

PostgreSQL 初心者向けセミナー

日本 PostgreSQL ユーザ会 沖縄支部
屋比久 友秀



PostgreSQL の特徴



PostgreSQL とは

- 1996年から正式に PostgreSQLとなる
 - Ingress カリフォルニア大学(UCB)
 - Postgres UCB Ver4.2
 - Postgre95 ~1995年まで
- インターネットを利用したボランティアでの開発体制
 - 中心となる開発者は、全世界に30人程度
- <http://www.postgresql.org/> (本家)
- <http://www.postgresql.jp/> (日本 PostgreSQL ユーザ会)



日本 PostgreSQL ユーザ会

The screenshot shows a web browser window with the title "NPO 法人 日本PostgreSQLユーザ会 - SeaMonkey". The address bar contains "http://www.postgresql.jp/". The website header includes the PostgreSQL logo and navigation links: "サイトマップ", "アクセシビリティ", "連絡フォーム", and a search box. A main navigation bar lists: "ホーム", "ユーザ会", "支部", "イベント", "分科会", "ダウンロード", "広報Blog", and "ニュース".

The main content area features a large banner for PostgreSQL 8.3 with the text: "PostgreSQL 8.3 もっと便利に、そして、さらに速く". Below this, it states: "最早「速い」だけではない。全文検索、XMLデータ型など多くの機能追加、性能、運用性も向上したPostgreSQL 8.3を是非お試し下さい。"

On the left, there is a section titled "日本PostgreSQLユーザ会" with a description: "日本PostgreSQLユーザ会は、PostgreSQLの普及促進を目的に活動する特定非営利活動(NPO)法人です。" followed by a list of items: "本会の目的", "協賛会員", "理事会 / 議事録", "会員 / 入会申し込み", and "メーリングリスト".

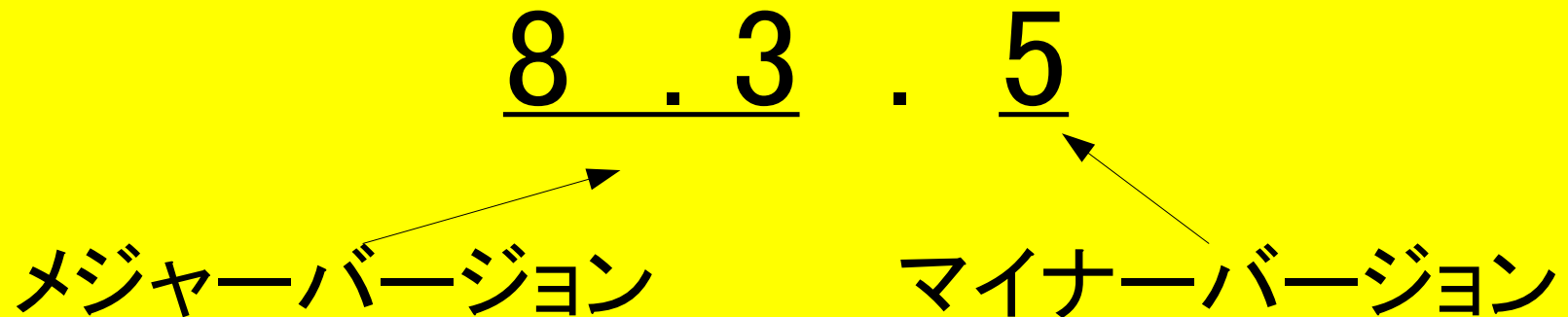
In the center, there is a section titled "PostgreSQL" with the text: "PostgreSQLはフリー（料金、ライセンス、二次配布）で利用できるオープンソースのオブジェクトリレーショナルデータベース管理システム (ORDBMS) です。" and "PostgreSQL Global Development Groupにより開発されています。" Below this is a "最新バージョン" section listing "8.3.3(8.3シリーズの最新版)" and "8.2.9(8.2シリーズの最新版)".

On the right, there is a "ニュース" (News) section with several entries: "8.3.3 日本語ドキュメント公開" (2008年06月17日), "「オープンセミナー 2008@四国」開催のお知らせ" (2008年06月10日), "PostgreSQL Conference 2008 開催のお知らせ" (2008年05月10日), "Jus研究会 JAPAN TOUR 2007-2008 仙台大会" (2008年04月02日), and "オープンソースカンファレンス2008 Tokyo/Spring にJPUGが参加します" (2008年02月20日). A "次のニュース" link is also present.

The browser's taskbar at the bottom shows various icons including the Start button, Internet Explorer, and other applications.



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8月現在の最新バージョンは
 - 8.3.5, 8.2.11, 8.1.15, 8.0.19, 7.4.23

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
 - SQL文法の充実度が高い



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（インストーラ有り）
 - 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

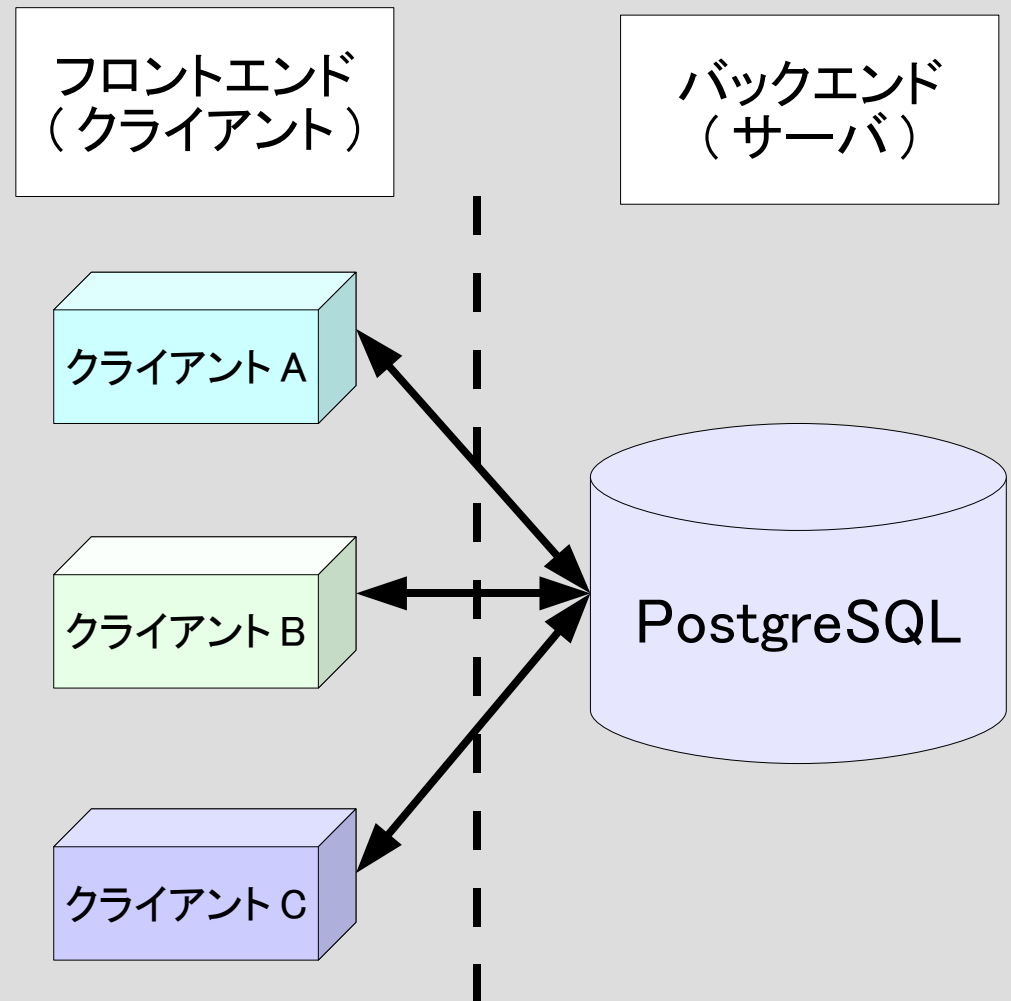
- 商用データベースと比べてコンパクト
- 機能でも勝らずとも劣らず
- 他のOSS-DBと比較しても性能に遜色はない
 - テーブル上のレコード数が数千万件でも問題なし
- 無料で利用できる
- 個人マシンなどに簡単にインストールできる

**リレーショナルデータベースの学習に最適！
ビジネスの現場でも十分に利用可！**



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ができる
 - クライアントごとに通信するエンコーディングを指定することができる
- 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
- JIS X 0213 (JIS 2004) にも対応

PostgreSQL を動かす



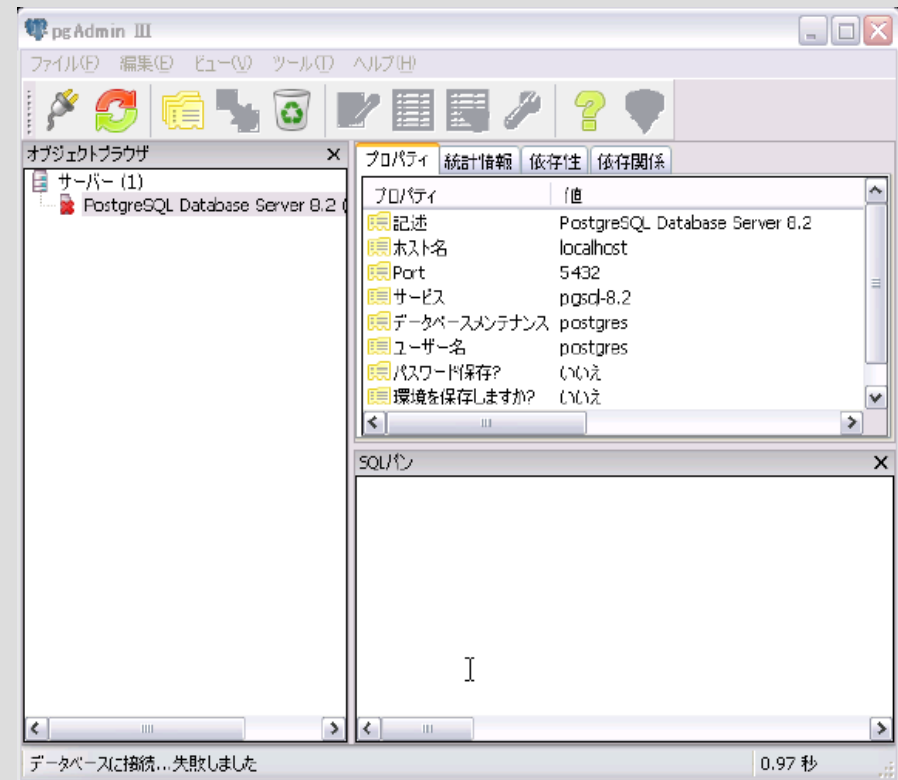
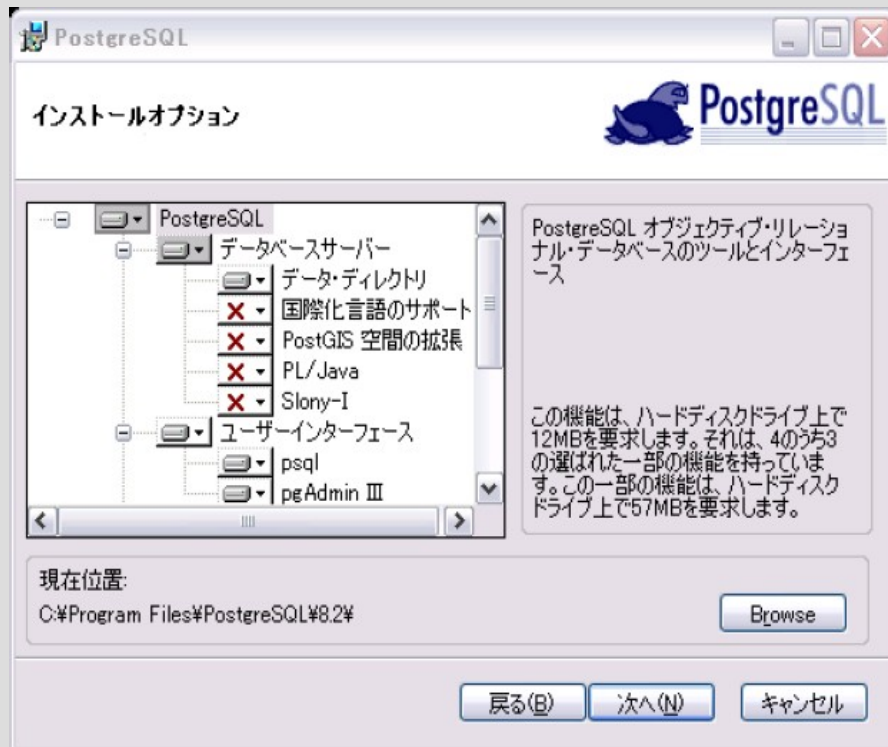
PostgreSQL を動かす (Linux)

- ディストリビューションパッケージから
 - 動作が比較的安定している
 - ただし最新版では無い (Ver 8.1, 8.2)
 - Debian 系 : apt-get, aptitude
 - RedHat 系 : rpm, yum
 - 初期インストールでは、DB クラスターの DIR が決められている
- ソースコードからコンパイルしてインストール
 - ./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
 - ソースからインストールは、他の RPM パッケージと競合しないよう注意が必要
 - ./configure する際に、パラメータでインストール先などを指定できる
 - ./configure --help でヘルプが表示



PostgreSQL を動かす (Windows)

- 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



PostgreSQL を動かす インストールされるコマンド

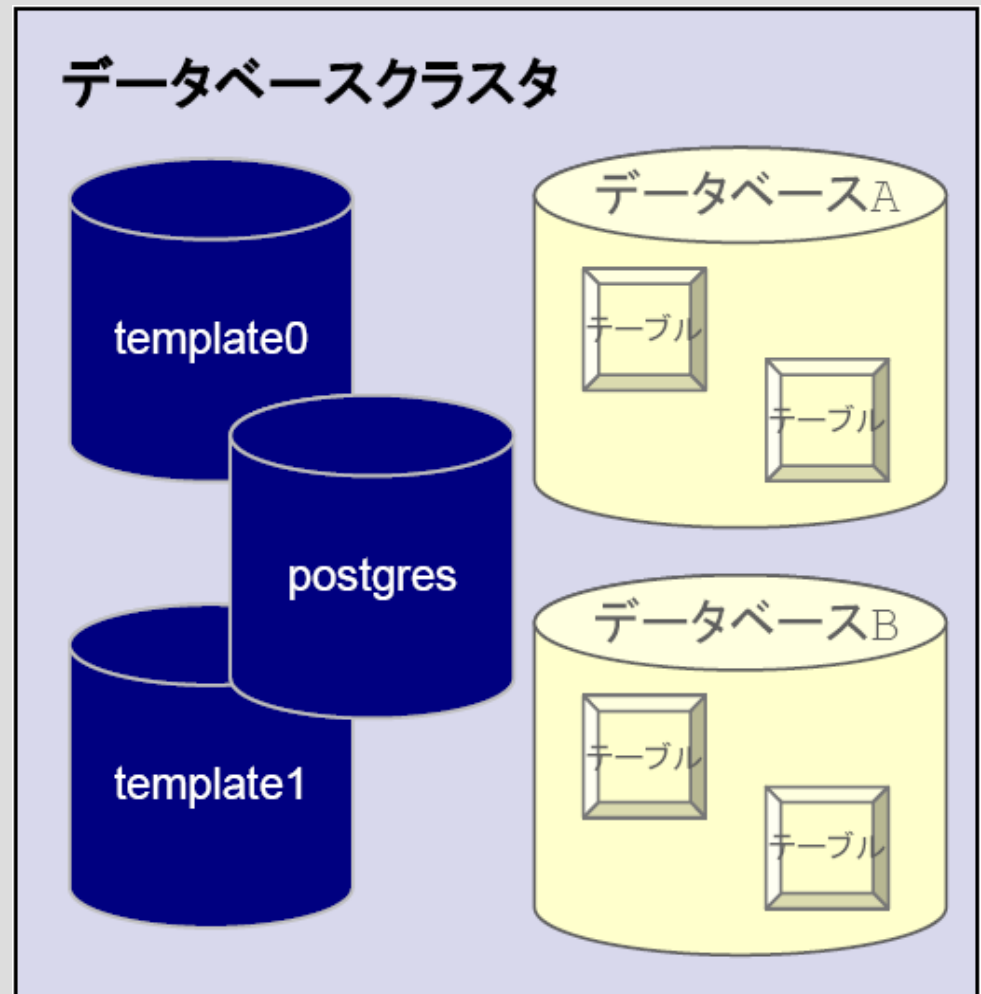
createdb	データベース作成	pg_config	インストール情報
dropdb	データベース抹消	pg_ctl	PostgreSQLの起動
createuser	ユーザ登録	pg_dump	データベースのバックアップ
dropuser	ユーザ抹消	pg_dumpall	データベース全体のバックアップ
createlang	プログラミング言語追加	pg_restore	データベース復旧
droplang	プログラミング言語削除	psql	SQLインタプリタ
initdb	データベースクラスタ初期化	vacuumdb	データベースのガベージコレクション



データベースクラスタ

PostgreSQL データ全てが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
- 教科書的には、
 - `/usr/local/pgsql/data`



起動・停止

root で実行できない

pg_ctl コマンド

- 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**
- 設定を反映させる為には

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

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

```
ファイル(F) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
File Edit Options Buffers Tools Help
#-----
# RESOURCE USAGE (except WAL)
#-----
# - Memory -
shared_buffers = 32MB          # min 128kB or max_connections*16kB
                                # (change requires restart)
#temp_buffers = 8MB           # min 800kB
#max_prepared_transactions = 5 # can be 0 or more
                                # (change requires restart)
# Note: increasing max_prepared_transactions costs ~600 bytes of shared memory
# per transaction slot, plus lock space (see max_locks_per_transaction).
#work_mem = 1MB                # min 64kB
#maintenance_work_mem = 16MB  # min 1MB
#max_stack_depth = 2MB         # min 100kB
# - Free Space Map -
max_fsm_pages = 204800         # min max_fsm_relations*16, 6 bytes each
                                # (change requires restart)
#max_fsm_relations = 1000      # min 100, ~70 bytes each
                                # (change requires restart)
# - Kernel Resource Usage -
#max_files_per_process = 1000  # min 25
                                # (change requires restart)
#shared_preload_libraries = '' # (change requires restart)
# - Cost-Based Vacuum Delay -
#vacuum_cost_delay = 0         # 0-1000 milliseconds
#vacuum_cost_page_hit = 1     # 0-10000 credits
#vacuum_cost_page_miss = 10   # 0-10000 credits
-uu-:---F1 postgresql.conf 25% (98,0) (Conf[Unix])-----
```



設定ファイル 主なパラメータ

<code>listen_addresses</code>	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。
<code>max_connections</code>	データベースサーバへの同時接続の最大数。
<code>silent_mode</code>	<code>true</code> 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。
<code>port</code>	接続ポート番号。デフォルトは5432。
<code>shared_buffers</code>	共有メモリバッファをページ数で指定。デフォルトは1000。
<code>max_files_per_process</code>	プロセスあたりのファイルオープン最大数。
<code>log_destination</code>	ログの出力先を指定。 <code>stderr</code> 、 <code>syslog</code> 、 <code>eventlog</code>
<code>log_connections</code>	クライアントが接続したことをログに取るようにする。
<code>redirect_stderr</code>	<code>stderr</code> に出力したエラーを <code>\$PGDATA/pg_log</code> 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。
<code>client_encoding</code>	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！
 - いわゆる「チューニング」！
 - JPUG沖縄支部のセミナー（1月予定）に期待！



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

- 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

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



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 (SQL文)
- ユーザ削除
 - **dropuser** コマンド
 - DROP ROLE 文 (SQL文)



SQL の実行

psql コマンド

- 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 [オプション].. [DB名 [ユーザ名]]
```



psql コマンド操作例

```
ファイル(E) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
-bash-3.2$
-bash-3.2$ psql -U squirreluser squirrelmail
Welcome to psql 8.3.3, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

squirrelmail=# select owner,nickname from address where owner='sei';
 owner | nickname
-----+-----
 sei   | php
 sei   | sysop
(2 rows)

squirrelmail=#
```

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 文

- 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



データベース型

- 1レコード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 日付 ネットワークアドレス 配列
 - 通貨 真偽 ビット列 XML
 - 文字 幾何 ラージオブジェクト



☐ 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
 - syslog にも出力可能
- postgresql.conf で設定する
 - ログ出力先
 - ログ出力内容
 - 時刻、PID、SQL文、ユーザ名、DB名など
 - ログのローテーション サイズとその時間



PostgreSQL をさらに 安心・安全に使うために

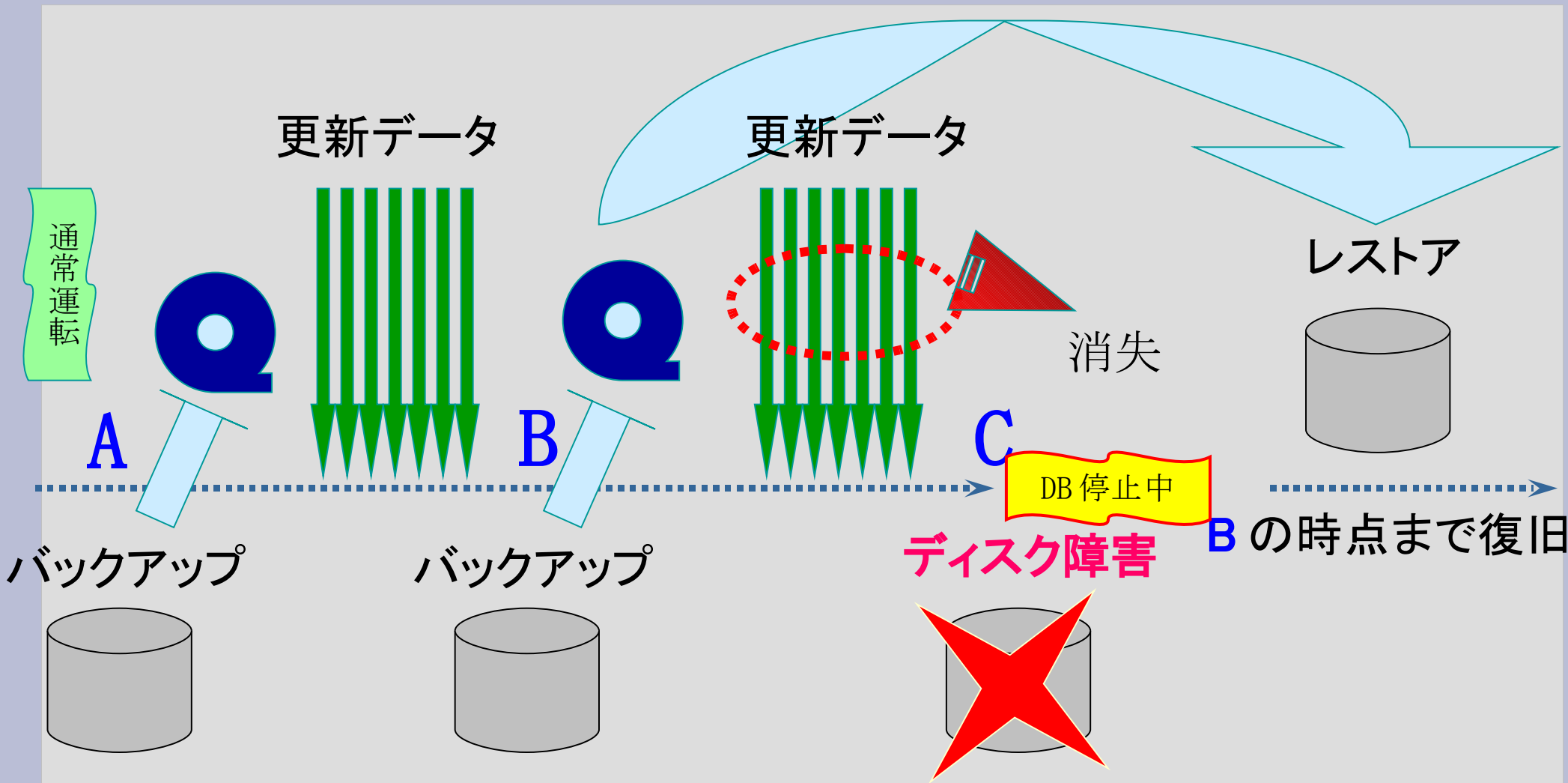


☐バックアップ / リストア

- 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
- dumpの為にデータベースを停止する必要はない
- リストアは psql を使い、SQL 文から復元する
- ディスク障害などがあり、この Dump ファイルから復旧（リストア）する場合、dumpした時点までの分しか復旧できない。



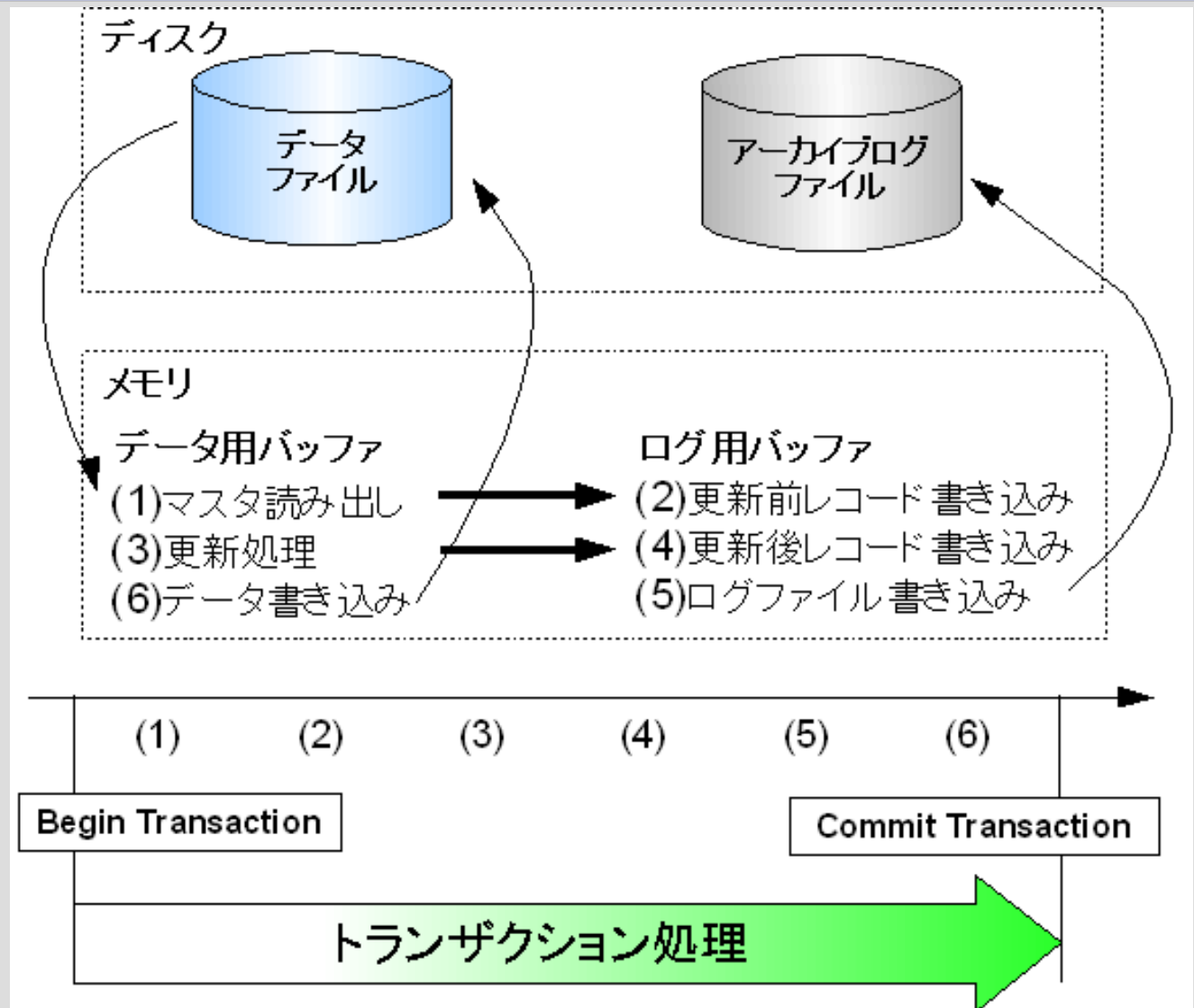
ディスク障害による更新データの消失



トランザクションについて

トランザクションとは？

アプリケーション
やユーザから見
て、「ひと纏まり」
のデータベース操
作の処理単位

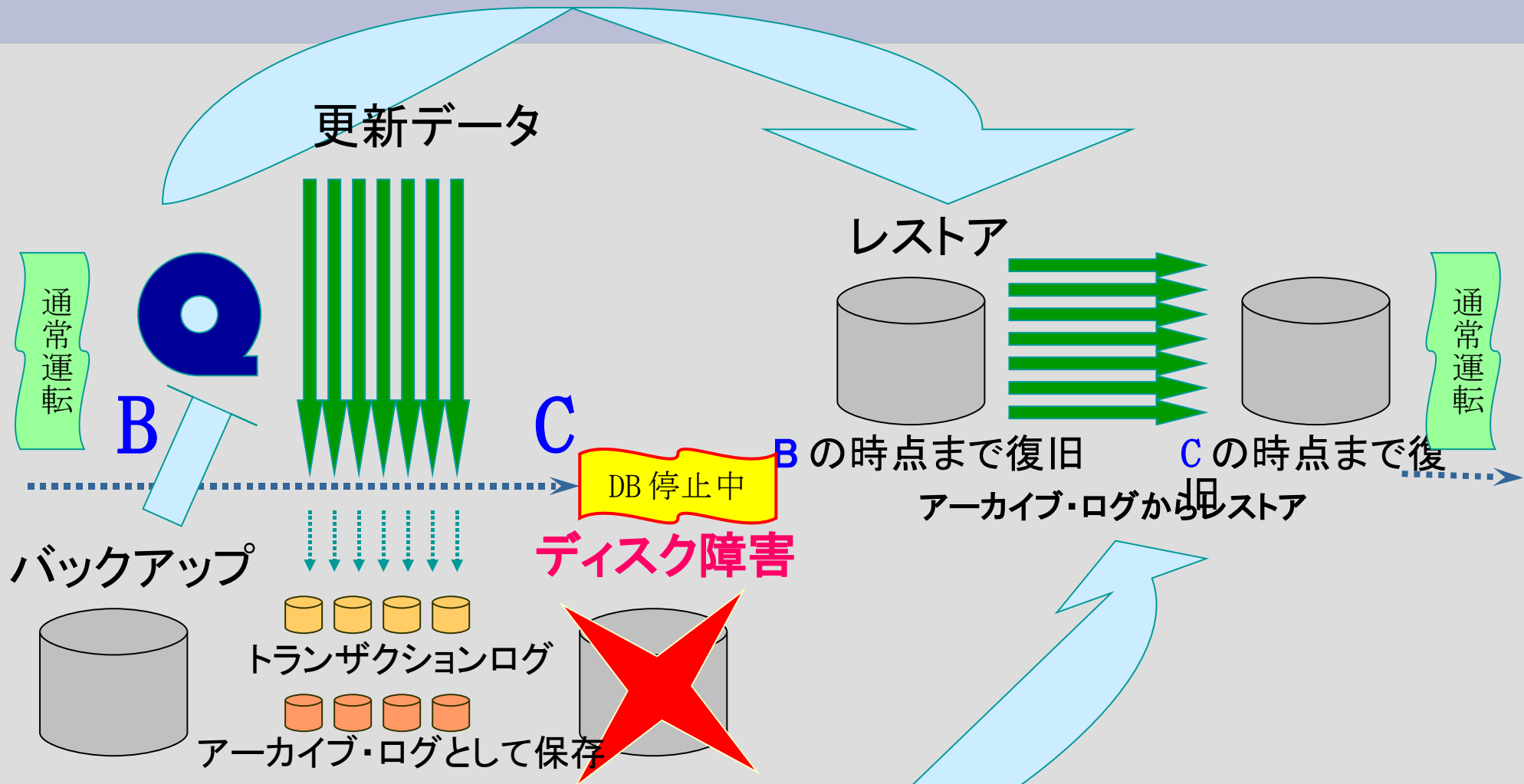


☐ データリカバリ

- ポイント・イン・タイム・リカバリ (Point In Time Recovery)
 - DB 本体がクラッシュなどで失われても、バックアップとトランザクションログから、最新の状態に復元することが可能
 - CheckPoint のタイミングで捨てていたトランザクションログ (WAL : Write-ahead Log) を、アーカイブ・ログとして保存しておく
 - DB 本体が、メディア障害によって失われるような重大な障害の際にも、障害直前の状態まで DB の復旧ができる
 - 最新の状態に復元するだけでなく、任意の時点へ戻すことも可能



PITR リカバリ動作



☐ 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

PostgreSQL を もっともっと使う



☐ 高機能いろいろ

- 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
- 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
- SQL/XML標準をサポート
 - 新しい演算子(xml関数)とXMLデータ型



PL/pgSQL

```
declare
  cu refcursor;
  rec record;
begin
  open cu for select * from hello_world_table;
  loop
    fetch cu into rec;
    if not found then
      exit;
    end if;
    raise debug '' language = % message = %'', rec.language, rec.message;
  end loop;
  close cu;
  return 0;
end;
```

language 'plpgsql'



二相コミット

銀行の口座テーブル a, b がデータベース DA, DB に作成されているものとする。
今, a から b に 1000 円送金したいとしよう。DB 内部では,

- (1) DA のテーブル a の該当レコードから 1000 円マイナス
- (2) Db のテーブル b の該当レコードに 1000 円プラス

```
DA: BEGIN;
```

```
DA: UPDATE a SET amount = amount - 1000 WHERE id = 'foo';
```

```
DB: BEGIN;
```

```
DB: UPDATE b SET amount = amount + 1000 WHERE id = 'bar';
```

PREPARE TRANSACTION を使ってトランザクションをセキユア状態に移行する。
2つの UPDATE の後にいきなりコミットするのではなく、一旦セキユア状態にする。
2つの DB に対してどちらのセキユアも成功したときのみコミットするのである。もしどちらかのセキユアが失敗したならロールバックすればよい。

```
DA: PREPARE TRANSACTION 'test1';
```

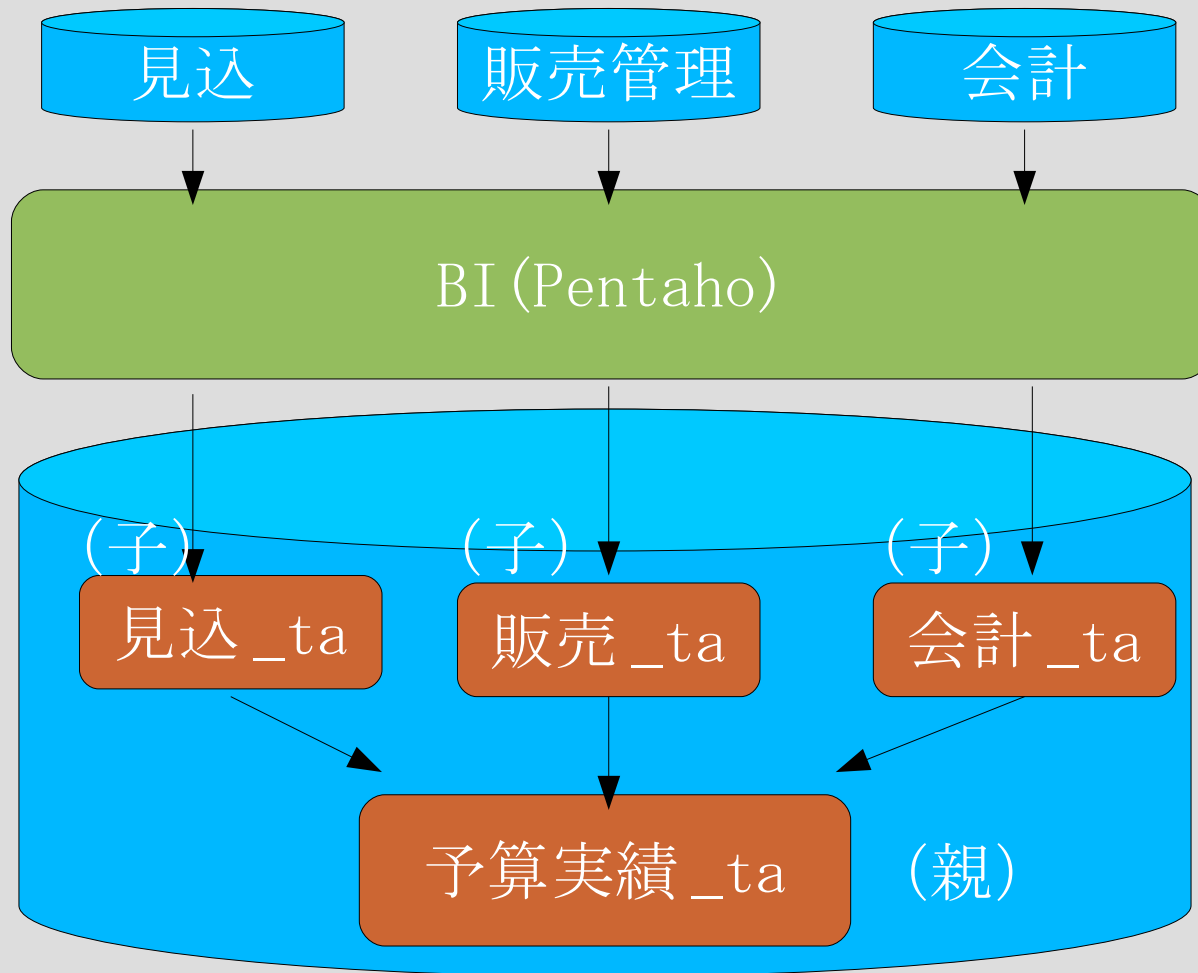
```
DB: PREPARE TRANSACTION 'test2';
```

```
DA: COMMIT PREPARED 'test1';
```

```
DB: COMMIT PREPARED 'test2';
```



テーブル継承



```
CREATE TABLE 子  
テーブル (  
...  
) inherits (親  
テーブル);
```

PostgreSQL を使ってみよう！

ご清聴ありがとうございました

