PostgreSQL で並列処理を実現する pgpool-II のご紹介

SRA OSS, Inc. 日本支社
技術部　盛　宣陽
Yoshiharu Mori
y-mori@sraoss.co.jp

PostgreSQLに対する要求

- 1 台では処理能力が限界にきている
  - システムのボトルネックになっている！
  - コネクションプーリングによる負荷の軽減
  - クラスタリングによる負荷分散
- 信頼性の向上
  - システムを止めたくない！
  - レプリケーションによるデータ冗長化

これらの要求に答えるため、pgpoolを開発

pgpoolとは？

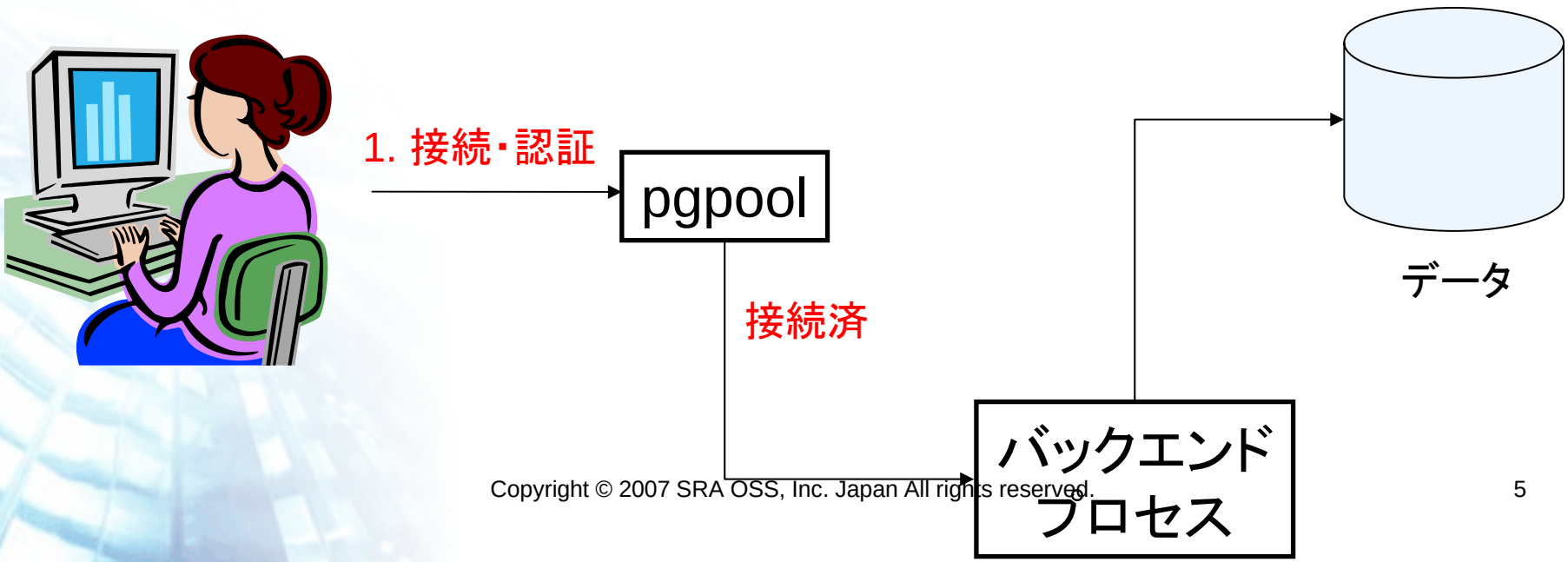
- 2003年開発スタート
  - 弊社石井が個人で開発
  - 最初はコネクションプーリング機能のみ(pgpoolという名前の由来はここからきている)
- ライセンス
  - BSDライセンス
  - ソースコードが公開され、自由に改造してもよい
- 現在の開発体制
  - pgpool Global Development という開発団体に移行

pgpoolの機能

- コネクションプーリング
  - 接続時のオーバーヘッドを軽減
- レプリケーション(データの複製を作る)
  - 2台までの制限あり
- ロードバランス
  - レプリケーション設定時にのみ検索を振り分けることが可能
- フェイルオーバ
  - 障害が発生したノードを切り離す

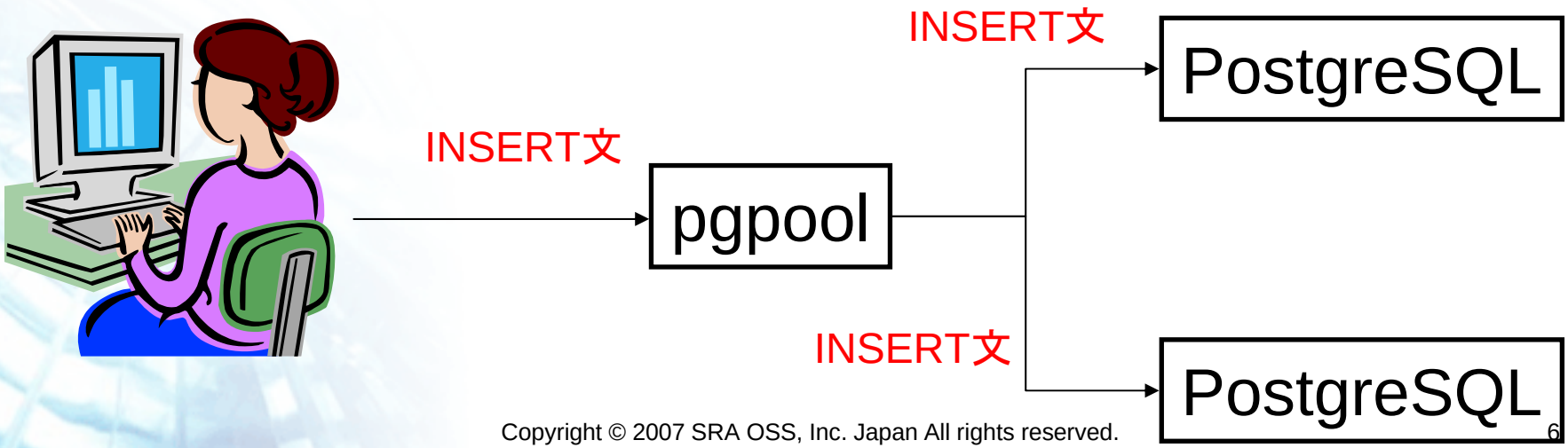
コネクションプーリングの利点

- 接続オーバーヘッドの軽減
  - pgpoolによるコネクションプールの接続処理の流れ
    - クライアントとPostgreSQLの間にpgpoolが入る
    - プロセス生成とディスクアクセスのための準備処理が省略可能



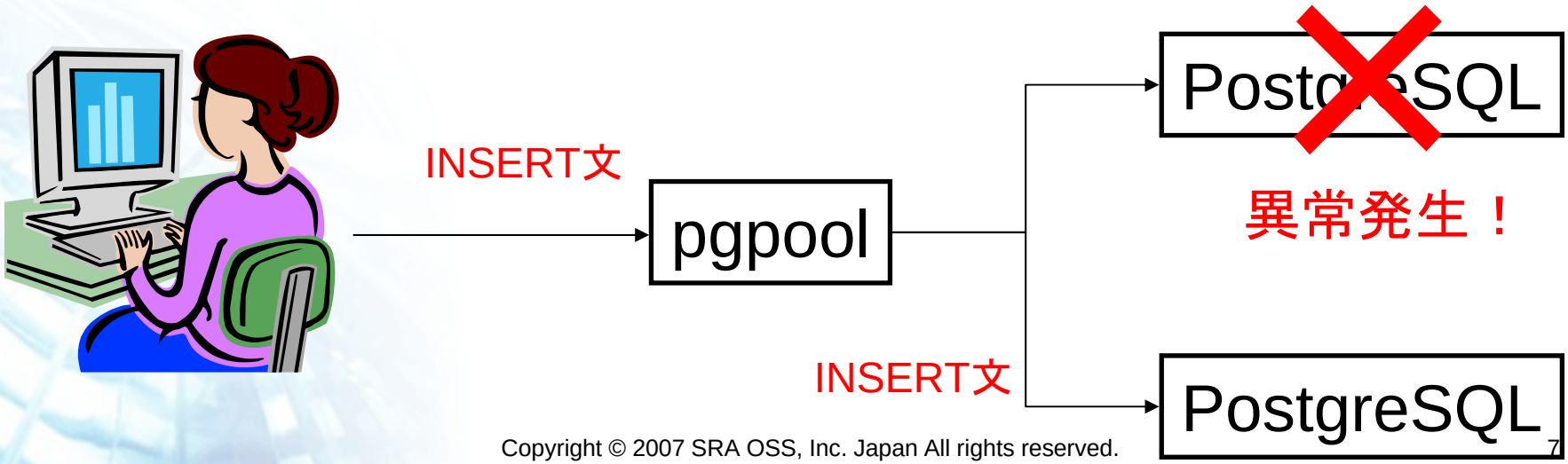
レプリケーションの利点

- 信頼性向上
  - 2台に更新系クエリを送信することで、データの同期を取る
    - 異常が発生した場合にも運用が可能(縮退運転)
    - 常に同期を取るため、通常より更新系クエリは時間がかかる



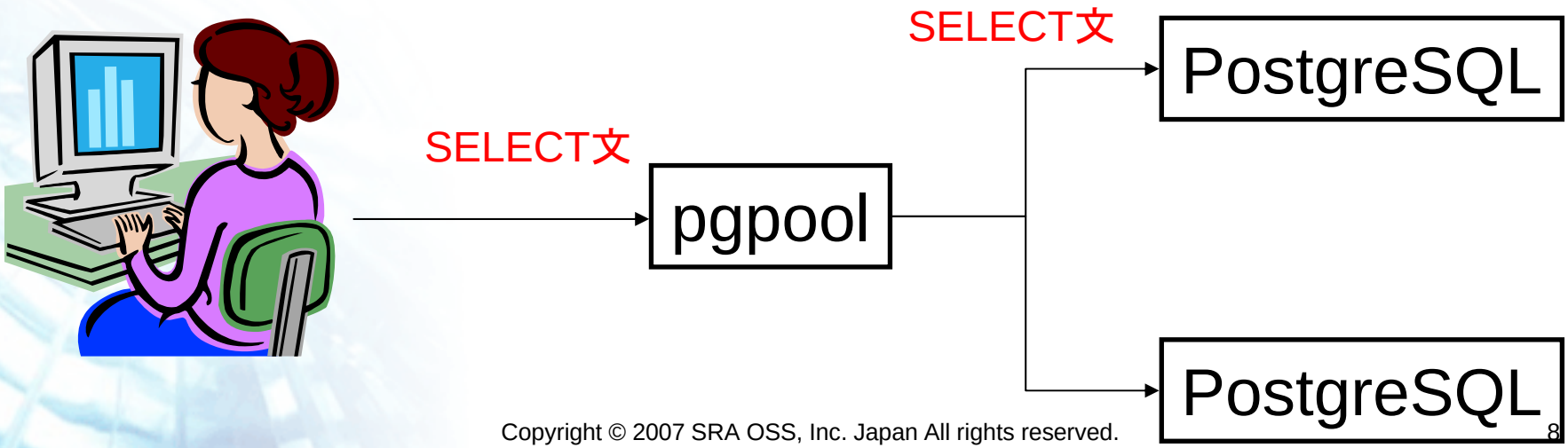
レプリケーションの利点

- 信頼性向上
  - 2台に更新系クエリを送信することで、データの同期を取る
    - 異常が発生した場合にも運用が可能(縮退運転)
    - 常に同期を取るため、通常より更新系クエリは時間がかかる



レプリケーションの利点

- ロードバランス
  - 同じデータを2台で持つので、検索についてはどちらかに問い合わせすればよいので、負荷分散が可能
  - ロードバランス先の割合を指定可能



pgpool-IIとは？

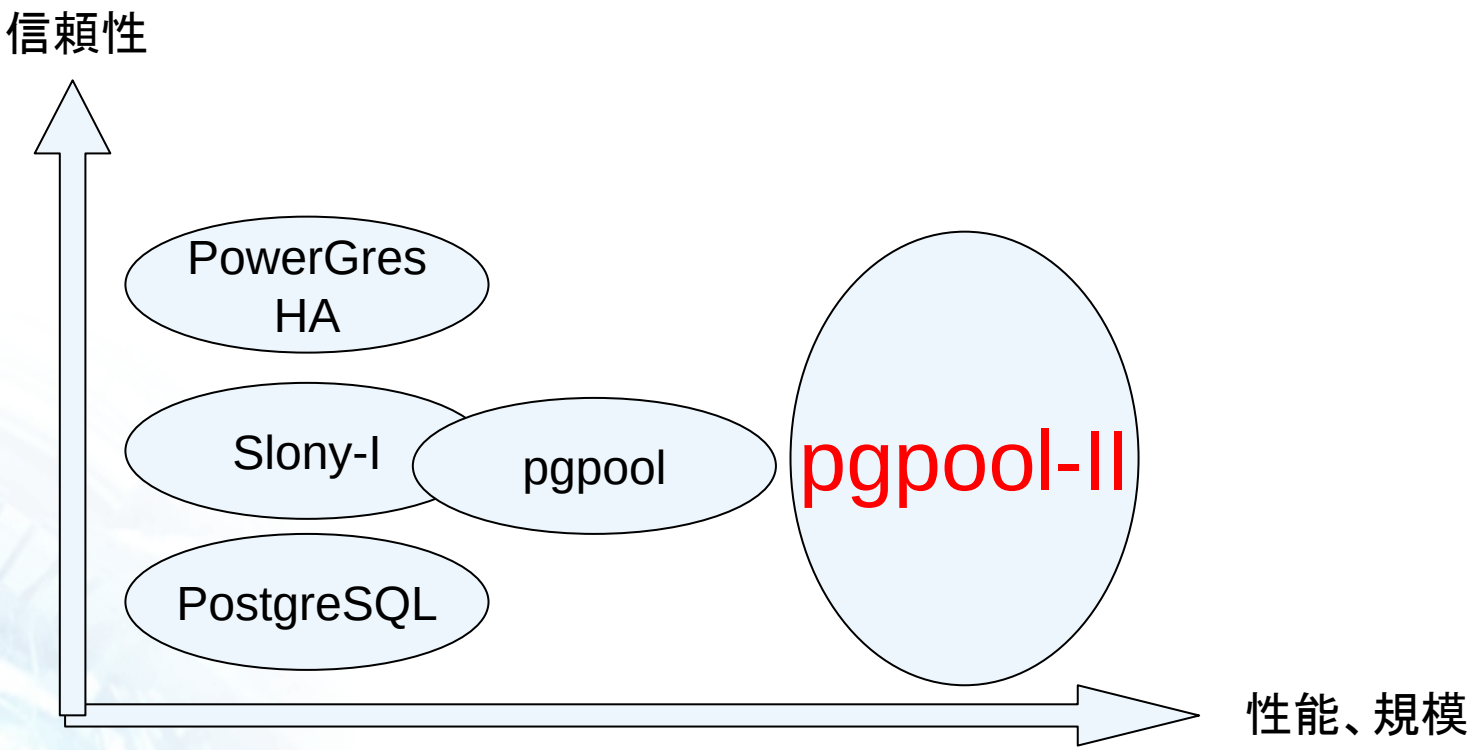
- IPA「オープンソースソフトウェア活用基盤整備事業」の支援により SRA OSS, Inc. 日本支社で開発

- 現在は pgpool Global Development Groupにソースコードを寄贈
  - 弊社の開発メンバも何人か所属
  - ライセンスはpgpoolと同じ

pgpool-IIの機能

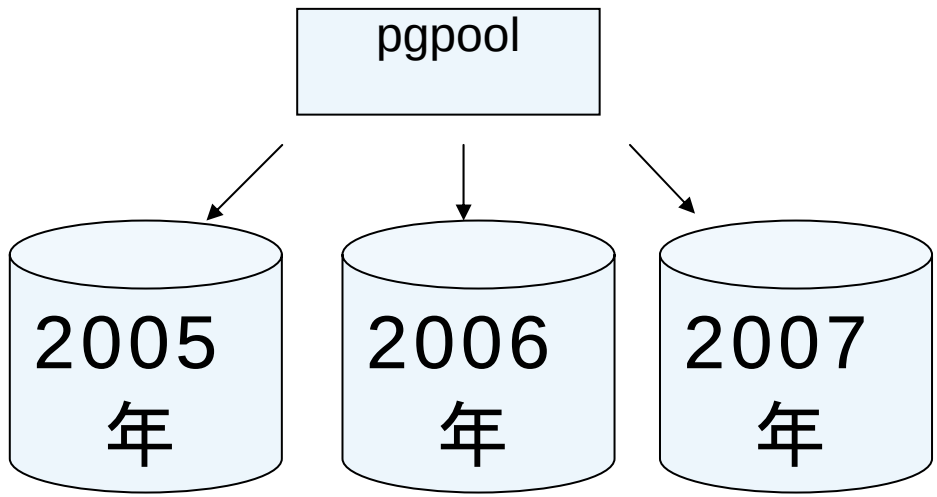
- pgpoolの後継プロダクト
  - pgpoolの機能を継承
  - レプリケーションを2台までしかできない制限を解除
- pgpool-II新機能
  - 管理ツールによるpgpoolの管理
  - クエリキャッシュ
  - パラレルクエリ

pgpool-IIのターゲット市場



パラレルクエリ

- レプリケーションでは複数のデータベースノードに同じデータを複製して格納するのに対して、パラレルクエリでは複数のデータベースノードに異なる範囲のデータをそれぞれ格納します



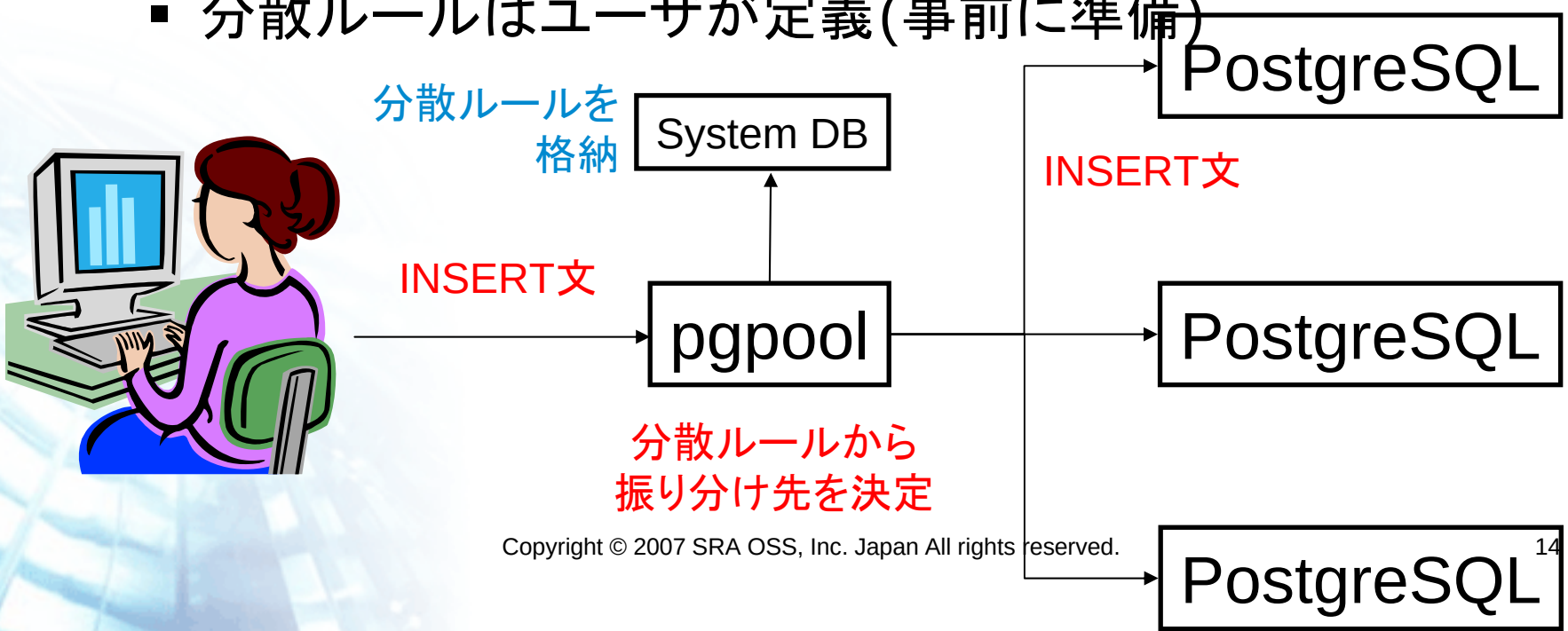
パラレルクエリ (準備)

- System DBと呼ばれるDBに右のテーブルを作成する
- パラレルモードで利用するテーブルの情報を右のテーブルにあらかじめ入れておく
- dist_def_funcに指定した関数をSystem DBに作成する
  (分散ルールの登録)
- pgpool起動時にSystem DBから情報を読み込む
- クエリ書き換えが必要な時にはこの情報を元に書き換えを行う

```
CREATE TABLE pgpool_catalog.dist_def (
   dbname text,           -- データベース名
   schema_name text, -- スキーマ名
   table_name text,      -- テーブル名
   col_name text
        NOT NULL CHECK (col_name = ANY
     (col_list)),          -- 分散キー列名
   col_list text[] NOT NULL,
                          -- テーブルの列名
   type_list text[] NOT NULL,
                         -- テーブルのデータ型
   dist_def_func text NOT NULL,
                          -- 分散ルール関数名
    PRIMARY KEY (dbname, schema_name,
     table_name)
);
```

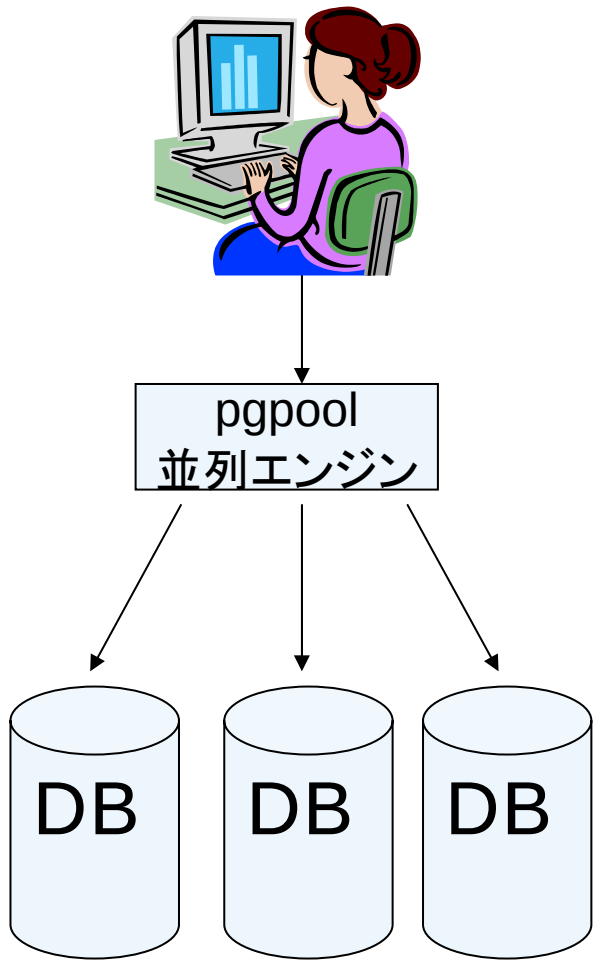
パラレルクエリ

- データを複数サーバに分散して配置
  - 複数のサーバに配置されたテーブルを仮想的に1つのテーブルとして扱う
  - 分散ルールはユーザが定義(事前に準備)



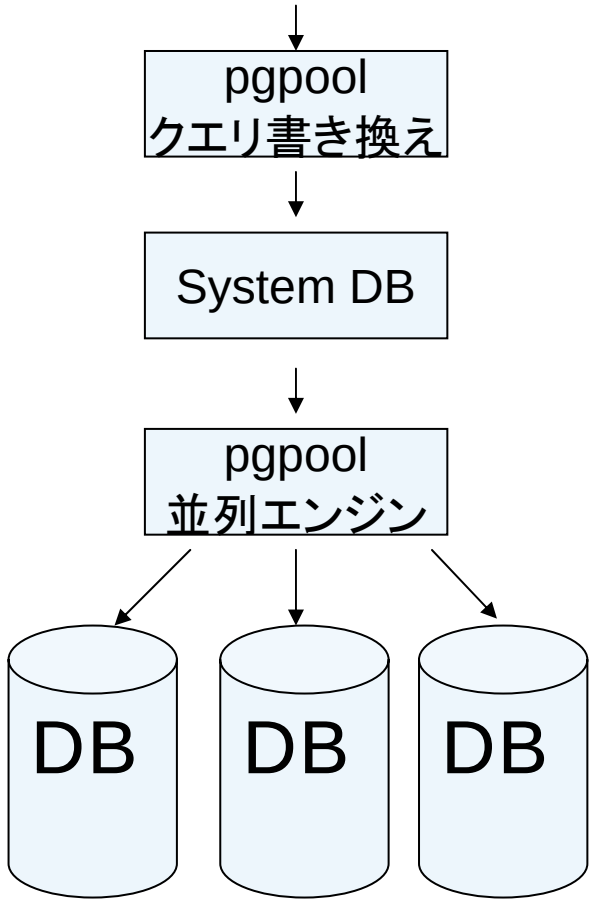
パラレルクエリ(Simple Select)

- pgpoolがクエリを解析して、書き換えが必要かどうか判断を行う
- 書き換えが**必要ない**クエリに対しては、並列エンジンを通して各バックエンドにクエリを送信
- 結果をまとめてクライアントに返す



パラレルクエリ (Complex Select　例：ソート処理)

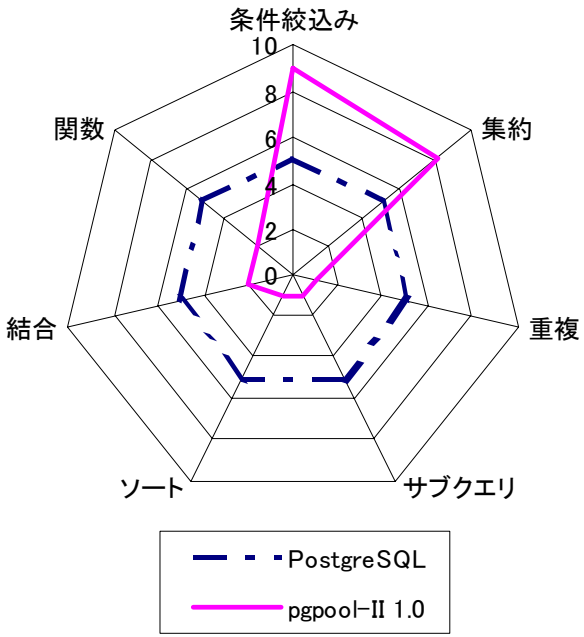
- pgpoolでクエリを解析
- ソート処理が使われている場合には、クエリを書き換えてsystem DBにクエリを送信する
- system DBでは、クエリの一部だけをpgpool並列エンジンに送り、戻ってきた結果にソート処理を行う



パラレルクエリの利点・欠点

- 利点
  - 大容量のテーブルへの検索の高速化
    - PostgreSQL単体の場合、I/Oの負荷がすべて集中する
- 欠点
  - データ量が多くないとそれほど効果があがらない
    - 一台で十分なパフォーマンスが出ている場合はpgpoolが入ることによりオーバヘッドが増える
  - トランザクション処理
    - system DB経由でバックエンドに送信されるクエリはトランザクションブロック外の処理となる
  - 苦手なクエリが多い

パラレルモードでの苦手なクエリ



- 条件絞込み
  - Where句の条件処理
- 集約
  - 集約関数、group by
- 重複処理
  - distinct,union,except..
- サブクエリ
- ソート
  - Offset limit 含む
- 結合
- 関数
  - ユーザ定義関数などを考慮

パラレルモードのチューニング

- 各ノードで完結する複雑なクエリは、あらかじめviewを作っておく(たとえばJOIN)
  - pgpoolはviewとtableの判断ができない
- **CE (Constraint Exclusion）機能を利用する**
  - 注) テーブルの継承を行う必要はない

pgpool-IIのパラレルクエリの効果

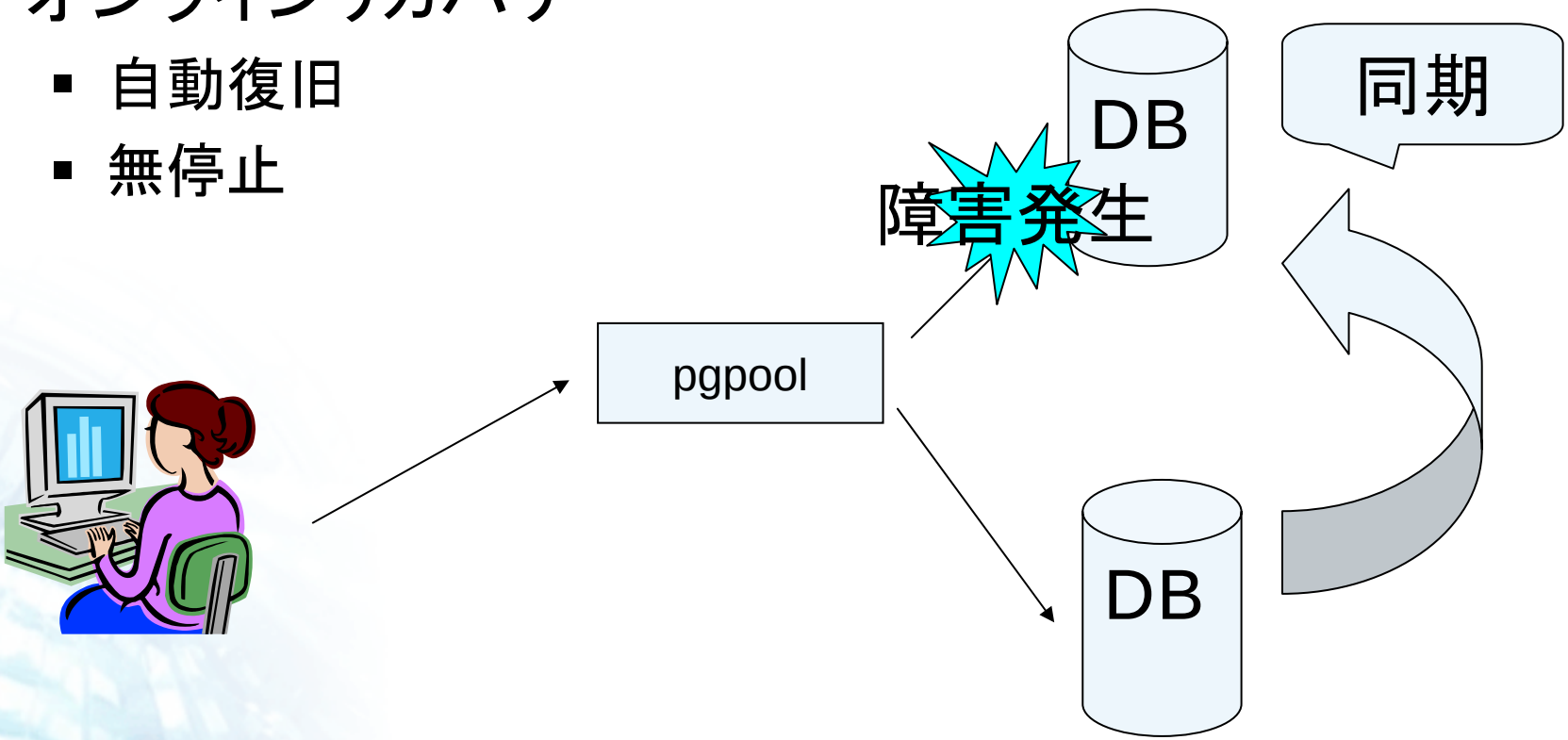
- 次のセッションで詳しくお話します

制限事項

- オンラインリカバリができない
  - ノード間のデータの再同期には、PostgreSQLを停止させ、rsync等で手動でコピーする必要がある
- 一部のSQL(nowなど)には対応していない
- パラレルモードではJDBCなどで使う問い合わせには今のところ対応できない(プリペアードステートメント等)
- パラレルモードでのトランザクション処理

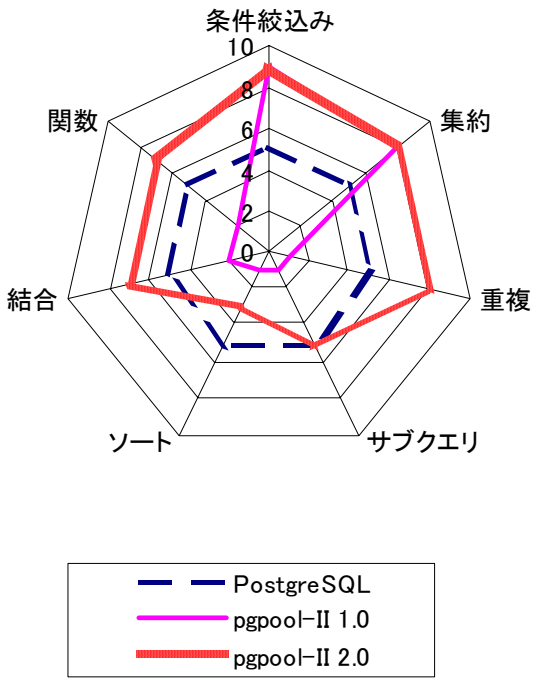
pgpool-II 2.0 今後の計画

- オンラインリカバリ
  - 自動復旧
  - 無停止



pgpool-II 2.0 今後の計画

- パラレルモードの改良
  - 高速化(関数、重複処理)
  - テーブル単位でレプリケーションorパーティショニングの選択が可能
  - トランザクションの改良



参考URL

- pgpool-II開発サイト
  - http://pgfoundry.org/projects/pgpool
- pgpoolコミュニティサイト
  - http://pgpool.sraoss.jp/
- 日経IT Proでの紹介記事
  - http://itpro.nikkeibp.co.jp/article/COLUMN/20060626/241783/