

■バックアップ例

```
$ pg_dump sample1 > sample1_backup_20080606
```

■リストア例

```
$ dropdb sample1
$ createdb sample1
$ psql sample1 < sample1_backup_20080606
```

※参考

ラージオブジェクトなどを用いている場合は、この限りではないので注意が必要です。

バックアップ・リストア（データベースクラスタ全体）

■ データベース停止時のバックアップ

- データベースクラスタのディレクトリ以下をまるごとバックアップする方法がある

■ 稼働中のバックアップ

- pg_dumpallコマンドを用いてバックアップを取得する。
- リストアはpsqlコマンドを用いて行う。データベースクラスタ作成後にデータベース template1に接続して、バックアップデータをリストアするのがポイント。

36

■バックアップ例

```
$ pg_dumpall > all_backup_20080606
```

■リストア例

```
$ pg_ctl stop
$ rm –Rf /usr/local/pgsql/data
$ initdb –encoding=UTF-8 –no-locale
$ psql template1 < all_backup_20080606
```

CVS形式でのエクスポート

■ psqlのCOPY文を使うと便利！

- DELIMITER句で区切り文字を指定する
- SQL「COPY文」を使う方法もある（バックエンド）

```
¥COPY テーブル名 TO ファイルパス DELIMITER ‘,’
```

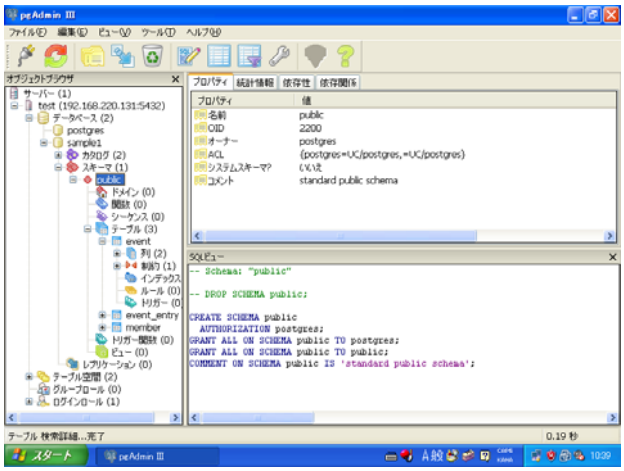
■ テーブル単位のバックアップにも利用できる

37

PgAdminIIIの利用

■ PostgreSQLのGUI管理ツール

- 管理・開発に便利

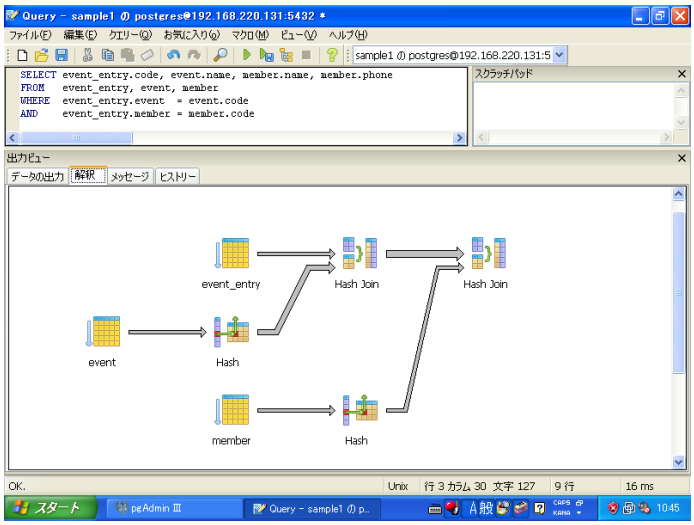


昨晩、1.8.4 が
リリースされた！

38

PgAdminIIIの利用

■ 実行計画なども見やすい！



39

まとめ

■ PostgreSQLは高機能・高性能
- 本格的な業務システムとして利用できる機能や性能
- 実績もかなりある

■ PostgreSQLは使いやすい
- すぐに使える、他のRDBMSの知識がそのまま使える
- 「ちょっとだけ知っている」必要があるが、それを理解すればやさしい！

■ PostgreSQLは奥が深い
- Postgresql.confのパラメータをチューニングしたり、データの格納場所（テーブルスペース）をうまく管理することで、より性能を引き出すことができる
- 便利なツールも実はいろいろある。

40

■配付資料の最新版

（配付資料には印刷されておらず）チュートリアル時に新たに追加したスライドなどは下記のURLで公開する予定です。

http://www.todo.ne.jp/doc/200806/pgcon2008/

■より詳しい研修は

リナックスアカデミーやSRA OSS, Inc.日本支社が実施しているPostgreSQL研修をご受講頂くことをオススメします。定期開催コースをご受講頂くこともできますし、1社研修の実施もご提案できます。

http://osu.linuxacademy.ne.jp/training/course/