SRA OSS, INC.

LPI-JAPAN

OSS-DB Exam Silver 技術解説無料セミナー

2018/1/20

SRA OSS, Inc. 日本支社
千田貴大

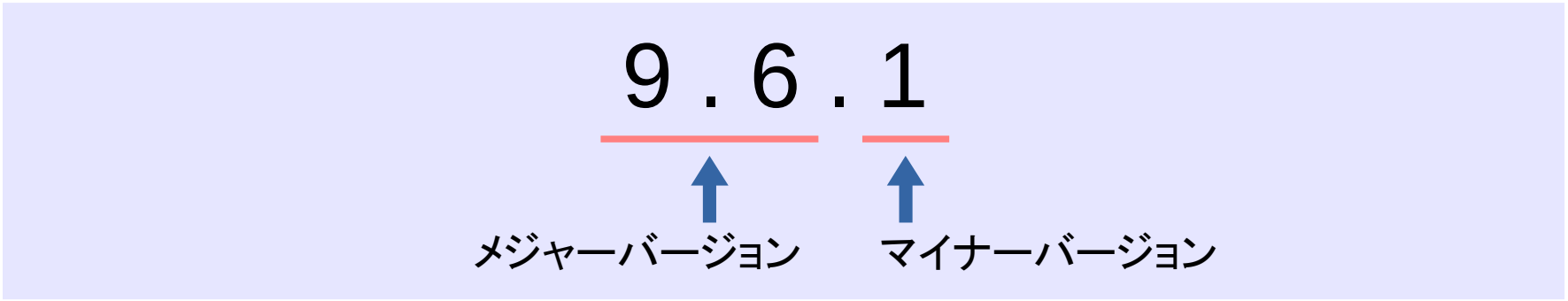
目次

- PostgreSQL の特徴
- インストール
  - インストール方法
  - 環境変数
  - データベースクラスタ
  - 起動・停止
- 設定ファイル
  - postgresql.conf
  - ログ出力設定
- 環境設定
  - ロール作成
  - データベース作成
  - psql
  - バックスラッシュコマンド
- バックアップ・リストア
  - PITR (Point In Time Recovery)
- VACUUM / ANALYZE
- セキュリティ管理
  - pg_hba.conf
  - ユーザ権限
  - テーブル権限

PostgreSQLの特徴

- 標準SQLの大部分とその他の先進的な機能をサポートする本格的なオープンソースRDBMS
- 20年以上の歴史を持ち、現在も活発な開発体制
  - 1年に1度のメジャーバージョンアップ
  - 1年に数回のマイナーバージョンアップ
- 豊富なプラットフォームに対応
  - Unix系OS全般、Windows 2000 SP4以降、Mac OS X
- 豊富なサポート言語
  - C, ECPG, C++, Java, Tcl/TK, Python, Perl, PHP, Ruby, ODBC, .NET Data Providerなど

バージョン番号規則



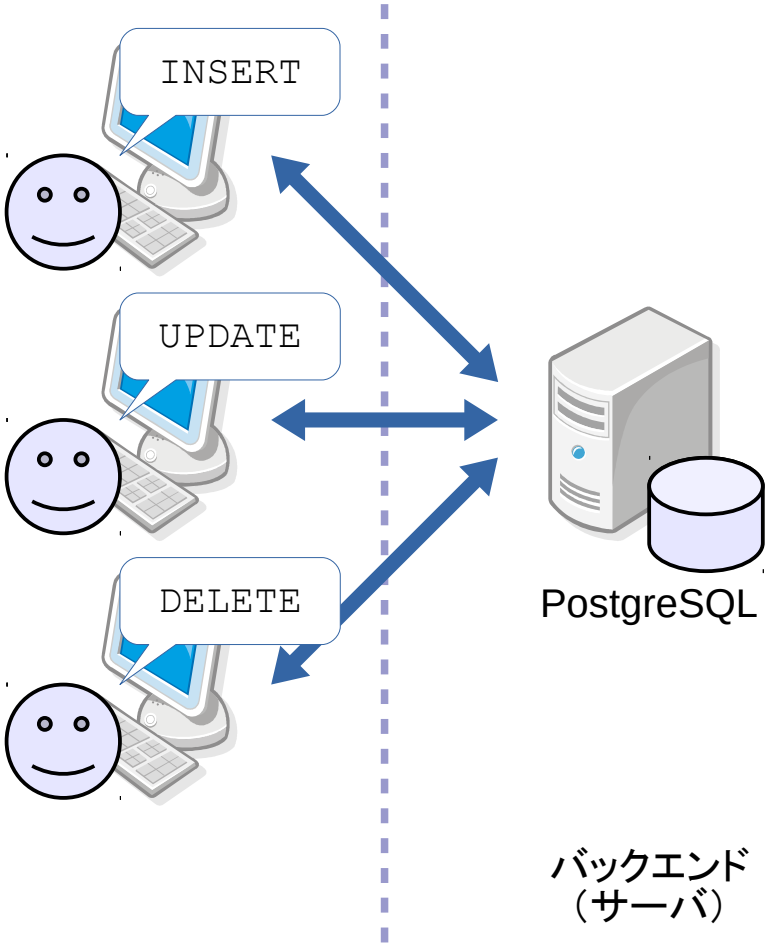
- メジャーバージョンアップでは仕様の追加・変更
  - 移行には付属コマンドでのバックアップ・リストアが必要
- マイナーバージョンアップでは主にバグ修正

コミュニティの役割

- PostgreSQL公式開発コミュニティ
  - 仕様検討、開発、リリース、不具合報告
- 日本 PostgreSQL ユーザ会（JPUG）
  - 迅速な最新ドキュメント翻訳
- PostgreSQL エンタープライズ・コンソーシアム（PGECons）
  - 毎年テーマ別に検証および結果公表
  - 公式開発コミュニティへの改善リクエスト

- PostgreSQLライセンス
  - BSDライセンスに類似
    - 広告条項はなし（修正BSDライセンス）
  - 使用、複製、改変、配布の自由
    - 複製においては以下を含めることが条件
      - 著作権表示
      - ライセンス条文
      - 免責条項
    - 具体的には、PostgreSQLソースコードに添付されている COPYRIGHTファイルを配布物に含めるか、その内容を マニュアルに印刷すればよい

- libpqプロトコル
  - クライアント・サーバのOSの違いを吸収
- 軽量クライアント
  - サーバの変更に影響されにくい



- 文字エンコーディング
  - データベースごとに指定
  - フロントエンド（バックエンドとの通信）ごとに変更できる

- 日本語を扱う一般的な組み合わせ

バックエンド（データベース）	フロントエンド（クライアント）
UTF8	UTF8, EUC_JP, SJIS
EUC_JP	UTF8, EUC_JP, SJIS

PostgreSQLのインストール

- Linux
  - ソースコードからのインストール
  - パッケージによるインストール
    - ディストリビュータ配布のパッケージ
    - PostgreSQL開発元配布のパッケージ
- Windows
  - one click installerによるインストール

ソースコードからのインストール

SRA OSS, INC. OSS-DB

- ソースコードの入手
  - https://www.postgresql.org/ftp/source/ など
- postgres ユーザの作成

```
# useradd postgres
```

存在していない場合は作成

- ダウンロードしたソースコードを展開

```
$ su -
# tar jxf postgresql-9.6.6.tar.bz2 -C /usr/local/src
# chown -R postgres /usr/local/src/postgresql-9.6.6
```

- インストール先ディレクトリ作成

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

省略可能。省略した場合、
インストール時に
root権限が必要

- configure
  - コンパイルの準備（必要なツールの有無を調査）

```
# su - postgres
$ cd /usr/local/src/postgresql-9.6.6
$ ./configure
```

  - よく使われるオプション

```
$ ./configure --help
```

    - --prefix=DIR

      インストール先を指定する（デフォルトは/usr/local/pgsql）

ソースコードからのインストール

SRA OSS, INC. OSS-DB

- コンパイル

```
$ make world
```

- リグレッションテスト
  - 仮のデータベースクラスタでPostgreSQLを起動し、想定どおりに正しく動作するかテストする

```
$ make check
```

- インストール

```
$ make install-world
```

  - インストール先(/usr/local/pgsql)が postgres 所有でなければ root 権限が必要

パッケージによるインストール

- PostgreSQLコミュニティがビルドして、提供しているバイナリパッケージ
  - 各Linuxディストリビューションに合わせたパッケージが必要
- yum レポジトリ
  - yum レポジトリを登録することで yum コマンドから各 PostgreSQL の RPM パッケージのインストールやマイナーアップデートが可能となる
  - http://yum.postgresql.org/ で yum レポジトリの設定を行う RPM ファイルを取得し、登録

```
# rpm -ivh pgdg-centos96-9.6-3.noarch.rpm
```

パッケージによるインストール

- 基本機能
  - postgresql96（クライアント）
  - postgresql96-server（サーバ）
  - postgresql96-libs（共有ライブラリ）
- オプション機能
  - postgresql96-contrib（追加モジュール）
  - postgresql96-docs（ドキュメント）
  - postgresql96-devel（開発用ライブラリ）
  - postgresql96-pl*（手続き言語）

パッケージによるインストール

- yum コマンドでインストール

```
# yum install postgresql96 postgresql96-libs postgresql96-server
```

- postgres ユーザアカウントが作成される
  - ホームディレクトリは /var/lib/pgsql
  - 環境変数の設定 /var/lib/pgsql/.bash_profile
- インストール先ディレクトリ
  - /usr/pgsql-9.6
- セットアップスクリプト
  - /usr/pgsql-9.6/bin/postgresql96-setup

- http://www.postgresql.org/download/windows
  - EnterpriseDB 社が提供している
  - 導入が簡単な one click installer
    - Windows版 PostgreSQL サーバ/クライアント
    - pgAdmin III
    - Stack Builder
  - データベースクラスタの作成
  - Windowsサービスとして動作
  - データベースのスーパユーザ "postgres" の作成

- インストーラの設定手順
  - ユーザアカウントの作成
  - データベースクラスタの作成
  - Windows サービスとして自動起動
- ディレクトリ構成
  - インストール時に指定したディレクトリ
    - C:¥Program Files¥PostgreSQL¥9.6¥
  - データベースクラスタ
    - C:¥Program Files¥PostgreSQL¥9.6¥data¥

- PostgreSQL コマンドディレクトリをコマンドパスに追加
  - ~/.bash_profileに以下を追加
    - /usr/local/pgsql（デフォルトのインストール先）にインストールした場合の例

```
$ vi ~/.bash_profile
PGHOME=/usr/local/pgsql
export PATH=$PGHOME/bin:$PATH
export LD_LIBRARY_PATH=$PGHOME/lib:$LD_LIBRARY_PATH
export MANPATH=$PGHOME/share/man:$MANPATH
export PGDATA=/usr/local/pgsql/data
```

  - 設定の反映

```
$ . ~/.bash_profile
```

    - 反映後 psql などのコマンドにパスが通っていることを確認

- $PGDATAで指定したディレクトリを作成し、
  データベースクラスタの初期化を行う

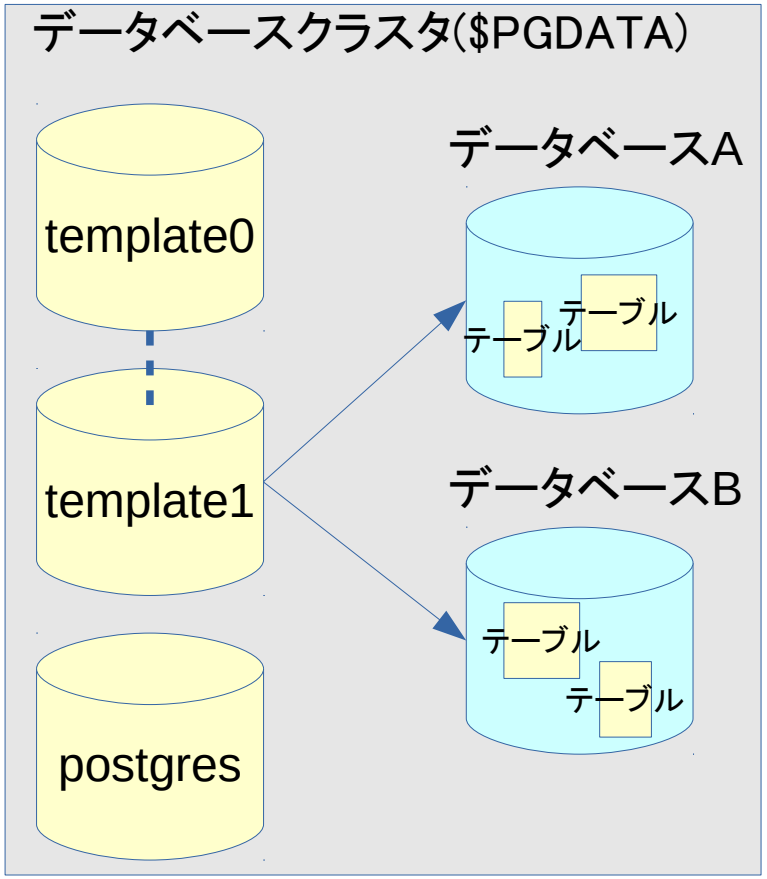
```
[postgres]$ initdb --encoding=UTF8 --no-locale
```

  - --encoding=UTF8
    - **デフォルトの文字エンコーディングをUTF8に設定**
  - --no-locale または --locale=C
    - **デフォルトのロケールを「利用しない」に設定**
      主に日本語と英語を格納するならロケールは不要

データベースクラスタ

SRA OSS, INC. OSS-DB

- **initdb**を実行したユーザが**PostgreSQL**の
  スーパーユーザになる
- テンプレートデータベース
  （template0, template1）と
  postgres データベースが
  作成される
  - 実際に使用すべき
    データベースではない
  - template0は書き込み不可
  - template1は書き込み可



- データベースデーモンの起動

```
$ pg_ctl start -w
```

- データベースデーモンの停止
  - 直ちに終了したい場合（fast mode）

```
$ pg_ctl stop
```

接続中のユーザが行っている処理は無効になる

  - 全てのクライアントが切断するまで待つ（smart mode）

```
$ pg_ctl stop -m s
```

  - 適切な終了処理をせずに直ちに停止（immediate mode）

```
$ pg_ctl stop -m i
```

次回起動時に復旧する

設定ファイル

- PostgreSQLの設定ファイル
  - postgresql.conf
  - pg_hba.conf（「セキュリティ管理」で取り上げます）
- デフォルトでは$PGDATAにある
  - 書式

```
変数　＝　値
```

  - 設定の反映

```
$ pg_ctl reload
$ pg_ctl restart -w
```

(change requires restart)
の記述があるパラメータを変更した場合

- データベースの実行時の設定項目のこと
  - 各項目をGUC変数と呼ぶことがある
- postgresql.confで設定
- SHOW／SETで閲覧・設定
  - 実行時に変更できる
    - client_encodingなど
  - reloadが必要
    - log_connectionsなど
  - 再起動が必要
    - shared_buffersなど

```
SET name TO value;
```

```
SHOW name;
```

```
$ psql
postgres=# SHOW listen_addresses;
 listen_addresses
------------------
 *
(1 row)

postgres=# SHOW ALL;
(省略)
```

\q で終了

ログ出力設定

SRA OSS, INC. OSS-DB

- ログ取得の目的
  - 不正アクセスの検知
  - 障害の早期発見
  - デバッグ
- ログ取得の方法
  - Syslogに取る
    - Windowsならイベントログに
  - 標準エラー出力のリダイレクト

ログ出力設定

- PostgreSQL自身がログ管理
  - $PGDATA/pg_log以下にログファイルができる
  - ファイルローテーションされる
- postgresql.confを設定

```
#log_destination = 'stderr'
logging_collector = on
#log_directory = 'pg_log'
log_filename = 'postgresql-%d.log'
log_line_prefix = '%t [%p] '
```

```
$ pg_ctl restart -w
```

```
postgres=# ERROR;
```

適当な構文エラーを
発生させてログ出力確認

```
$ less $PGDATA/pg_log/postgresql-<今日の日付>.log
```

ログ出力設定

- Syslog（Unix系OSのログを記録する機能）を利用
  - /var/log 以下のログファイルに出力
  - 複数のアプリケーションのログを一元管理できる
- postgresql.confを設定

```
log_destination = 'syslog'
#syslog_facility = 'LOCAL0'
#syslog_ident = 'postgres'
log_line_prefix = ''
```

  - postgresql.conf再読み込み（再起動でも可）

```
postgres$ pg_ctl reload
```

- /etc/rsyslog.confを変更
  - ファシリティがLOCAL0のログを/var/log/postgresql というファイルへ出力する設定

```
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
```

```
*.info;mail.none;authpriv.none;cron.none;local0.none   /var/log/messages
local0.*                                               /var/log/postgresql
```

  - Syslogをrestart

```
root# service rsyslog restart
```

- 設定変更を確認

```
postgres=# ERROR;
```

適当な構文エラーを発生させてログ出力

```
# less /var/log/postgresql
```

環境設定

ロール作成

- データベースユーザ(ロール)を登録し
  そのロールで操作する
- createuser　(PostgreSQLコマンド)
  - -d オプションでデータベース作成権限を与えます

```
$ createuser --help
$ createuser -d ロール名
```

- CREATE ROLE (SQL文)
  - CREATEDB でデータベース作成権限を与えます
  - LOGIN でログイン権限を与えます
    (createuser の場合デフォルトで付与されています)

```
=# CREATE ROLE ロール名 CREATEDB LOGIN
```

データベース作成

- データベースクラスタ（$PGDATA）には複数のデータベースが作成できる
- 作成方法
  - createdb （PostgreSQLコマンド）

```
[postgres]$ createdb データベース名
```

    - データベース名を省略すると、コマンドを実行しているOSのユーザと同じ名前のデータベースを作成

  - CREATE DATABASE文 （SQL）

```
=# CREATE DATABASE データベース名
```

- psql ＝ SQL発行ツール
- psqlの起動方法

```
psql [ オプション ] . . [ DB 名 [ ユーザ名 ] ]
```

  - 例

```
$ psql postgres <一般ユーザ名>
psql (9.6.1)
Type "help" for help.
postgres=>
```

接続しているデータベース名

- プロンプト
  - 「**=#**」… 接続しているユーザはスーパーユーザ
  - 「**=>**」… 接続しているユーザは一般ユーザ
    - 2行目以降は「**-#**」「**->**」

```
userdb=> SELECT * FROM --ここにコメントが書けます
userdb-> pg_user;      --ここにコメントが書けます
```

- SQLの発行
  - 「;」でSQLの終わりを意味し、コメントは「--」
  - SQL構文中の空白やタブ、改行は一つの空白と扱う
  - SQL構文の大／小文字は区別されない　日本語も可
  - Linuxでは「Ctrl-C」で入力をキャンセル
  - psqlの終了方法 … 「\q」または「Ctrl-D」

psql

-h ホスト名	データベースホスト名を指定
-p ポート番号	TCP/IPポートの指定
-d データベース	接続するデータベース名を指定
-U ユーザ名	接続ユーザ名を指定
-c “SQL文”	引数で与えたSQL文（１つ）を実行させる
-f ファイル名	ファイルに記述してあるSQLコマンドを受け付け
-o ファイル	出力結果をファイルに格納する
-l	データベースの一覧を表示
-t	カラム名と行数フッターなどの表示を無効にする
-V	psqlのバージョンを表示（--version）
-?	psqlのヘルプを表示（--help）

バックスラッシュコマンド

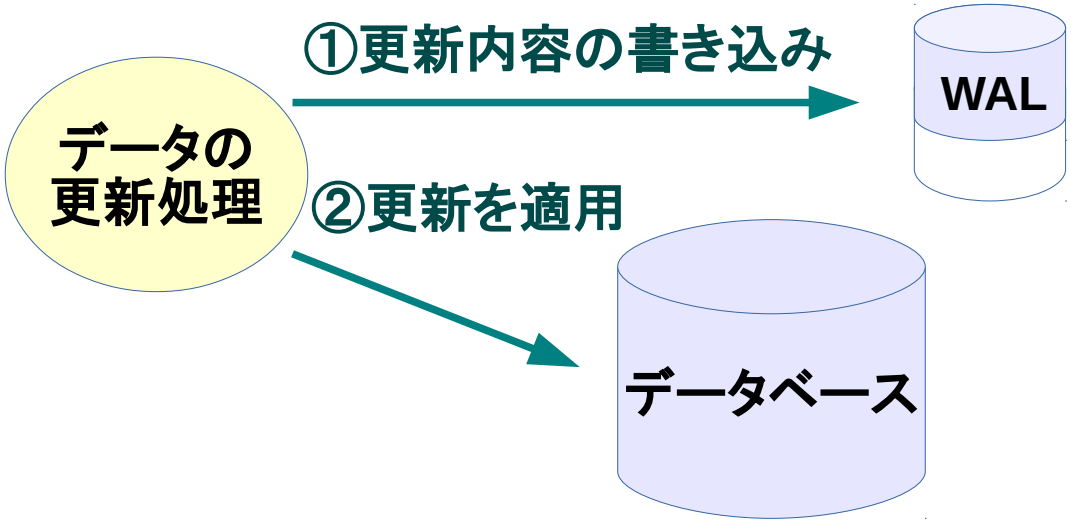
SRA OSS, INC. OSS-DB

\?	コマンド一覧
\encoding	エンコーディングの表示・設定
\h [SQL 文]	SQL 文のヘルプ
\l	データベース一覧
\d[t|i|s|v|S]	テーブル、インデックス、シーケンス、ビュー、システムテーブルの一覧
\d [NAME]	指定したテーブル、インデックス、シーケンス、viewを表示
\d+ [NAME]	指定したオブジェクトの詳細を表示、view定義を表示
\dp （\z）	テーブル、ビュー、シーケンスのアクセス権限一覧
\du	ロール（データベースユーザ）一覧
\df	関数一覧
\ef [NAME]	関数編集、または作成
\x	テーブルの表示モードを変更

バックアップ・リストア

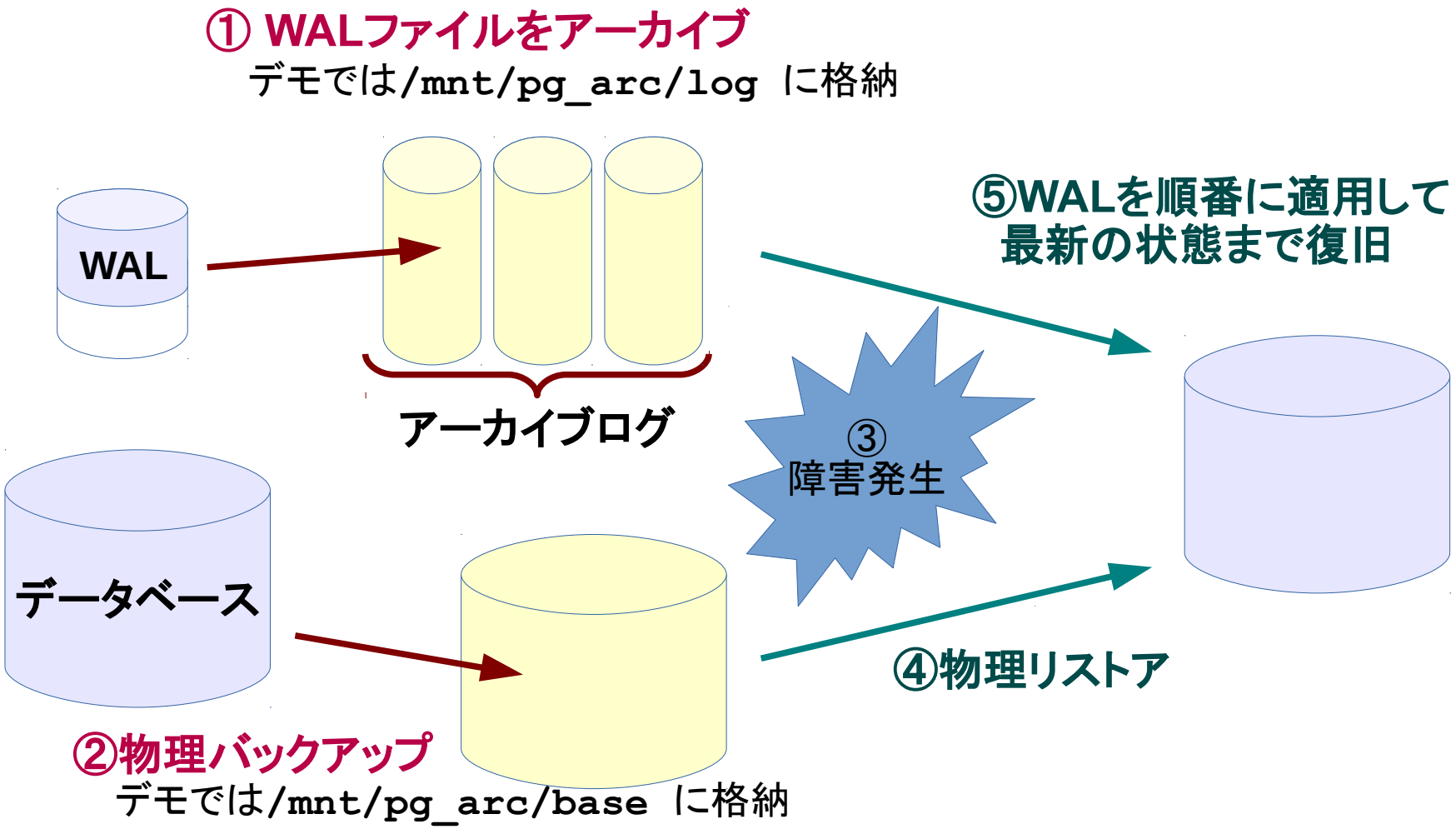
- バックアップ戦略
  - pg_dumpコマンドによる論理バックアップ
  - データベースクラスタ全体の物理バックアップ
  - PITR (Point-In-Time Recovery)
- その他
  - レプリケーション機能の利用
    - PostgreSQLのストリーミングレプリケーション
    - レプリケーションソフトウェアの利用
  - 外部ツールを使う

- Point In Time Recovery
  - ベースバックアップとアーカイブログを使って
    データベースを最新の状態までリストアする手法
- PostgreSQL のデータ書き込みの流れ



(Write-Ahead Log)
データベースクラスタ内の
pg_xlog ディレクトリに格納

※不要になったWALは
通常削除される



WALアーカイブの設定

- 準備
  - PITRで使用するディレクトリを作成

```
$ su -
# mkdir /mnt/pg_arc
# chown postgres /mnt/pg_arc
$ su - postgres
$ mkdir /mnt/pg_arc/base
$ mkdir /mnt/pg_arc/log
```

- WALファイルをアーカイブする設定
  - postgresql.conf を編集

```
wal_level = replica
max_wal_senders = 2
archive_mode = on
archive_command = 'cp "%p" "/mnt/pg_arc/log/%f"'
```

%p WALの相対パスに置き換わる
%f WALのファイル名に置き換わる

```
$ pg_ctl restart -w
```

WALアーカイブ設定の確認

SRA OSS, INC. OSS-DB

- pgbench
  - PostgreSQLに同梱されているベンチマークツールで、通常は性能測定に利用しますが、ここでは多数のデータ更新を発生させる目的で利用

```
$ pgbench -i -s 10
$ pgbench
```

-i 初期化
-s スケール

  - データが投入されたことで、WALがアーカイブされていることを確認

```
$ ls /mnt/pg_arc/log
000000010000000000000003
000000010000000000000004
```

pg_basebackup

SRA OSS, INC. OSS-DB

- データベースサーバを停止せずに
  物理バックアップを取得するコマンド

- バックアップモードへの移行を自動的に行い
  データベースクラスタ全体をコピー
  - スーパーユーザ(postgres)で実行
  - PostgreSQLのレプリケーション接続を経由してコピーを取得
  - あらかじめサーバ側へ
    ストリーミングレプリケーションと同等の設定が必要

- postgresql.conf
  - wal_level
    - WALをどのくらい詳細に出力するか指定
    - replica … ストリーミングレプリケーション機能や pg_basebackup を利用する場合
  - max_wal_senders
    - WAL送信プロセスの最大数

```
$ cd $PGDATA
$ vi postgresql.conf
$ vi pg_hba.conf
$ pg_ctl restart -w
```

```
wal_level = replica
max_wal_senders = 2
```

- pg_hba.conf（クライアント認証設定）
  - local接続でreplication(仮想データベース)へ postgresロールでの接続について無条件で接続を許可

```
local   replication   postgres                    trust
```

- ベースバックアップを作成

```
$ pg_basebackup -D /mnt/pg_arc/base/001 -P
46282/46282 kB (100%), 1/1 tablespace
$ ls /mnt/pg_arc/base/001
```

物理バックアップが取得できていることを確認

  - -D　　バックアップ先ディレクトリを指定
  - -P　　進行状況を表示
  - -h, -p　接続先ホスト、ポートを指定
    （デフォルトではlocalの5432ポートへ接続）
  - -Xs　　リストアに必要なWALを同時に取得

- リカバリ確認用にデータ更新
  - pgbench にて適当にデータ更新後確認用にテーブルを作成

```
$ pgbench -c 30 -t 50
$ psql
=# CREATE TABLE test (id INT);
=# \d
=# \q
```

-c 同時接続数
-t トランザクション数

- データベースクラスタが壊れたことを想定
  - PostgreSQLを強制停止し、$PGDATA を別名に変更します

```
$ pg_ctl stop -m i
$ cd /usr/local/pgsql
$ mv data data_crash
```

- ベースバックアップを展開しログを削除

```
$ pwd
/usr/local/pgsql
$ cp -r /mnt/pg_arc/base/001 data
$ ls
bin  data  data_crash  include  lib  share
$ cd data/
$ rm -rf pg_log/*
```

- リカバリ設定ファイルを作成

```
$ vi recovery.conf
```

%p WALの相対パスに置き換わる
%f WALのファイル名に置き換わる

```
restore_command = 'cp "/mnt/pg_arc/log/%f" "%p"'
```

- リカバリモード
  - リカバリ設定ファイルrecovery.confがデータベースクラスタ内に存在している状態で起動
  - restore_command
    - アーカイブされたWALファイルを取得するためのコマンド
    - アーカイブ領域に存在する最新のWALファイルまでリカバリ
  - 最後に稼働していたデータベースクラスタ内のアーカイブされていないWALファイルをアーカイブ領域へ手動コピー

```
$ cd /usr/local/pgsql/data_crash
$ ls pg_xlog
$ ls /mnt/pg_arc/log
$ cp pg_xlog/* ../data/pg_xlog/
```

両ディレクトリのWALを比較し、アーカイブされていないWALがあればコピー

- リカバリモードで起動

```
$ pg_ctl start -w
```

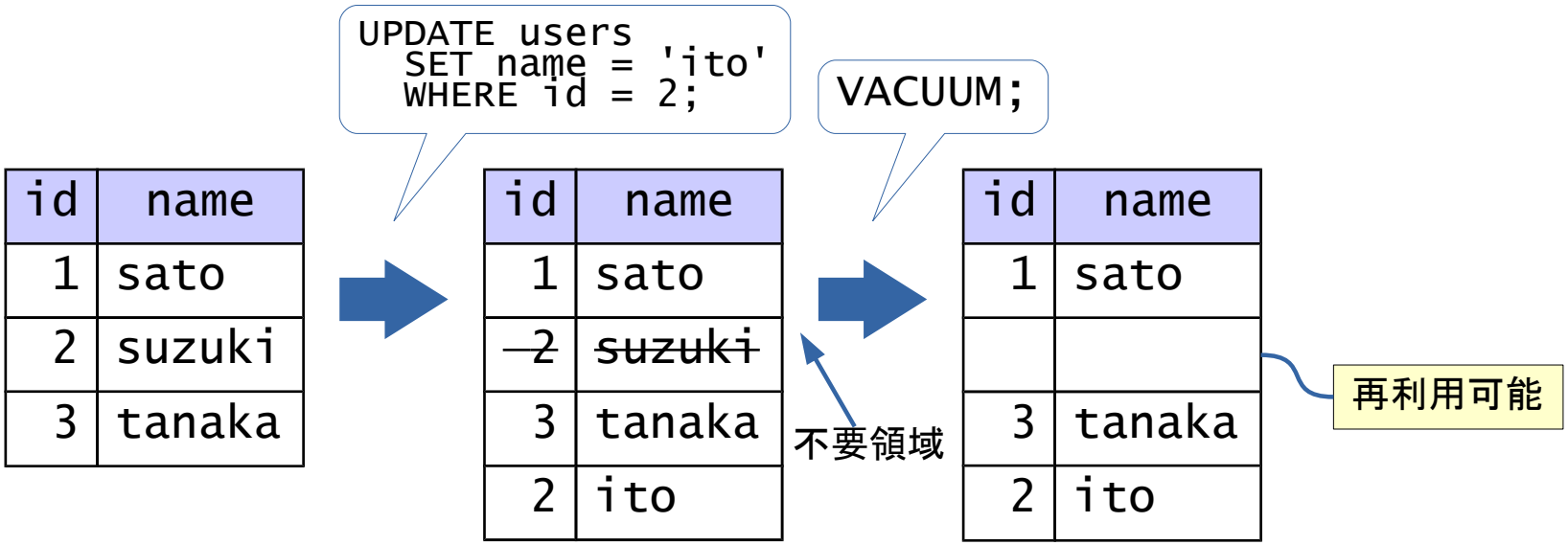
  - リカバリ中はデータベースに接続できない
    - 別端末で実行（サーバがリカバリ中は、以下の出力になる）

```
$ psql
psql: FATAL: the database system is starting up
```

- リカバリ終了
  - recovery.conf の名称が recovery.done に変わる
  - pg_log に “archive recovery complete” が出力
  - 確認用に作成した **test** テーブルが復旧

VACUUM / ANALYZE

- 不要領域の削除（ガベージコレクション）
  - 更新しても古い行は残す仕組み
  - どのトランザクションからも参照されない行は、無駄なデータなのでVACUUMコマンドで削除する
    - 削除しないと、ファイルの肥大化し、パフォーマンスが低下



- vacuumdb（PostgreSQLコマンド）

```
postgres$ vacuumdb [-t テーブル名] [データベース名]
```

- VACUUM文（SQL）

```
userdb=# VACUUM [テーブル名];
```

  - 不要となった行を探し、再利用できるようにする
    - ファイルサイズは減らない
- VACUUM FULL文
  - ファイルサイズを縮める
  - 実行中はSELECTもできなくなるので、運用中には使えない

- 統計情報の更新が必要

```
userdb=# ANALYZE [テーブル名];
```

  - インデックスを使うかどうかは、統計情報による
  - ANALYZEで統計情報を更新する
    - ANALYZEしないと最適な検索方法が使用されない
    - テーブルの内容が大幅に変わった後には、ANALYZE実行を推奨

```
postgres$ vacuumdb -az
```

すべてのデータベースで
VACUUMとANALYZEを実行する

- 自動バキューム（autovacuum）
  - postgresql.confで設定
    - デフォルトでon

```
#autovacuum = on
```

  - データベースの更新量に合わせて、適宜VACUUMやANALYZEを自動で実行
    - データベース更新量を把握するには、統計情報コレクタの機能を使用
    - その分の負荷はわずかながら余分にかかる

セキュリティ管理

pg_hba.conf

- クライアント認証
  - クライアントアプリケーションがデータベースに接続するときの認証
  - `$PGDATA/pg_hba.conf`で設定
- ロール（ユーザ）の属性
  - ロールが行う操作の権限の一部はロールの属性として管理される
- テーブル単位のアクセス権
  - テーブルを作成したユーザだけにアクセス権がある
    - GRANT文（各種権限を与える）
    - REVOKE文（各種権限を取り上げる）

クライアント認証

- $PGDATA/pg_hba.confファイルで設定
  - データベースサーバを起動時に読み込まれ、複数行ある場合は、上から評価される
  - 1行に1つの設定を記述
    - 接続タイプ、データベース名、ユーザ名、IPアドレス範囲
    - 認証方法とそのオプション
  - 接続しようとしているクライアントに関する設定がない場合、アクセスは拒否される
- pg_ctl reloadで修正を反映

```
local      dbname  user                                auth-type  [option]
host       dbname  user  IP-address  IP-mask          auth-type  [option]
hostssl    dbname  user  IP-address  IP-mask          auth-type  [option]
hostnossl  dbname  user  IP-address  IP-mask          auth-type  [option]
host       dbname  user  CIDR-address                  auth-type  [option]
hostssl    dbname  user  CIDR-address                  auth-type  [option]
hostnossl  dbname  user  CIDR-address                  auth-type  [option]
```

- 接続形式

local	ローカル接続（UNIXドメイン接続）の場合に対応
host	ホスト接続（TCP/IP）の場合に対応。ホストを指定した場合が該当する。SSL接続でも通常の接続でもどちらでもよい
hostssl	ホスト接続でSSL接続である場合に対応
hostnossl	ホスト接続でSSL接続でない場合に対応

- dbname
- user
  - カンマ区切りで複数指定可能

dbname 接続するデータベース	
all	すべてのデータベース
sameuser	接続ユーザと同じ名前のデータベース
samerole (samegroup)	接続ユーザが属しているグループと同じ名前のデータベース
@*filename*	ファイルに書いてあるデータベース

user 接続時のPostgreSQLのユーザ	
all	すべてのユーザ
+*groupname*	指定したグループに属しているメンバ
@*filename*	ファイルに書いてあるユーザ

- CIDR-address(CIDR記法によるアドレス)
  - ホスト接続のクライアントIPアドレス範囲
  - local指定のときは空欄とする
  - ホスト名での指定も可能

  - 例：ローカルホストからのTCP/IP接続のみに限定する

```
127.0.0.1/32
```

    - 代わりにIP-addressとIP-maskによる記述も可能

```
192.168.128.0  255.255.255.0
```

pg_hba.conf

SRA OSS, INC. OSS-DB

auth-type	
trust	無条件で許可
reject	無条件で拒否
md5	MD5暗号化パスワード認証
password	平文パスワード認証
gss	GSSAPI認証
sspi	SSPI認証
peer	Peer認証
ident	Ident認証
ldap	LDAP認証
radius	RADIUS認証
cert	SSLクライアント証明書認証
pam	PAM認証

※ gss は、接続形式がlocalの場合は使用できません

- データベースユーザにパスワードを与える
  - パスワードが設定されていない状態ではパスワード認証でログインできない
- パスワード設定方法
  - ユーザ作成時にいっしょにパスワードを設定する

```
$ createuser -P <一般ユーザ名>
Enter password for new role: (パスワード入力)
Enter it again: (再度パスワード入力する)
```

  - 後からパスワードを設定する

```
userdb=> \password <一般ユーザ名>
```

- host接続(TCP/IP)でuserdbデータベースならmd5で認証する例
  - 先頭行に追加する

```
# TYPE  DATABASE    USER        CIDR-ADDRESS            METHOD
host    userdb      all         0.0.0.0/0               md5
# "local" is for Unix domain socket connections only
local   all         all                                 trust
# IPv4 local connections:
host    all         all         127.0.0.1/32            trust
# IPv6 local connections:
host    all         all         ::1/128                 trust
```

- パスワード認証になるか確認

```
postgres$ pg_ctl reload
postgres$ psql -h 127.0.0.1 userDB <一般ユーザ名>
```

ユーザ権限

- ロールは（主に権限の有無を示した）属性を持つ
  - ロールの属性
    - ログイン権限　　LOGIN
    - データベース作成権限　CREATEDB
    - ロール作成権限　　CREATEROLE
    - スーパーユーザ権限 SUPERUSER
    - レプリケーション権限 REPLICATION
    - 行単位セキュリティポリシーを無視する権限
      BYPASSRLS
    - 接続数の上限　　CONNECTION LIMIT ***connlimit***
    - パスワード　PASSWORD '***password'***
    - パスワード期限　　VALID UNTIL ***'timestamp'***
    - 権限継承　　IN ROLE ***role_name***

ユーザ権限

- 属性はロール作成時に定義される
- ロール作成後も ALTER ROLE 文によって変更可能
- ロールの作成時、他のロールを指定してそのロールの権限を継承させることができる（グループとしてのロール）

```
=# CREATE ROLE group1 LOGIN;
CREATE ROLE
=# CREATE ROLE user1
   LOGIN CREATEDB CREATEROLE NOSUPERUSER
   CONNECTION LIMIT 1 PASSWORD 'pass'
   IN ROLE group1;
CREATE ROLE
=# ALTER ROLE user1 SET client_encoding TO 'EUC_JP';
=# \du
```

テーブル権限

- デフォルトでは、テーブルを作成したユーザのみすべての権限がある
- GRANT文で権限を与える、REVOKE文で権限を取り上げる

```
GRANT 権限 ON テーブル TO 対象;
REVOKE 権限 ON テーブル FROM 対象;
```

  - 権限の種類
    - 右表参照
    - ALL PRIVILEGESを指定するとすべての権限が与えられる
  - 対象
    - PUBLICを指定すると全ユーザ
    - ユーザ名かグループ名

SELECT	SELECT文・COPY TO文の実行
INSERT	INSERT文・COPY FROM文の実行
UPDATE	UPDATE文・SELECT FOR { UPDATE | SHARE }文の実行
DELETE	DELETE文の実行
TRUNCATE	TRUNCATE文の実行
REFERENCES	外部キー制約の作成
TRIGGER	トリガの作成
CREATE	スキーマ作成・オブジェクト作成
CONNECT	データベースへの接続
TEMPORARY	一時テーブル作成
EXECUTE	指定の関数・演算子の実行
USAGE	指定言語での関数作成

- 権限の確認

```
userdb=> \dp fruit
                                Access privileges
 Schema | Name  | Type  | Access privileges | Column
--------+-------+-------+-------------------+--------
 public | fruit | table |                   |
(1 row)
```

- PUBLICにSELECTの権限を与える

```
userdb=> GRANT SELECT ON fruit TO PUBLIC;
userdb=> \dp fruit
```

  - アクセス権限は、以下のようになる

```
<一般ユーザ名>=arwdDxt/<一般ユーザ名>+
=r/<一般ユーザ名>
```

r	SELECT
w	UPDATE
a	INSERT
d	DELETE
D	TRUNCATE
x	REFERENCES
t	TRIGGER
X	EXECUTE
U	USAGE
C	CREATE
c	CONNECT
T	TEMPORARY

ご清聴ありがとうございました。

■お問い合わせ■
SRA OSS, Inc. 日本支社
千田貴大
E-mail: sales@sraoss.co.jp