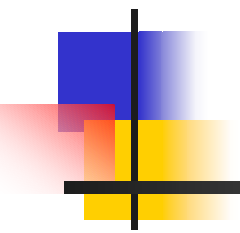
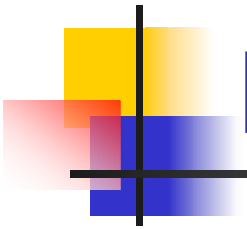


PostgreSQLによる
データベースサーバ構築技法

日本PostgreSQLユーザー会

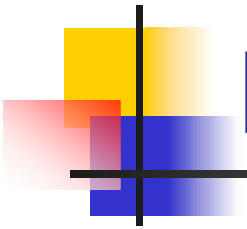
理事長

石井達夫



PostgreSQLとは

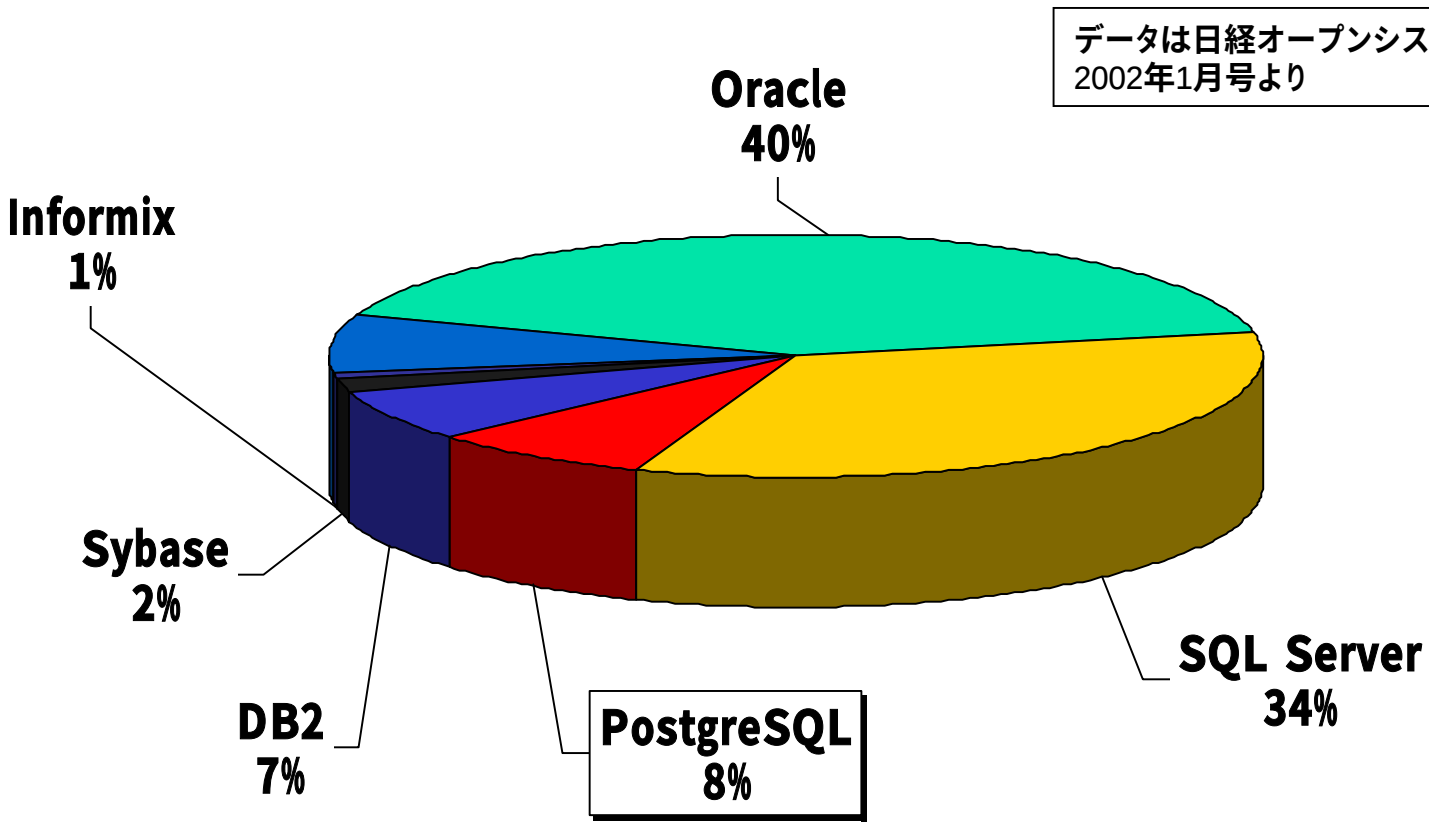
- カリフォルニア大学バークレー校(UCB)で開発
- 現在は世界中のボランティアの手によって維持
- 本格的なオープンソースデータベース
- Unix/Linux/Windowsなどで稼動
- 無償利用、自由なライセンス
- 商用サポートあり



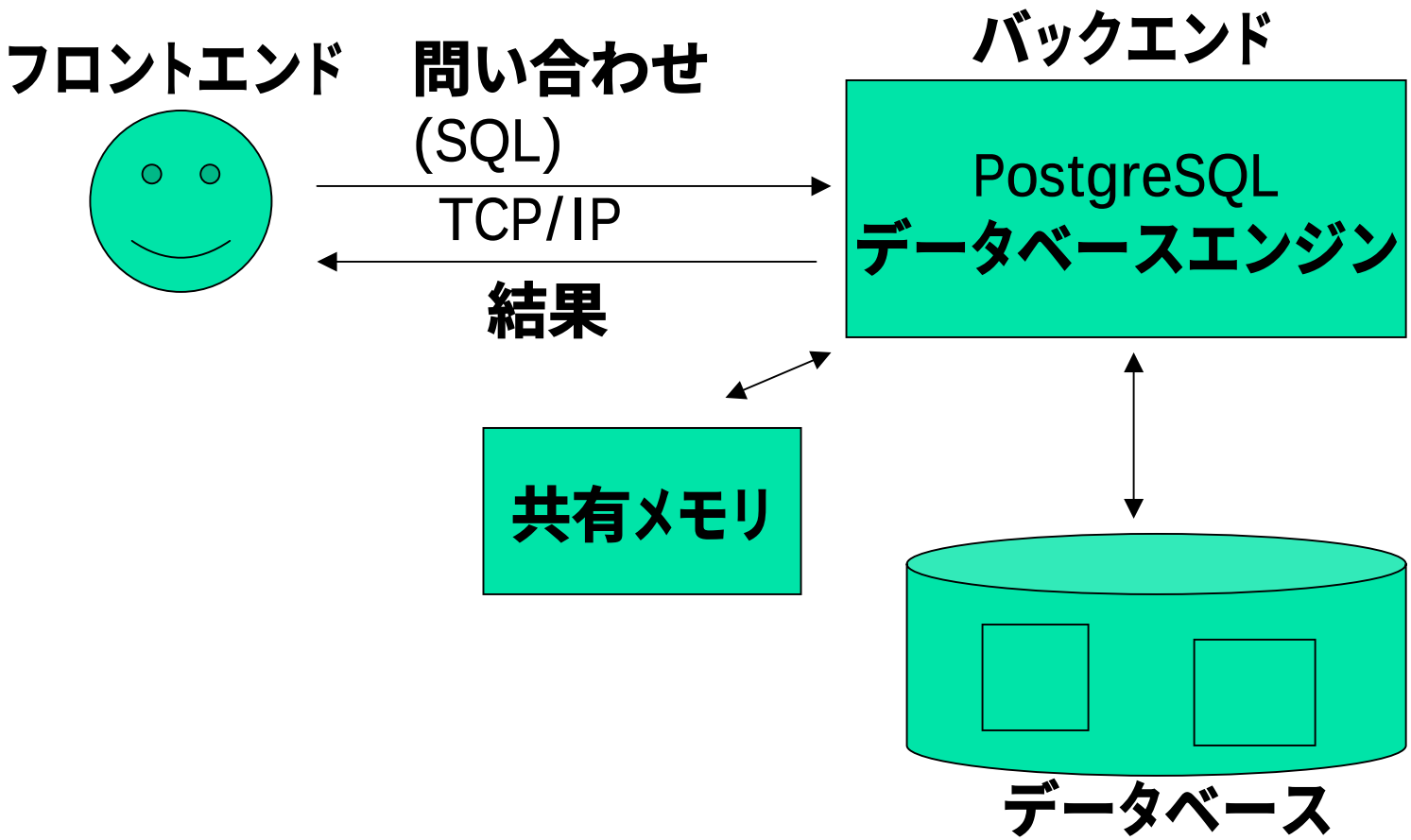
PostgreSQLの特徴

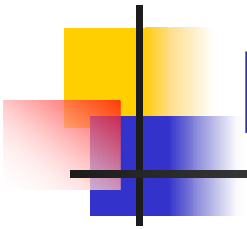
- 関係データベース＋オブジェクト指向拡張
- 強力なトランザクション管理機能
- 容易な管理
- 大規模データ・大規模ユーザの管理可能
- 多彩なAPIをサポート
  - C, C++, Java, Perl, Tcl/Tk, PHP, Ruby…

PostgreSQLの利用実績



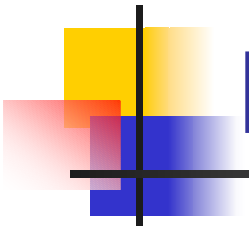
PostgreSQLの構造





PostgreSQLの機能(1)

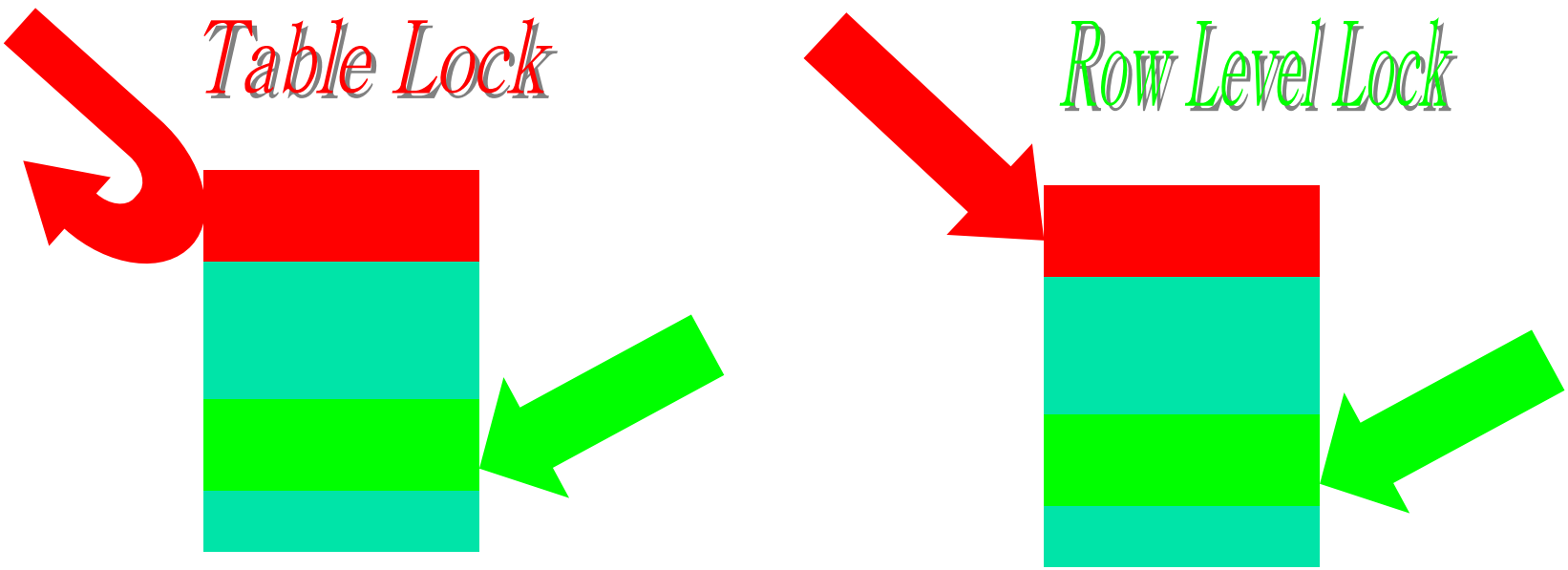
- SQL92/99のサポート
- 非標準SQL
  - トリガー
  - ストアドプロシジャ
  - シーケンス
  - 幾何データ型
  - ユーザ定義データ型
  - テーブルの継承



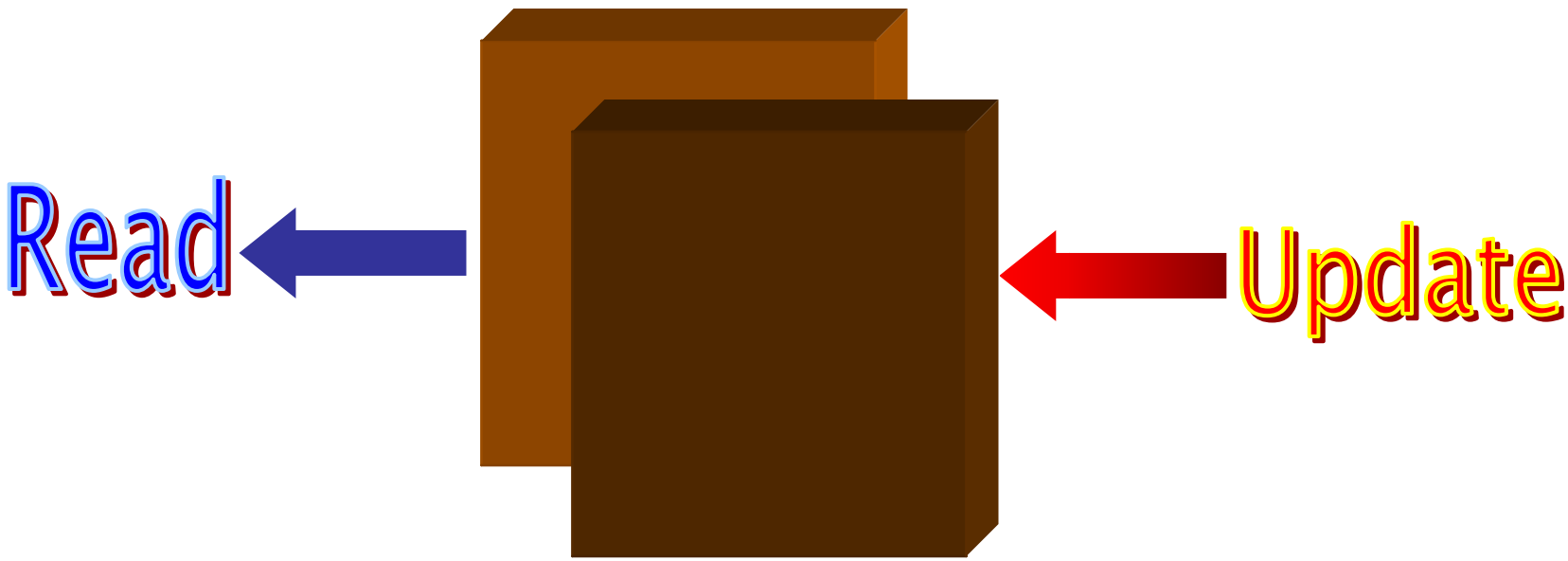
PostgreSQLの機能(2)

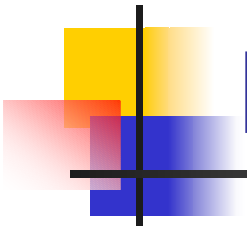
- 行ロック
  - テーブルロック
  - ページロック
- MVCC(Multiversion Concurrency Control)
  - 多版式同時実行制御
  - Oracleの「読み取り一貫性」と同じ機能

行ロック(Row Level Lock)



MVCC(Multi Version Concurrency Control)



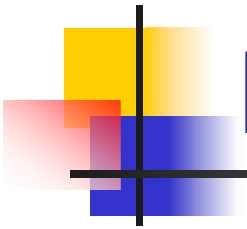


PostgreSQLの機能(3)

- 国際化、日本語対応
  - 日本語をはじめ、ほとんどの国の言語を利用可能
  - Unicode(UTF-8)に対応
  - UnicodeとSJIS/EUC-JPの相互変換をサーバ側で対応 → Unicode対応のアプリケーションがなくてもUnicodeを利用可能

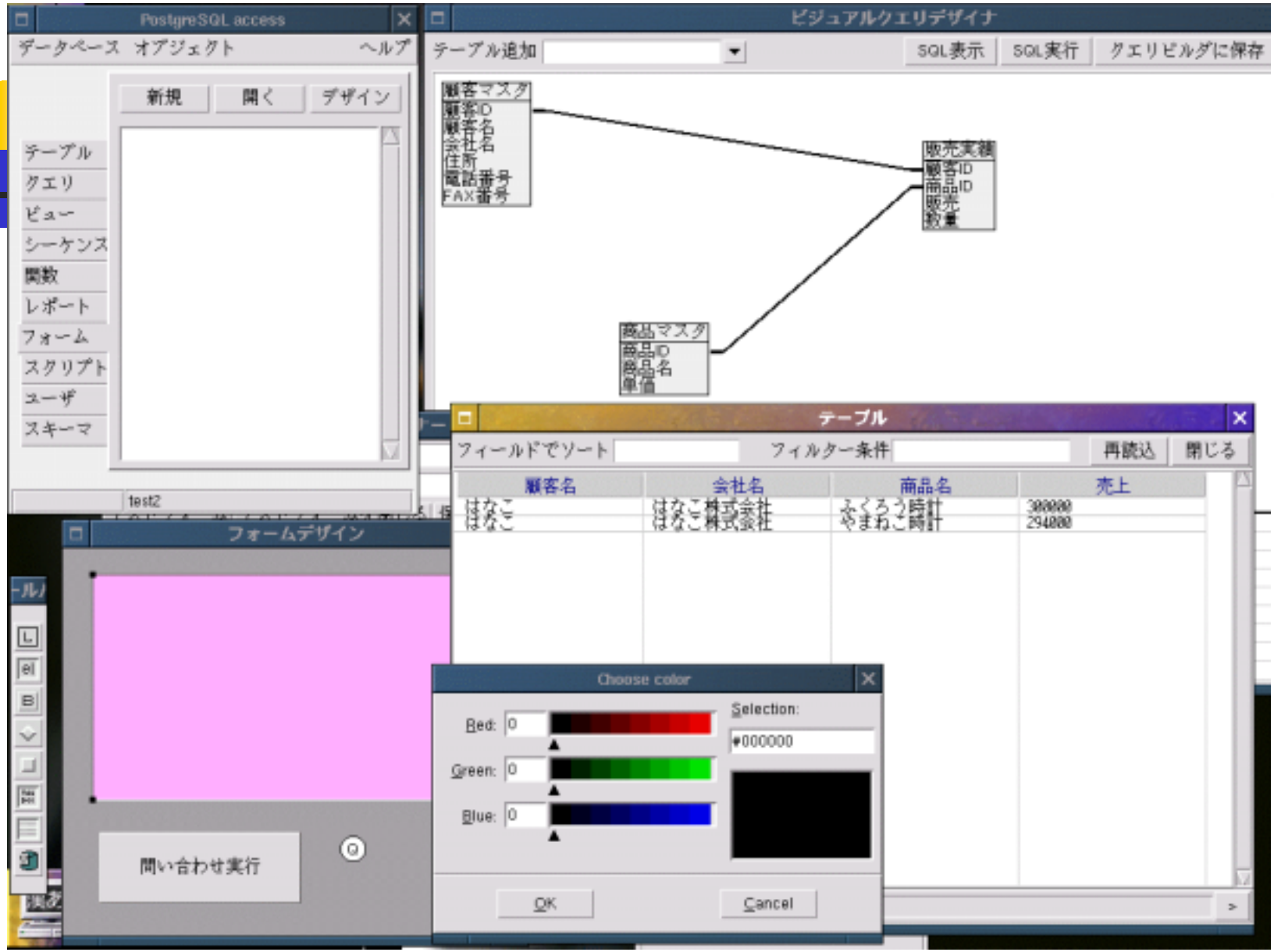
PostgreSQLの機能(4)

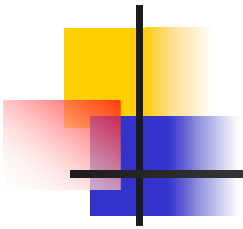
- データベース最大容量
  - 特に制限なし。ただし、1パーティションにデータを収める必要あり
- テーブル最大行数 -- 特に制限なし
- 最大列数 -- 約2000
- 最大データサイズ (1アイテム)
  - 1GB – BLOB/CLOBでの利用も可能
- 最大同時接続数
  - 特に制限はないが、実際問題として100-1000位



PostgreSQLの機能(5)

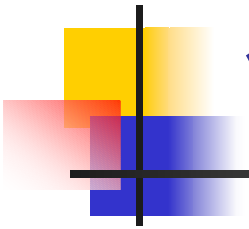
- トランザクションログによるリカバリ
- ホットバックアップ (オンラインバックアップ) 可能
- コマンドラインのSQLインタプリタ
- GUIベースの管理ツールあり





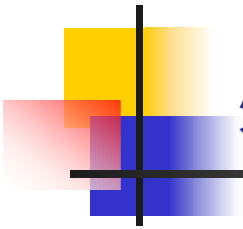
PostgreSQLの導入

- ソースからのコンパイル
  - 最新版を利用できる
  - バグ修正が迅速に行える
  - RPMなどの管理から外れる
  - コンパイル環境が必要
- バイナリパッケージの導入
  - インストール／アンインストールが簡単
  - パッケージ管理との整合性
  - カスタマイズ、バグ対応が困難
  - 中身がブラックボックス（駄目駄目パッケージの存在）
- 現状ではソースからのインストールにメリットあり



ソースの入手先

- **一時配布先**
  - http://www.postgresql.org
- **ファイル名**
  - postgresql-7.2.1.tar.gz

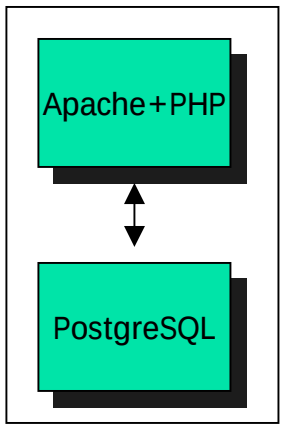


必要条件

- **ディスク容量：120MB(うちソースが60MB)＋ユーザデータベースサイズ**
- **メモリ：512MB以上を推奨(Xなどを動かさなければ64MB程度でも動作する)**
- **最低必要なツール**
  - GNU make
  - gcc

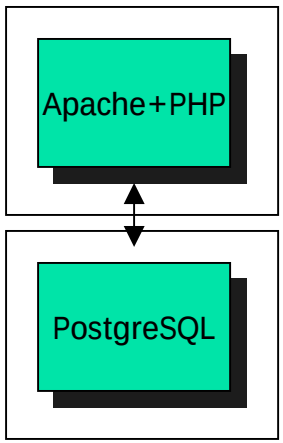
構成の検討

同一ホストにフロントエンドとバックエンド



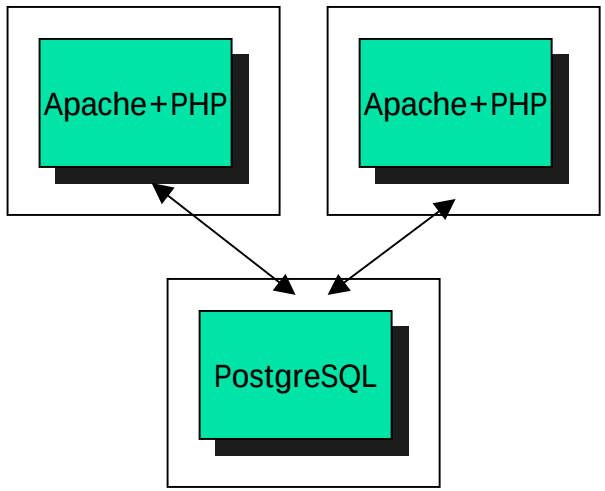
- フロントエンドとバックエンドの通信はUnixドメインソケットまたはTCP/IP
- 小規模、低コスト
- イントラネット

フロントエンドとバックエンドが別ホスト

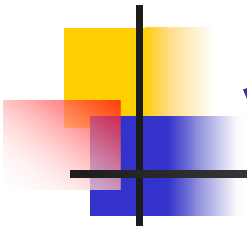


- フロントエンドとバックエンドの通信はTCP/IP
- 中規模システム
- インターネット

複数フロントエンド

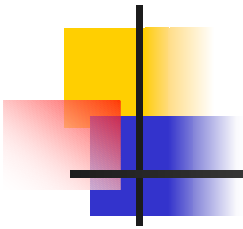


- フロントエンドとバックエンドの通信はTCP/IP
- 大規模システム
- インターネット
- フロントエンドの高可用性、負荷分散



インストールの流れ

- 専用アカウントの作成
- コンパイル
- 回帰テスト(オプション)
- インストール
- .bashrcなどの設定
- postgresql.conf/pg_hba.confの設定
- postmasterの起動



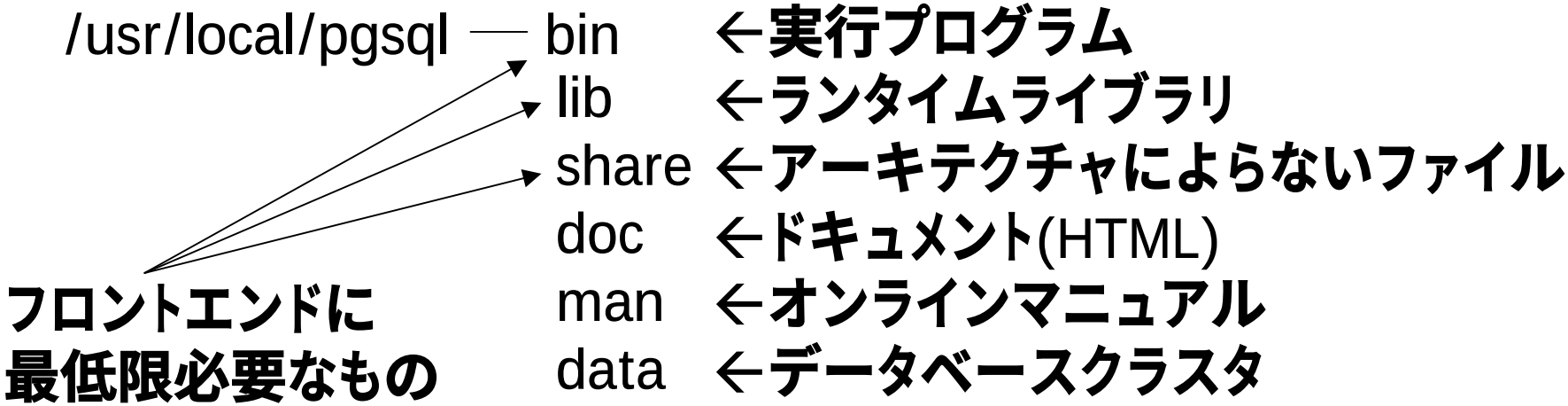
インストールの実際例(1)

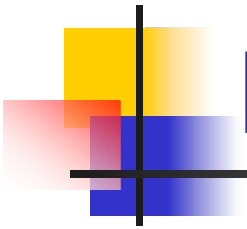
```
# useradd postgres ←postgresユーザの追加
# mkdir /usr/local/src/postgresql-7.2.1 ←コンパイルディレクトリ
# mkdir /usr/local/pgsql ←インストール先
# chown postgres /usr/local/src/postgresql-7.2.1 /usr/local/pgsql
# su postgres
$ cd /usr/local/src
$ tar xfz /tmp/postgresql-7.2.1.tar.gz
$ cd postgresql-7.2.1
$ ./configure --enable-multibyte –enable-unicode-conversion --
    enable-syslog ←マルチバイト対応、Unicode対応、syslog対応
$ make ← コンパイル
$ make check   ← 回帰テスト
```

インストールの実際例(2)

$ make install　← **インストール**

- /usr/local/pgsql/**以下のファイル**(data**を除く**)**はフロントエンドにもコピー可能（ハード、**OS**が同一という前提）**





PostgreSQLの起動と設定

- postgresユーザの.bashrcの設定

PG=/usr/local/pgsql
export PGLIB=$PG/lib ← ランタイムライブラリなど
export PGDATA=$PG/data ←データベースクラスタ
export LD_LIBRARY_PATH=$PG/lib
PATH=$PG/bin:$PATH ←コマンドサーチパス

- データベースクラスタの初期化

$ initdb

- データベースサーバ(postmaster)の起動

$ postmaster -S -i

データベースクラスタの内部構造

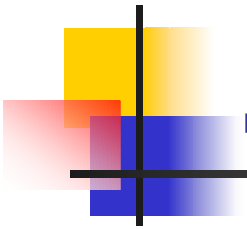
/usr/local/pgsql/data

PG_VERSION	バージョン情報
postmaster.opts	postmasterへの引数
postmaster.pid	postmasterのプロセスID
pg_hba.conf	認証設定
pg_ident.conf	ident認証設定
global	データベース共通テーブル
pg_xlog	トランザクションログ
pg_clog	トランザクション情報
base	この下に各データベースが格納されている

postgresql.conf

- **主設定ファイル**
- **変数 = 値**
- **設定例：syslogに実行SQL文を表示**
  - syslog = 2
  - debug_print_query = true
  - /etc/syslog.conf

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



ユーザとデータベースの作成

- **ユーザの作成**

  $ createuser foo

  Shall the new user be allowed to create databases? (y/n) n

  Shall the new user be allowed to create more new users? (y/n) n

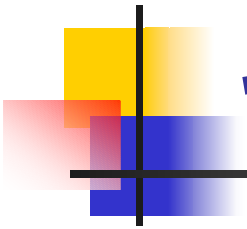
  CREATE USER

- **データベースの作成**

  $ createdb -E エンコーディング foo

  CREATE DATABASE

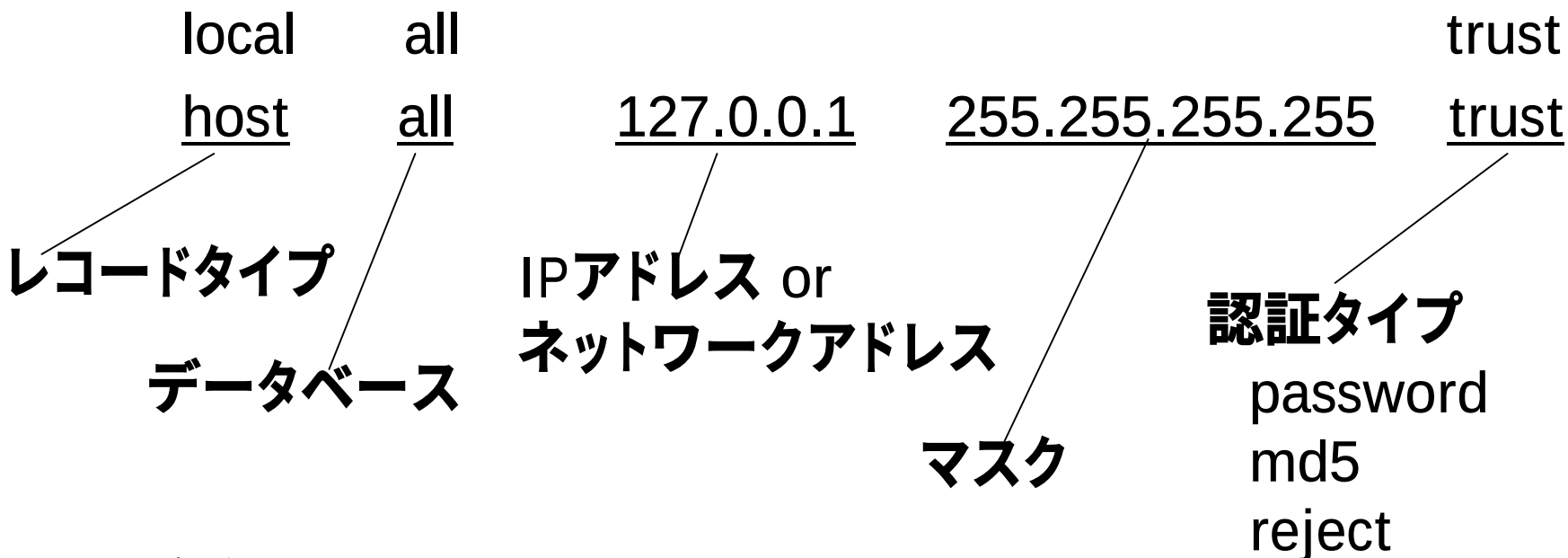
  - **日本語の使用できるエンコーディング**
    - EUC_JP, UNICODE(UTF-8)



セキュリティ

- パスワード認証
- ホストIPアドレスによる認証
- テーブルなどのデータベースオブジェクトレベルでの権限設定
  - GRANT/REVOKE

pg_hba.conf



認証条件: IP = **フロントエンド**IP & **マスク**

例:IP = 192.168.1.0, **マスク** = 255.255.255.0
なら 192.168.1.x**はすべて**OK

運用管理(1)

- VACUUM
  - **テーブルのガーベジコレクション**
  - PostgreSQL**のもっとも重要な運用管理**SQL**コマンド**
  - **通常の**VACUUM
    - **空き領域の登録**
    - **運用中でも実行可能**
    - **少なくとも1日に一回実行**
  - VACUUM FULL
    - **空き領域を物理的に圧縮**
    - **運用中でも実行できるが、処理中のテーブルにロックがかかる**
    - **運用に支障のないときに実行**

運用管理(2)

- VACUUMの設定
  - postgresql.confのmax_fsm_pages ＝ ページ数を設定
    - データベースクラスタ(/usr/local/pgsql/data)の容量を計算(du -sの出力を参考にする)
    - FSMページ数 > DB容量(バイト数)/8192 になるように設定
- VACUUMの実行例
  - $ vacuumdb -a

運用管理(3)

- REINDEX
  - **インデックスの再構築**
  - **運用中にも実行できるが、処理中のテーブル、インデックスにロックがかかる**
  - **目安として、1週間～1ヶ月に1回運用に支障のないときに実行**
    - $ psql -c “REINDEX TABLE foo” bar
  - **数ヶ月に1回、スタンドアローンpostgresから実行することを推奨**(man reindex**参照**)
    - **システムインデックスの再構築**

運用管理(4)

- ANALYZE
  - **問い合わせオプティマイザが使用する統計情報の更新**
  - ANALYZEをしておかないと、正しい問い合わせプランが作成されない→SELECTが遅くなる
  - **データを大量に追加、更新、削除したときに実行することを推奨**
  - **実行例**
    - psql -c “ANALYZE” bar

運用管理(5)

- Web用バックエンドDBの設定ポイント
  - 多数の同時コネクション
  - postgresql.confの設定例
    - max_connections = 128
    - shared_buffers = 1024
    - deadlock_timeout = 128
    - max_files_per_process = 40
  - OS設定例
    - /etc/sysctl.conf
      - fs.file-max = 16384
      - kernel.shmmax = 134217728

運用管理(6)

- バックアップ(ラージオブジェクトがない場合)
  - データベース単位のバックアップ

    ```
    $ pg_dump データベース名 > /tmp/db.out
    ```
  - データベース単位のリストア

    ```
    $ psql -f /tmp/db.out データベース名
    ```

運用管理(8)

- データベースクラスタ全体のデータベース単位のバックアップ
  - バックアップ

```
$ pg_dumpall > /tmp/db.out
```

  - リストア

```
$ initdb
$ psql -f /tmp/db.out template1
```

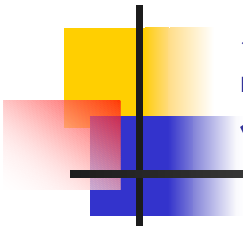
運用管理(9)

- バックアップ(ラージオブジェクトがある場合)
  - データベース単位のバックアップ

    ```
    $ pg_dump -b -F cデータベース名 >
      /tmp/db.out
    ```

  - データベース単位のリストア

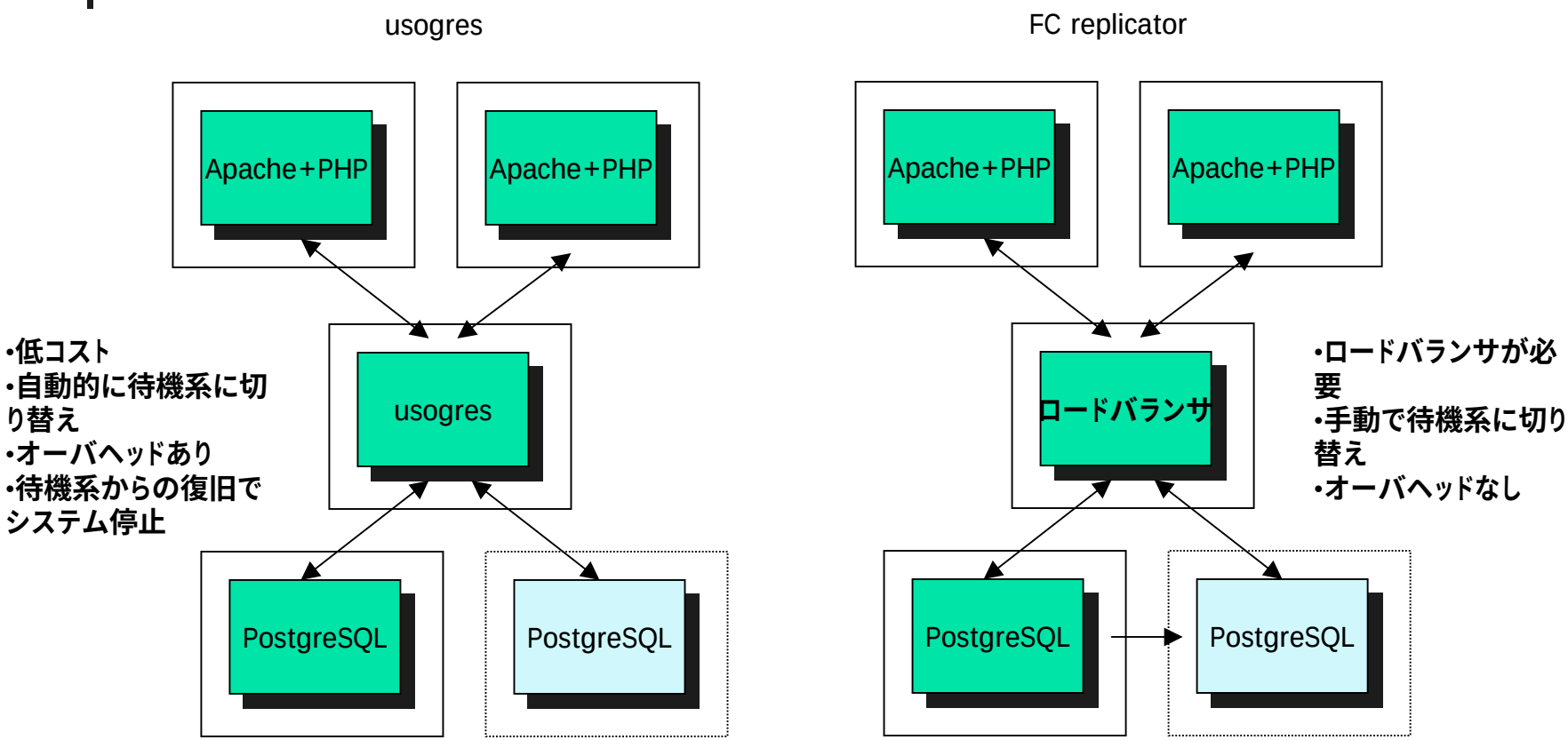
    ```
    $ pg_restore –d データベース名 /tmp/db.out
    ```



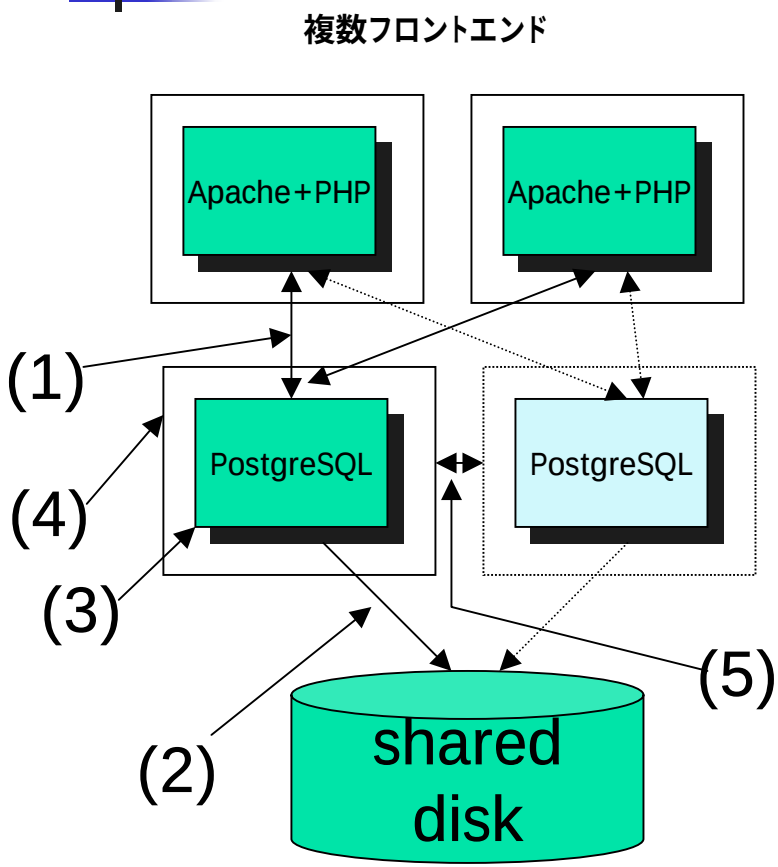
運用管理(10)

- **レプリケーションの利用**
  - **差分バックアップ、オフラインログによるリカバリができない→バックアップだけでは最新のトランザクションを復旧できない**
  - PostgreSQL**が停止しても運用を続けたい**
  - **レプリケーションの導入で解決**
    - usogres
      - http://usogres.good-day.net
    - FC Replicator
      - http://www.fastconnector.com

運用管理(11)
レプリケーション



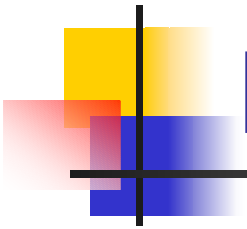
運用管理(12)
高可用性(High Availability)



- フロントエンドとバックエンドの通信はTCP/IP
- 大規模HAシステム
- インターネット
- フロントエンドの高可用性、負荷分散
- バックエンドの高可用性
- いろいろな製品がある

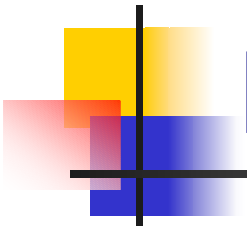
LifeKeeper+PostgreSQL Ark **によるシステムの場合に対応できる障害例**:

(1)**ネットワーク障害**
(2)**ディスクケーブル障害**
(3)postmaster**ダウン**
(4)OS**ダウン**
(5)**ハートビート障害**



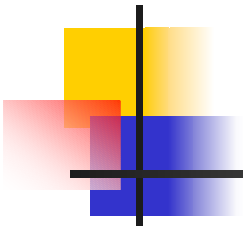
PostgreSQL 7.3について

- DOMAIN
- SCHEMA



DOMAIN

- SQL**標準**
- **データ型＋デフォルト値＋制約＝**DOMAIN
- **同じようなデータ型の宣言の繰り返しを避けることができる**



DOMAIN: 構文

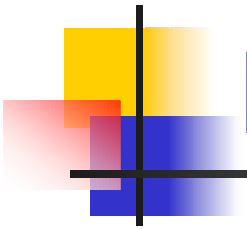
```
CREATE DOMAIN domainname [AS] data_type
    [ DEFAULT default_expr ]
    [ constraint [, ... ] ]

where constraint is:

[ CONSTRAINT constraint_name ]
{ NOT NULL | NULL }
```

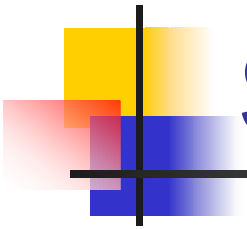
DOMAIN: 使用例

```
test=# CREATE DOMAIN customer_id CHAR(16) DEFAULT
   CURRENT_DATE::TEXT NOT NULL;
CREATE DOMAIN
test=# CREATE TABLE customers(custid customer_id, custname
   TEXT);
CREATE
test=# \d customers
          Table "customers"
  Column  |    Type     | Modifiers
----------+-------------+-----------
 custid   | customer_id | not null
 custname | text        |
```



DOMAIN: 使用例

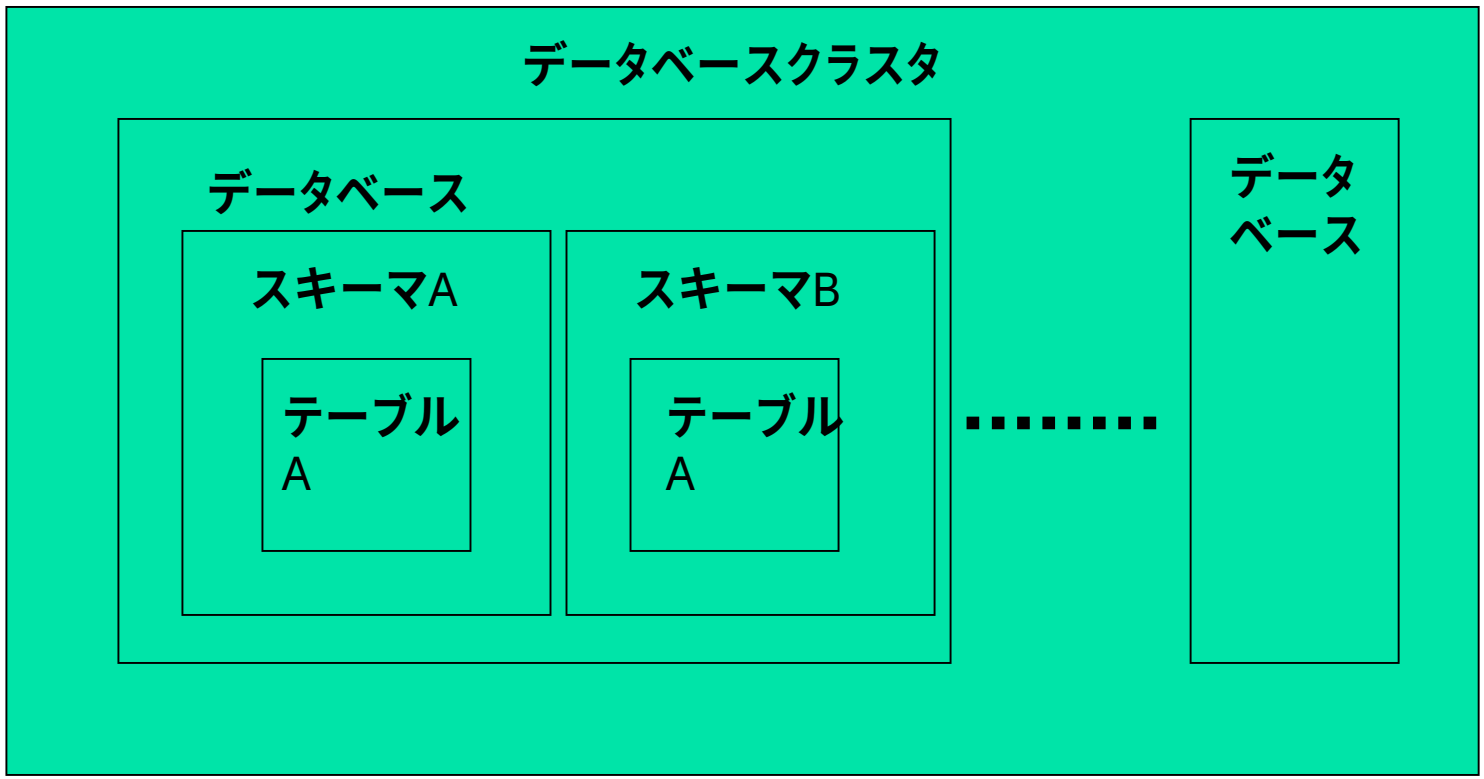
```
test=# INSERT INTO customers(custname)
  VALUES('foo');
INSERT 16607 1
test=# SELECT * FROM customers;
      custid       | custname
-------------------+----------
 2002-05-10        | foo
(1 row)
```

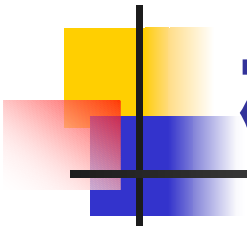


SCHEMA

- SQL**標準**
- **ユーザごとに別の名前空間を持てる**
  - SELECT * FROM foo.table1;

SCHEMA

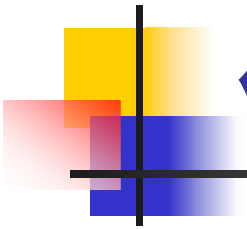




構文

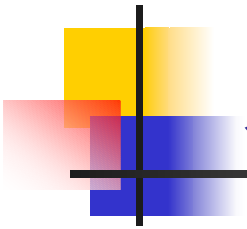
CREATE SCHEMA schemaname
[AUTHORIZATION username]
[schema_element [...]]

CREATE SCHEMA AUTHORIZATION username
[schema_element [...]]



使用例

- myschemaという名前のスキーマを作る。所属はコマンドを実行したユーザ
  - CREATE SCHEMA myschema;
- ユーザfooのためにfooschemaというスキーマを作る
  - CREATE SCHEMA fooschema AUTHORIZATION foo;



スキーマの一覧を見る

```
test=# select * from pg_namespace;
  nspname   | nspowner | nspacl
------------+----------+--------
 pg_catalog |        1 | {=U}
 pg_toast   |        1 | {=}
 public     |        1 | {=UC}
 pg_temp_1  |        1 |
 myschema   |        1 |
 fooschema  |      100 |
(6 rows)
```

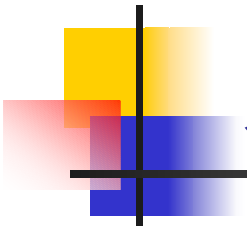
スキーマにオブジェクトを所属させる

```
test=# CREATE TABLE myschema.t1(i INTEGER);
CREATE
test=# CREATE SCHEMA myschema2
test-# CREATE TABLE t1(i INTEGER)
test-# CREATE TABLE t2(i INTEGER)
test-# ;
CREATE
```

スキーマが違えば同じテーブル名が存在できる

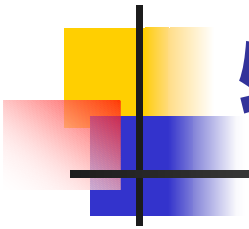
```
test=# create schema myschema;
CREATE SCHEMA
test=# create table t1(i int);
CREATE TABLE
test=# create table myschema.t1(i int);
CREATE TABLE
test=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+---------
 public | t1   | table | t-ishii
(1 row)

test=# set search_path to 'myschema','public';
SET
test=# \dt
         List of relations
  Schema  | Name | Type  |  Owner
----------+------+-------+---------
 myschema | t1   | table | t-ishii
(1 row)
```



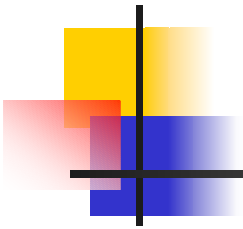
スキーマの使い分け

```
SELECT * FROM myschema.t1;
SELECT * FROM myschema2.t1;
SELECT * FROM customers;
SELECT * FROM publc.customers;
```



特別なスキーマ

- public
  - **すべてのユーザから見える**
- pg_catalog
  - **システムカタログを格納したスキーマ**
  - search_path**に関わらず、必ず検索される**



スキーマと権限

- GRANT { { CREATE | USAGE } [,...] | ALL [PRIVILEGES] } ON SCHEMA schemaname [, ...] TO { username | GROUP groupname | PUBLIC } [, ...]
- CREATE
  - **スキーマ内でのオブジェクトの作成権限**
- USAGE
  - **スキーマ内のオブジェクトの参照権限**

行を返す関数

```
test=# CREATE TABLE foo (fooid int, foosubid int, fooname
   text, primary key(fooid,foosubid));
test=# CREATE FUNCTION getfoo(int) RETURNS setof foo AS
   'SELECT * FROM foo WHERE fooid = $1;' LANGUAGE SQL;
CREATE
test=# SELECT * FROM getfoo(1) AS t1;
  fooid | foosubid | fooname
-------+----------+---------
      1 |        1 | Joe
      1 |        2 | Ed
(2 rows)
```

参考文献・URLなど

- PostgreSQL**オフィシャルマニュアル**
  - ISBN 4-8443-1589-7
- **「本家」サイト**
  - http://www.postgresql.jp
- **日本**PostgreSQL**ユーザー会**
  - http://www.postgresql.jp
  - pgsql-jp ML
- **商用サポート、技術情報提供**
  - http://www.sra.co.jp

おしまい