SRA OSS, INC.

# PostgreSQL 入門

SRA OSS, Inc. 日本支社

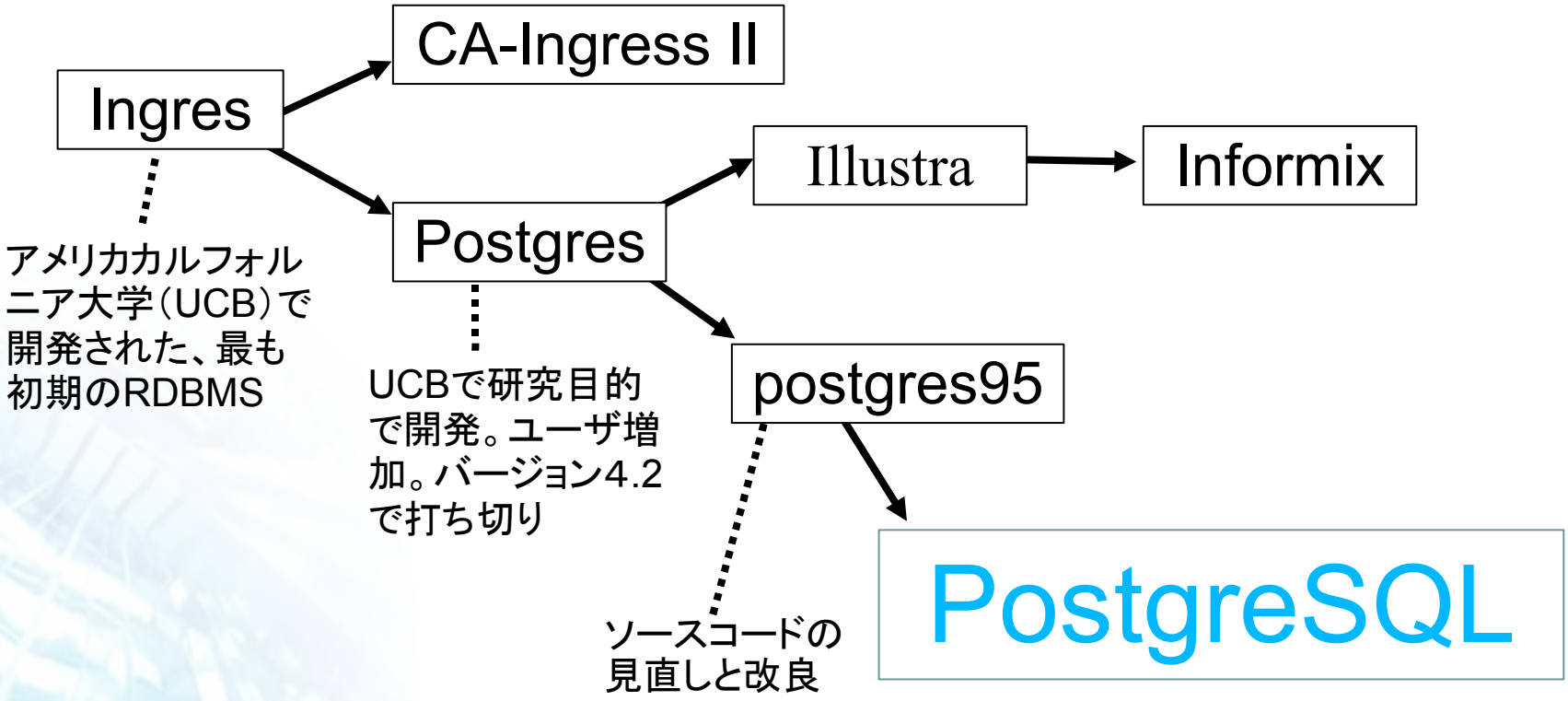
# PostgreSQL の特徴

# PostgreSQL とは

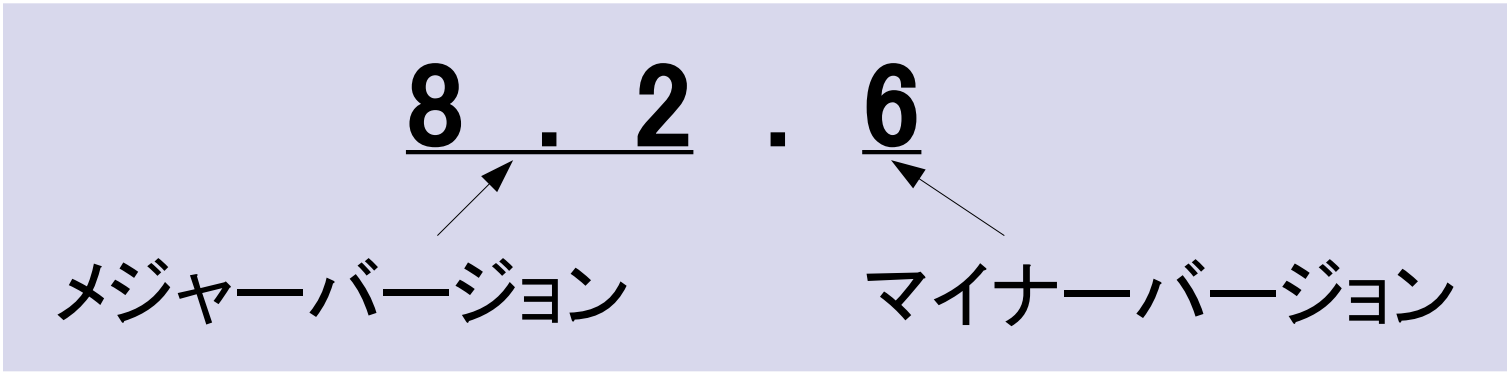
- 1996年～
- インターネットを利用したボランティア開発体制
  - 中心となる開発者は30人程度
- バージョンは 6.0 から
- http://www.postgresql.org/ （本家）

  http://www.postgresql.jp/ （日本PostgreSQLユーザ会）

# PostgreSQL の生い立ち



# バージョン番号規則




- メジャーバージョンは仕様の追加・変更がある
  - 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は主にバグ修正
- 2008年1月現在の最新バージョンは
  8.2.6 , 8.1.11 , 8.0.15 , 7.4.19

# PostgreSQL の主な特徴

- フリー＆オープンソース
  - BSDライセンス
  - 利用、コピー、配布の自由
  - 義務は著作権表示のみ
    - 広告情報は無し（いわゆる修正BSDライセンス）
  - 免責
- SQL92/99/2003 のサポート

# PostgreSQL の主な特徴

- 豊富なプラットフォームに対応
  - Linuxを含むほとんどのUNIX系システム
  - Windows
  - Mac OS X
- 本格的なDBMS

  トランザクション、オプティマイザ、マルチバイト
  サブクエリー、外部キー、手続き言語、トリガー
  トランザクションログ、テーブルスペース、PITR
  テーブル継承

# PostgreSQL の主な特徴

- Windows版はインストーラが用意されている

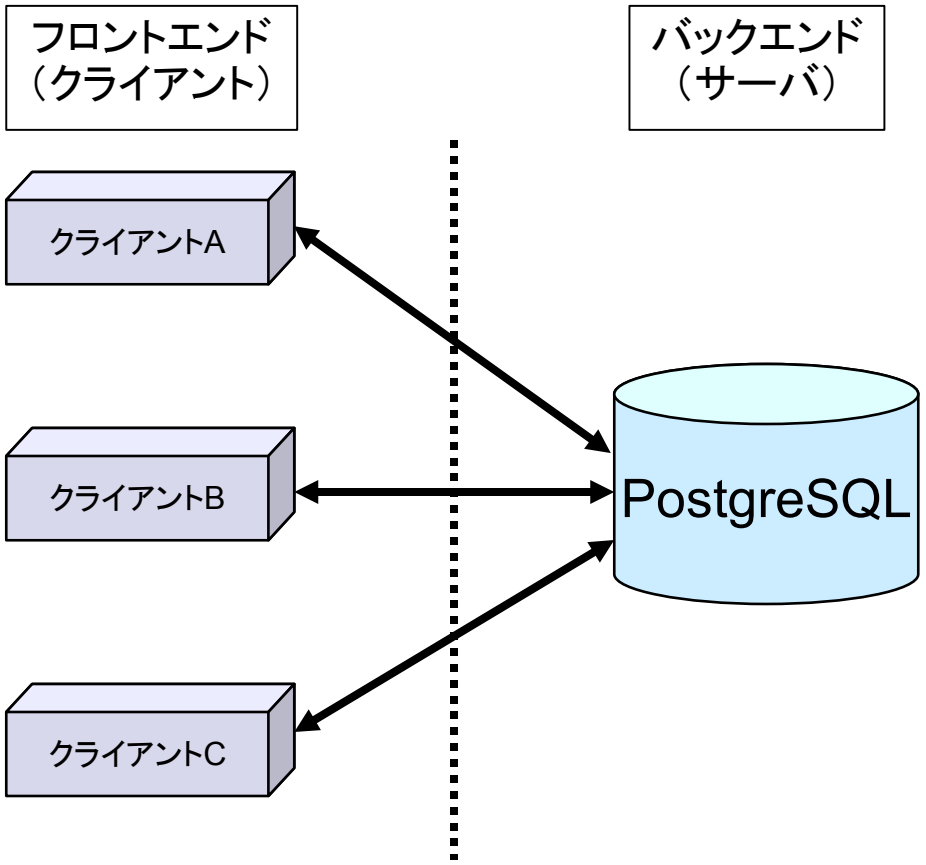
- 商用データベースと比べてコンパクト
- でも機能は勝らずとも劣らず
- 性能もそこそこ （数千万件でも問題なし）

- 無料で利用できる
- 個人マシンなどに簡単にインストールできるので

リレーショナルデータベースの学習にも最適！

# クライアント / サーバ 構成

- ネットワーク対応
- 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
- データベースエンジンの 変更に影響されにくい



# PostgreSQL に接続可能な言語

| | API |
|---|---|
| C | ○ |
| ecpg (CにSQLを埋めこむプリプロセッサ) | ○ |
| Java | ○ |
| Tcl/TK | ○ |
| Python | ○ |
| C++ | ○ |
| Perl | ○ |
| PHP | ○ |
| Ruby | ○ |
| ODBC | ○ |
| .Net Data Provider | ○ |
| VisualWorks Smalltalk | ○ |

# マルチバイト対応

- 文字エンコーディング
  - データベースごとに指定
  - クライアントごとに通信するエンコーディングを指定可能
    - 自動的に変換することで

    EUC_JP のデータベースに SJIS クライアントで通信したり

    UNICODE のデータベースに EUC_JP で通信することができる
  - 日本語では以下の組み合わせが可能

| バックエンド<br>（データベース） | フロントエンド<br>（クライアント） |
|---|---|
| EUC_JP | EUC_JP、SJIS、UNICODE |
| UNICODE （UTF-8） | EUC_JP、SJIS、UNICODE |

# PostgreSQL を動かす

# Linux の場合

# PostgreSQL のソースコード

- 本家、もしくは JPUG ホームページからダウンロード
- サイズ目安
  - tar.bz2 アーカイブ 12MB
  - アーカイブを展開して 130MB

  コンパイル前

  - プログラム本体 13MB
  - オンラインマニュアル 9MB
  - ライブラリ＋ヘッダファイル 9MB

  コンパイル後

# コンパイルとインストール

**./configure → make → make install**

- 最近の Linux ディストリビューションであれば特別な手順はなくインストールできる
  - ソースコードからインストールする場合は、RPMパッケージと競合しないように注意

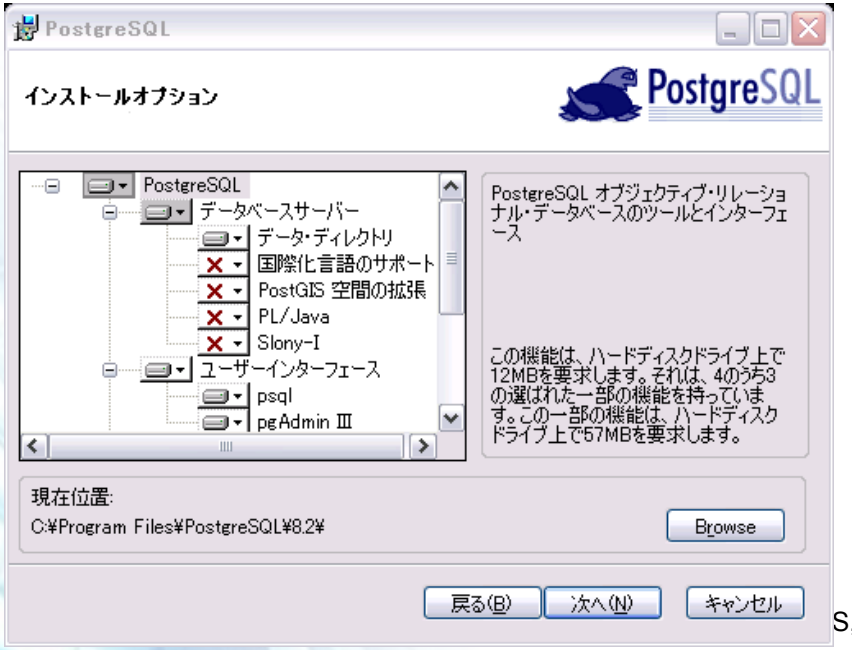
./configure 時にインストール先などを指定できる

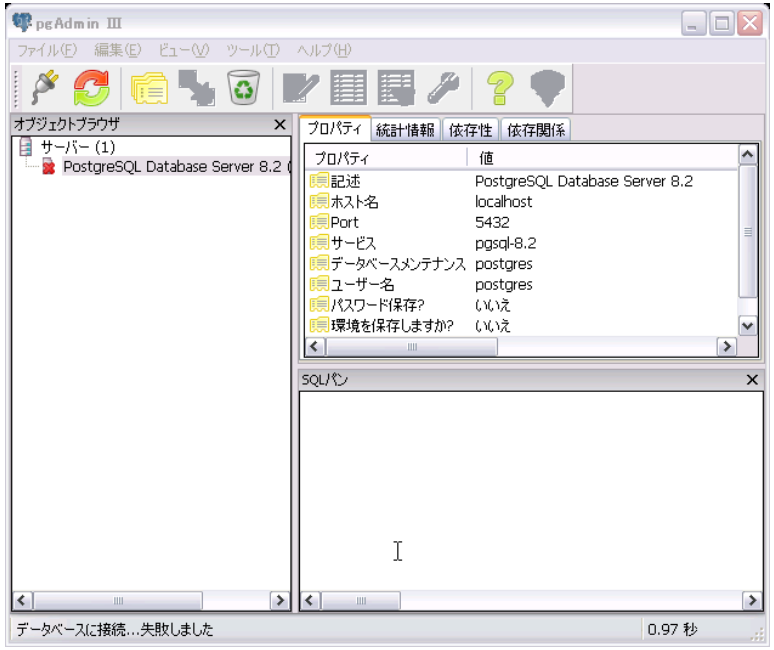
./configure --help でヘルプ表示

# Windows の場合

# Windows版バイナリ

- 日本語版インストーラから一発インストール
- サービスの登録、管理ユーザ作成なども自動化
- データベースを操作する GUI ツールも付属



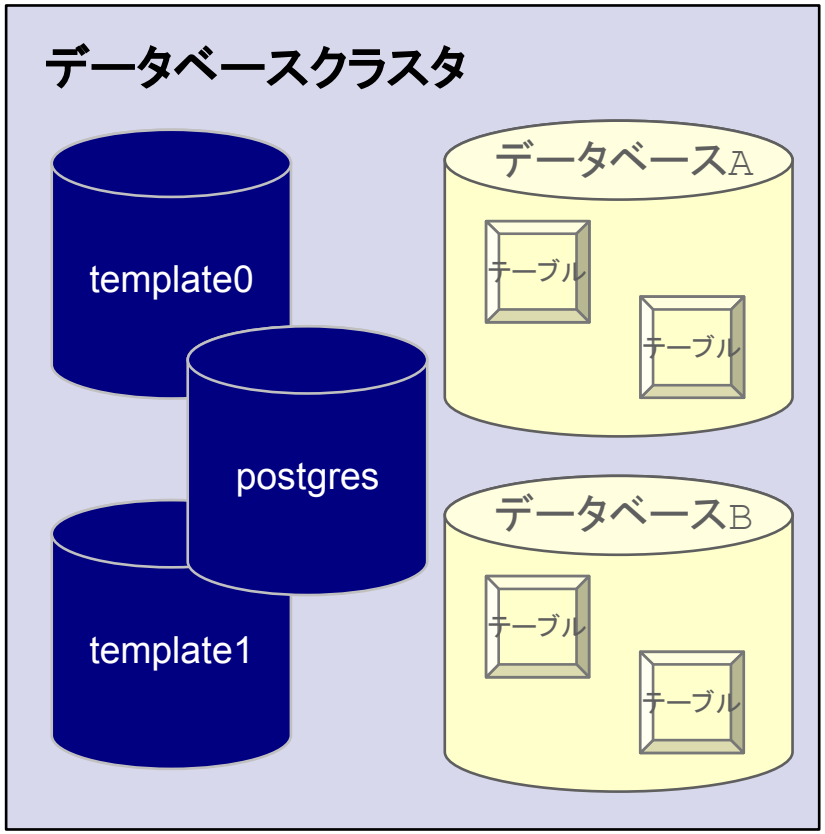
# Linux / Windows 共通

# インストールされるコマンド

| | | | |
|---|---|---|---|
| **createdb** | データベース作成 | **pg_config** | インストール情報 |
| **dropdb** | データベース抹消 | **pg_ctl** | PostgreSQLの起動 |
| **createuser** | ユーザ登録 | **pg_dump** | データベースのバックアップ |
| **dropuser** | ユーザ抹消 | **pg_dumpall** | データベース全体の<br>バックアップ |
| **createlang** | プログラミング言語追加 | **pg_restore** | データベース復旧 |
| **droplang** | プログラミング言語削除 | **psql** | SQLインタプリタ |
| **initdb** | データベースクラスタ初期化 | **vacuumdb** | データベースの<br>ガベージコレクション |

# データベースクラスタ

- PostgreSQLのデータが全て格納されるディレクトリ
- **initdb** コマンドで最初の1回だけ実行
- システムデータベースがデフォルトで作成される
- 環境変数 $PGDATA で指定
  - 一般的に /usr/local/pgsql/data



# 起動・停止

- pg_ctl コマンド

root では実行できない

- 起動

```
pg_ctl -D $PGDATA -w start
```

- 再起動

```
pg_ctl -D $PGDATA restart
```

- 停止

```
pg_ctl -D $PGDATA -m f stop
```

**-w**
wait の略。処理が完了したらプロンプトを戻す。停止の時は、デフォルトで有効になっている。

**-m** { *s* | *f* | *i* }
modeの略。
smart　全てのクライアントが切断するまで待つ（デフォルト）
fast　クライアントが切断するまで待たない）
immediate
クリーンアップ処理なしで、全サーバプロセスを中断

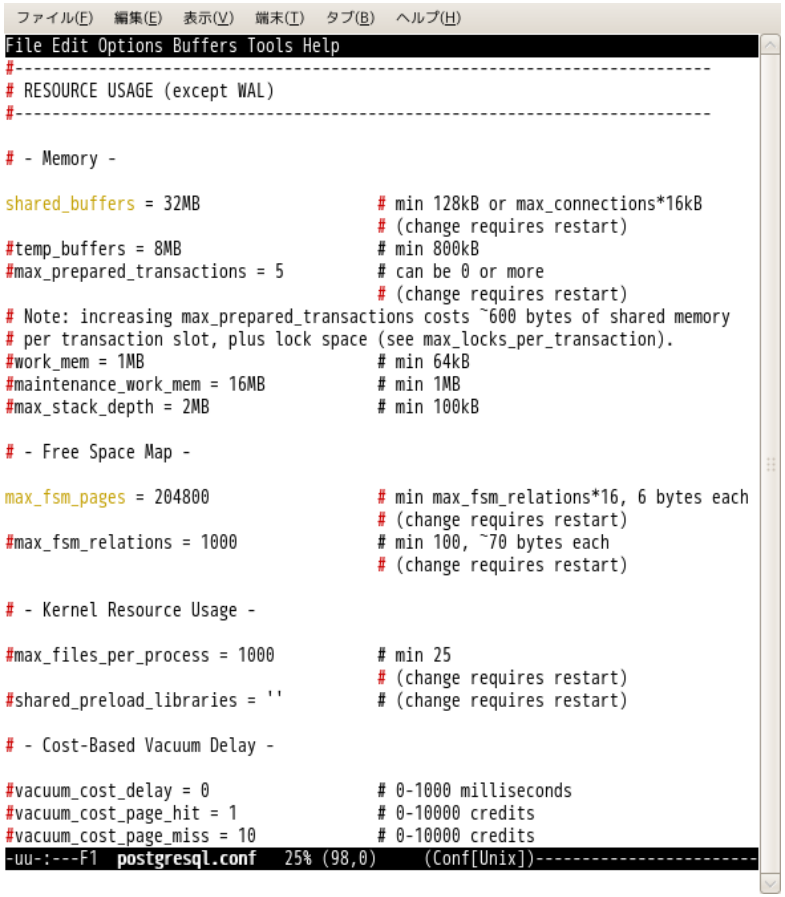
# 設定ファイル

## postgresql.conf

- $PGDATA 配下にある
- 書式
  - 「#」で始まる行はコメント
  - *パラメータ = 値*
- 変更を反映させるためには

```
pg_ctl -D $PGDATA restart
```

```
pg_ctl -D $PGDATA reload
```



# 設定ファイル　主なパラメータ

| | |
|---|---|
| **listen_addresses** | 接続を受け付けるIPアドレスを記述。*なら全てのIPインターフェイスで受付。空ならUNIXドメイン接続のみ。 |
| **max_connections** | データベースサーバへの同時接続の最大数。 |
| **silent_mode** | trueの場合、サーバはバックグラウンドで起動し、制御端末は切り離される。 |
| **port** | 接続ポート番号。デフォルトは5432。 |
| **shared_buffers** | 共有メモリバッファをページ数で指定。デフォルトは1000。 |
| **max_files_per_process** | プロセスあたりのファイルオープン最大数。 |
| **log_destination** | ログの出力先を指定。stderr、syslog、eventlog |
| **log_connections** | クライアントが接続したことをログに取るようにする。 |
| **redirect_stderr** | stderrに出力したエラーを $PGDATA/pg_log 以下のローテーションするログファイルにリダイレクトする。 |
| **client_encoding** | クライアント側文字エンコーディングのデフォルトを指定。 |

# PostgreSQL の設定

- PostgreSQL の動作、リソースに関しては全て postgresql.conf で設定する
- インストール直後（デフォルトの設定）の postgresql.conf は非力なマシンでも動作するように設定されている
- PostgreSQL の性能を引き出すためには、個々のマシンに沿った設定が必要！

# PostgreSQL を使う

# ユーザ作成

**createuser** コマンド

- PostgreSQL 8.1 からは「ロール」でユーザとグループを管理する
- SQL の「CREATE ROLE」文も利用可能

```
[postgres]$ createuser taro
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
CREATE ROLE
```

# データベース作成

**createdb** コマンド

```
[postgres]$ createdb データベース名
CREATE DATABASE
```

- データベースクラスタ（$PGDATA）には複数のデータベースを作成できる
- データベースごとに所有者を決められる
- SQL の「CREATE DATABASE」文も利用可能

# アンインストール
# データベース / ユーザ削除

- アンインストールはデータベースクラスタとインストールディレクトリを削除するだけ

```
[postgres]$ rm -rf $PGDATA
[postgres]$ rm -rf /usr/local/pgsql
```

- データベース削除
  - **dropdb** コマンド
  - DROP DATABASE 文
- ユーザ削除
  - **dropuser** コマンド
  - DROP ROLE 文

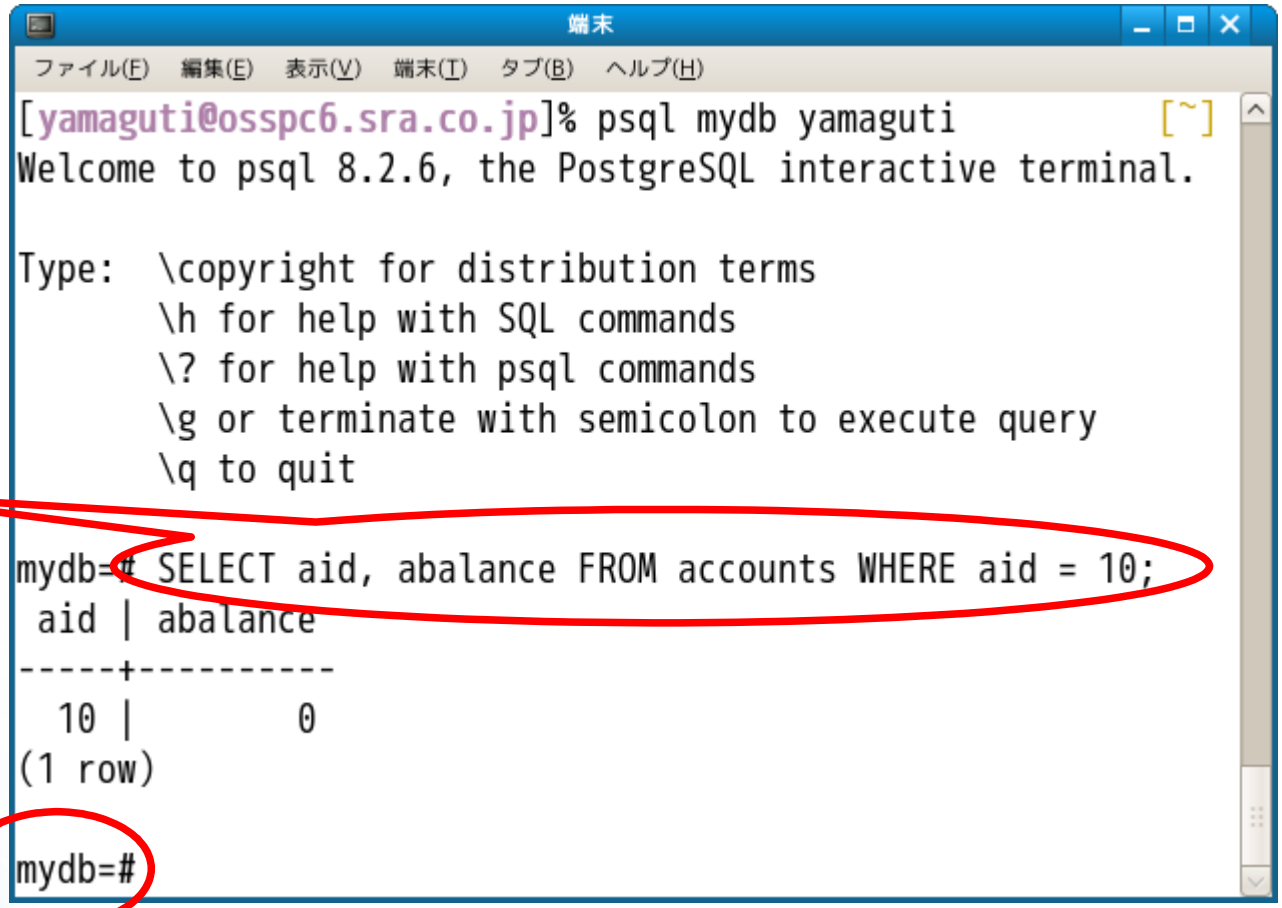
# SQL の実行

**psql** コマンド

```
psql ［オプション］..　［DB名 ［ユーザ名］］
```

- 対話型 SQL インタプリタ （クライアントアプリ）
- ユーザが入力した SQL をデータベースサーバ（バックエンド）に接続して送信する
- SQL の処理結果を受け取りユーザに表示する

# psql スクリーンショット




SQL 入力

バックエンドからの結果表示

接続中のデータベース

# SQL

- 標準SQL に準拠しているので、通常の SQL は問題なく使える
  - SELECT, UPDATE, INSERT, DELETE
    CREATE TABLE, JOIN, OUTER JOIN
    インデックス、ビュー、外部キー、スキーマ
    サブクエリー、トリガ、シーケンス、カーソル ...
- 一部方言、独自拡張がある

# データ型

- 1レコードの1カラムあたり、1GBまで
- 豊富な組み込みデータ型
- ユーザがデータ型を作れる（ユーザ定義データ型）

- 主な組み込みデータ型
  - 数値
  - 通貨
  - 文字
  - 日付
  - 真偽
  - 幾何
  - ネットワークアドレス
  - ビット列
  - ラージオブジェクト
  - 配列

# ログ

- 独自のログ機能で取得、ローテーションなどを行える
- postgresql.conf で設定
  - ログ出力先
  - ログ出力内容
    - 時刻、PID、SQL文、ユーザ名、データベース名など
  - ローテーション サイズ / 時間

syslog にも出力可能

```
redirect_stderr = on
```

ログが $PGDATA/pg_log/
以下に出力される

# バックアップ / リストア

- **pg_dump** コマンド
  - データベース単位でバックアップを取得
- **pg_dumpall** コマンド
  - データベースクラスタ全体のバックアップを取得
- デフォルトは SQL 文で論理バックアップを作成する
- データベースを停止する必要がない

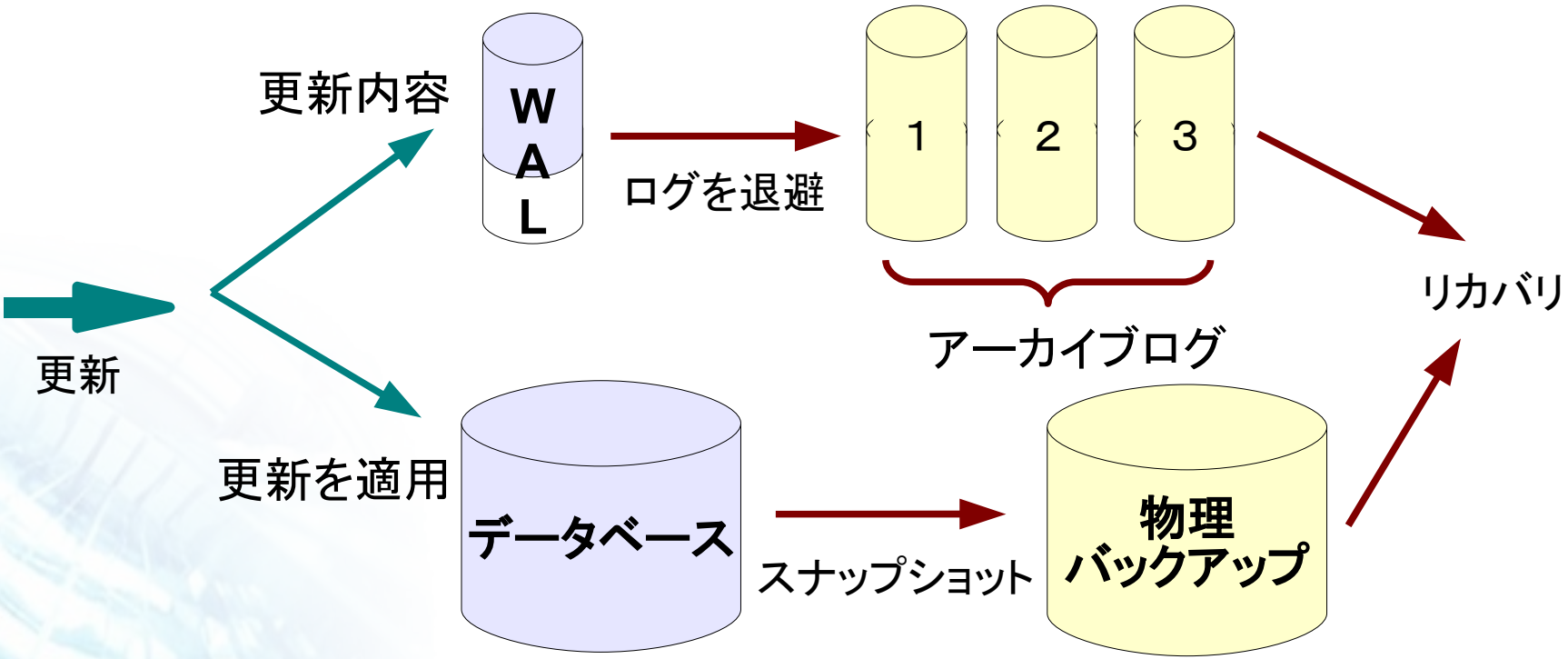
- リストアは **psql** を使い、SQL 文から復元する
- ポイント・イン・タイム・リカバリ（PITR）機能もある

# PostgreSQL の機能

# PITR

- バックアップとトランザクションログを使って…
  - データベースを最新状態へリカバリ
    - ディスク障害などへの対応
  - 特定の時点へデータベースをロールバック
    - データを誤って削除してしまった場合などへの対応

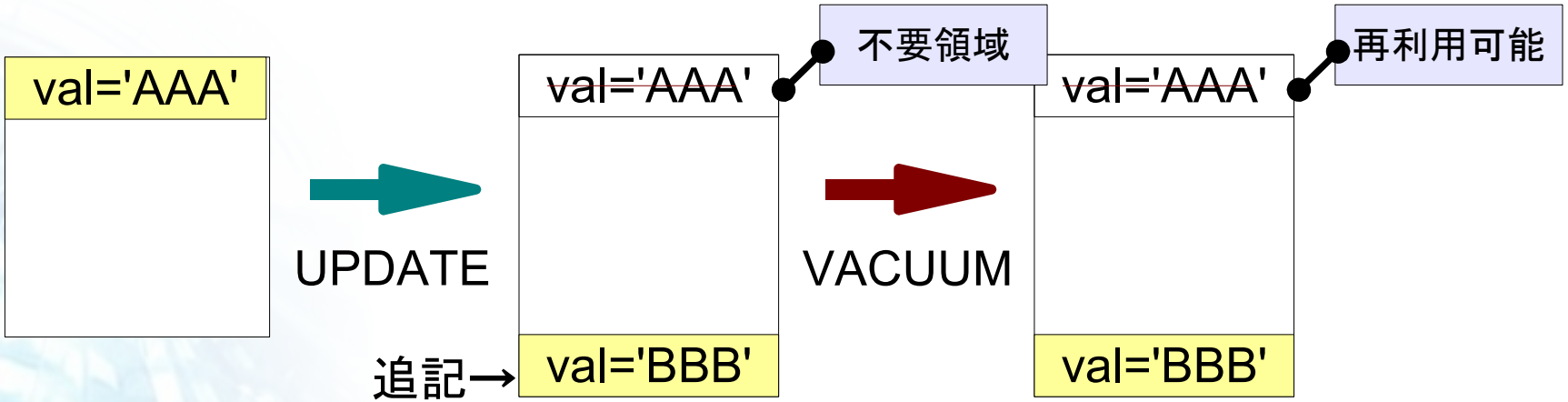
# PITR の仕組み



# VACUUM

- データベースのガベージコレクション
- PostgreSQL は追記型アーキテクチャ

```
UPDATE tbl SET val = 'BBB' WHERE val = 'AAA';
```



# VACUUM の種類

- **vacuumdb** コマンド
- VACUUM
  - 運用を妨げずに並行して実行可能
  - データベースのサイズは基本的に変わらない
    - 再利用されるだけで、ファイルサイズが小さくなるわけではない
- VACUUM FULL （**vacuumdb -f**）
  - 実行中はテーブルを完全にロックしてしまう
  - ファイルの再利用可能領域を詰めて、サイズを小さくする

# VACUUM はいつ実行すればよい？

- 全データベースに対して最低1日に1回
- 更新頻度によっては、個別テーブルに対して頻度を上げる
- VACUUM FULL は通常使わなくてよい

# セキュリティ

- ホストによる認証
  - pg_hba.conf 設定ファイル
  - パスワード認証、ident認証、pam認証などを、IPアドレス、ユーザ名、データベース名ごとに設定できる
- ユーザの権限
  - テーブル、ビュー、シーケンスといったオブジェクトに、検索、更新、削除などの権限を設定できる
  - GRANT / REVOKE 文

# その他の機能

- テーブルスペース
  - データベースクラスタとは別のディスクにテーブルを構築できる
- PL/pgSQL
  - SQL 手続き言語でユーザが関数を定義できる
- 二相コミット
  - 分散されたトランザクション間での同期をとることができる
- テーブル継承
  - 親テーブルを継承する子テーブルを複数作成し、データをクラスタ化できる

# PostgreSQL を使ってみよう！

ご清聴ありがとうございました