最新版PostgreSQL9.1の新機能

と、ちょこっただけ9.2への展望

Edit : 2012.02.11

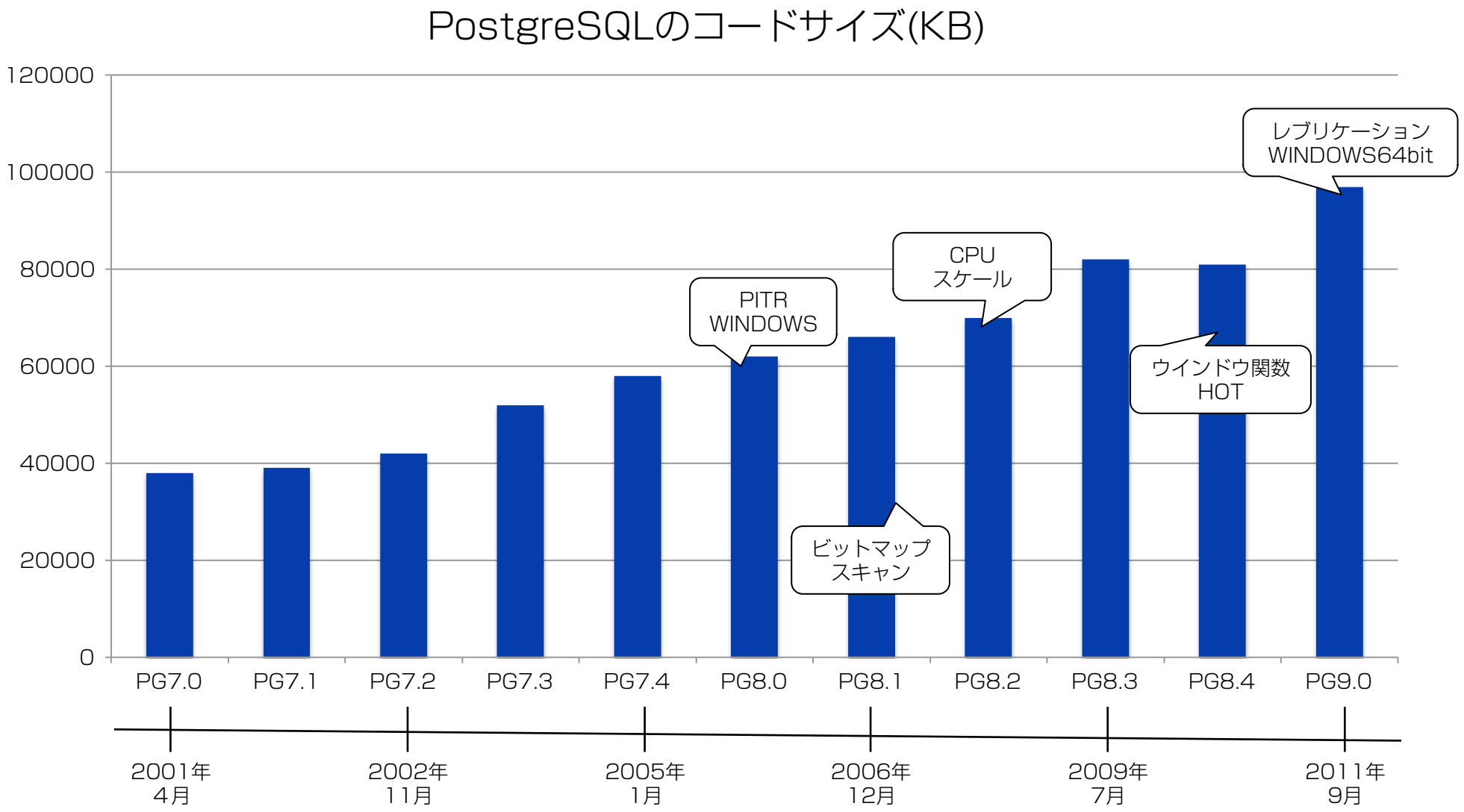
PostgreSQLとは？

- オープンソースのRDBMSで、UNIX系、windows等々色々なプラットフォームで動作します。
- Ingres(1970～)を先祖に持つ。PostgreSQL6.0(1996～)から10年以上の歴史
- BSDタイプのライセンスで配布。PostgreSQL Global Development Group と UCB(UC Berkeley)が著作権を持つ。

PostgreSQLとは？

- ひとつのオーナ企業、オーナー個人を持たない。
  技術者を提供している企業がいくつかある。
- http://www.postgre.org が本家サイト
- SRA OSS, Inc. 日本支社さんとかが、サポート＆保守サービスをしたりしています。

PostgreSQLの歩み



こんなところにも利用されている。

- 国内初の流通小売業向け SaaS 型基幹システム
  2010カンファレンス基調講演から
- (CNAF)　-3,700万人の関係者(受給者)がいる(フランス国民の約半数)毎月約25億ユーロ(2500億円程度)の支給システム。毎日10億SQL文を実行
  PostgreSQL Conference ２０１２基調講演

現在の最新バージョン

- ９／１２に9.1がリリース
- ９／２６日にマイナーバージョンUP
  9.1.2が最新版(2011.12.05)
- その他のバージョン
  9.1.1 / 9.0.5 / 8.4.9 / 8.3.16 / 8.2.22
  8.2は2011年12月でサポート終了。

9.1にそなわった機能

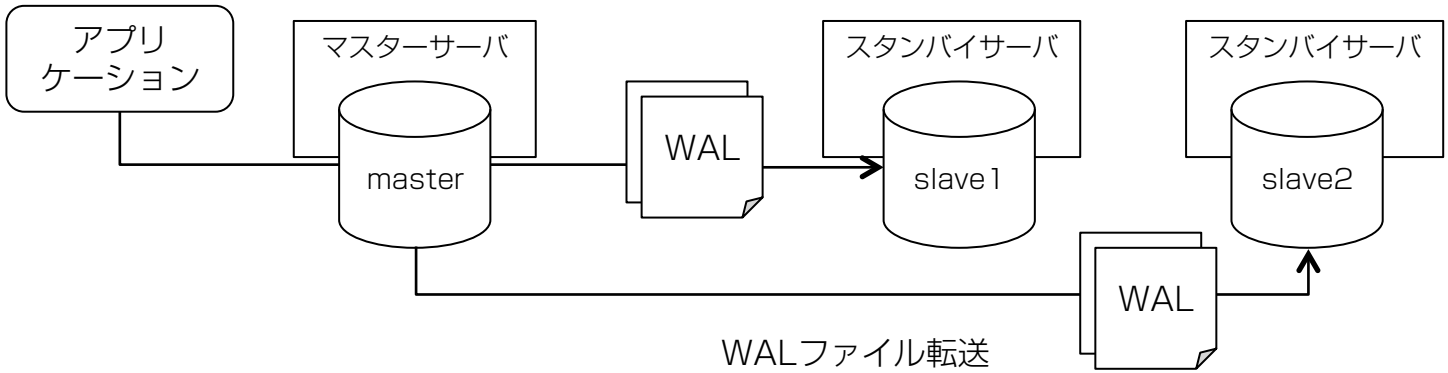
- 同期レプリケーション
- レプリケーション周りの管理コマンド
- SQL/MED（Management of External Data)
- CREATE EXTENSION
- UNLOGGEDテーブル
- 継承テーブルでのINDEX利用
- 賢いCLUSTERコマンド
- KNN GiST インデックス

9.1にそなわった機能

- Contrib/sepgsql
- さまざまな文字列関数の追加 format()等。
- 更新を行えるWITH句 MERGEの代用ができる。
- ビューに対するトリガー
- カラム単位のロケール指定
- Contrib/ pg_trgmの拡張
- エスケープでの注意点(機能じゃないけど・・・)

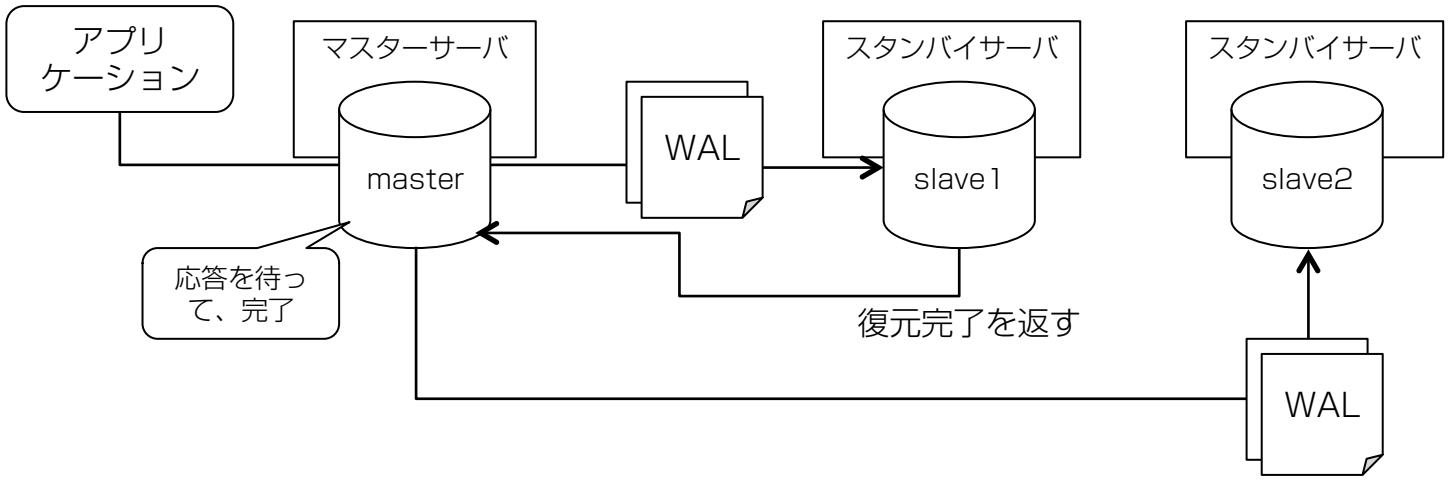
同期レプリケーション

- 9.0では、非同期のみ(=障害直近データ損失あり)
  - WALデータの転送するだけ。
  - カスケーディングはできない。(9.1も)



同期レプリケーション

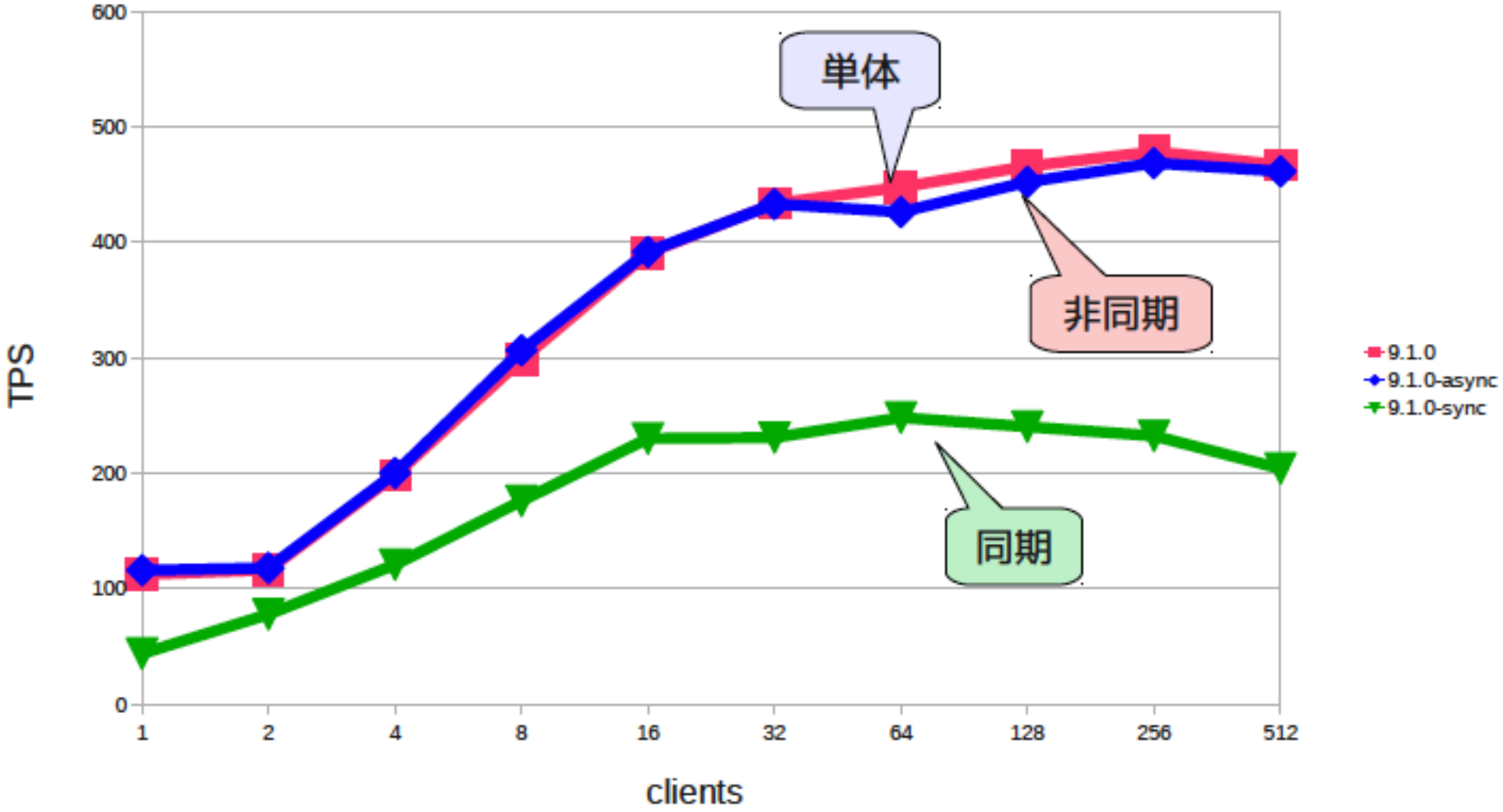
- 同期出来るスレーブは一台のみ。
  2台目からは、非同期。処理速度の問題。
  - 同期したという応答をマスターが待つため。



マスターのpostgresql.conf に synchronous_standby_names = 'slave1,slave2'
//スタンバイサーバ名を記載、カンマ区切りで左から優先度が高い。この場合slave1がダウンするとslave2が同期対象になる。

同期レプリケーション

- パフォーマンス(三谷さんの資料から)



同期レプリケーション

- レプリケーションは高可用性用途
  - 負荷分散用途には向かない（トレードオフの関係）
  - ただし、参照負荷の分散には使える（同期だから）
- レプリケーションはバックアップではない（ここ重要）

  履歴をもっていない（ミラーリングと同じ）

同期レプリケーション(MySQL比較)

- カスケード接続可
- スレーブの接続はわりと自由。
- テーブル単位、DB単位でレプリケーション可
- レプリケーションするテーブルやDBを指定できる。

同期レプリケーション

- マスタDB故障時にtake overが可能
  - マスタDBをメンテナンス停止する場合もサービスを止めなくてよい
    一時的に synchronous_commit パラメータを “local” にすることでも可能。
  - take over やrecoveryはコマンドが用意されている
    ただし、自動ではない（手動）

```
$ pg_ctl promote -D standby_data // マスターへ昇格させている
//その後、元マスタサーバにスタンバイの設定をしてあげるとスタンバイサーバとして使える。
// recovery.conf で、 recovery_target_timeline = leastest と設定する。
```

同期レプリケーション

- ネットワーク設計は注意が必要

  takeoverしてもIPアドレスはかわらない。

  PostgreSQLで管轄することでない。(役割分担に悩む)
- PaceMakerのMaster-Slave型リソースとかと組み合わせて考える必要がありそう。HA構成。
- 最後のスレーブが壊れないこと。(マスタが応答待ちになる)

レプリケーション周りの管理コマンド

- pg_stat_replicationビュー

  マスターでレプリケーションの状況を一覧できる。

  SELECT * FROM pg_stat_replication ;

- pg_basebackup

  ベースバックアップがこれで完了。コマンド一つで、スレーブへデータが作成できる。Pg_start_backup()->pg_dump->転送->復帰などをしなくてOK.

```
$ pg_basebackup -h db1 -U postgres -x -checkpoint=fast -P -D standby_data
// pg_hba.conf や、postgresq.conf(listen_address)で接続できるように設定は必要
// standby_data 以下にDB一式が作成される
```

レプリケーションの簡単手順

- マスター側のサーバにPostgreSQLをインストール
- スタンバイ側のサーバにPostgreSLQをインストール
- 両方のpg_hba.confに接続情報を記載。
- スタンバイ側から、pg_basebackupコマンドを発行
- スタンバイにコピーされたdataフォルダの適度修正
  postgresql.confのスタンバイ用の修正、pidファイルの削除等。
- recovery.confの作成
- 両方起動、コマンド等でレプリケーションを確認。

レプリケーション周りの管理コマンド

- pg_ctl promote

  スタンバイのレプリケーションを終了させて通常起動(=マスタ昇格)させるコマンド。

  これまでは、トリガーファイルを置く方法のみだった。

```
$ pg_ctl promote -D standby_data // マスターへ昇格させている
//その後、元マスタサーバにスタンバイの設定をしてあげるとスタンバイサーバとして使える。
// recovery.conf で、 recovery_target_timeline = leastest と設定する。
```

SQL/MED (Management of External Data)

- 実際の利用方法

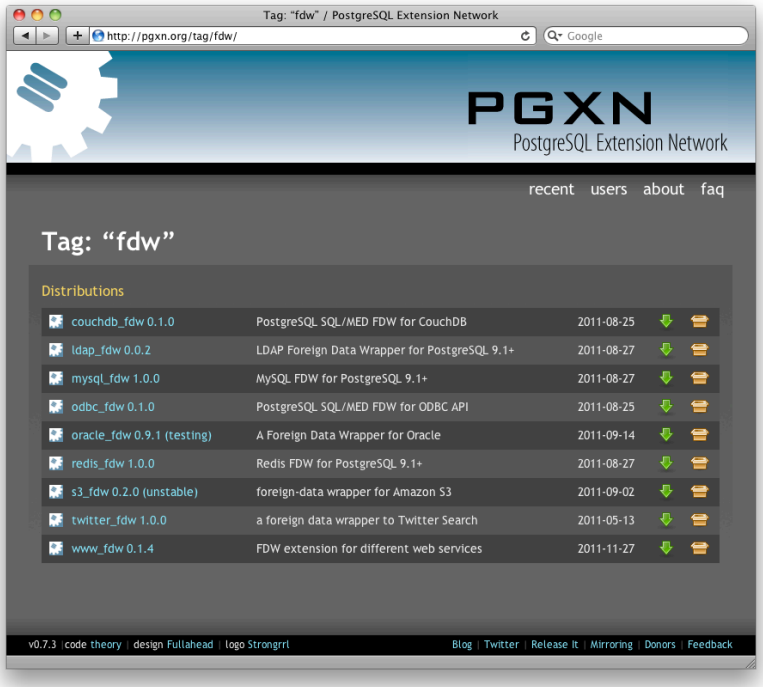
  fdw(Foreign Data Wrapper)を追加して利用。

  CREAET FOREIGN TABLE で定義。

  PostgreSQL Extension Network(PGXN)

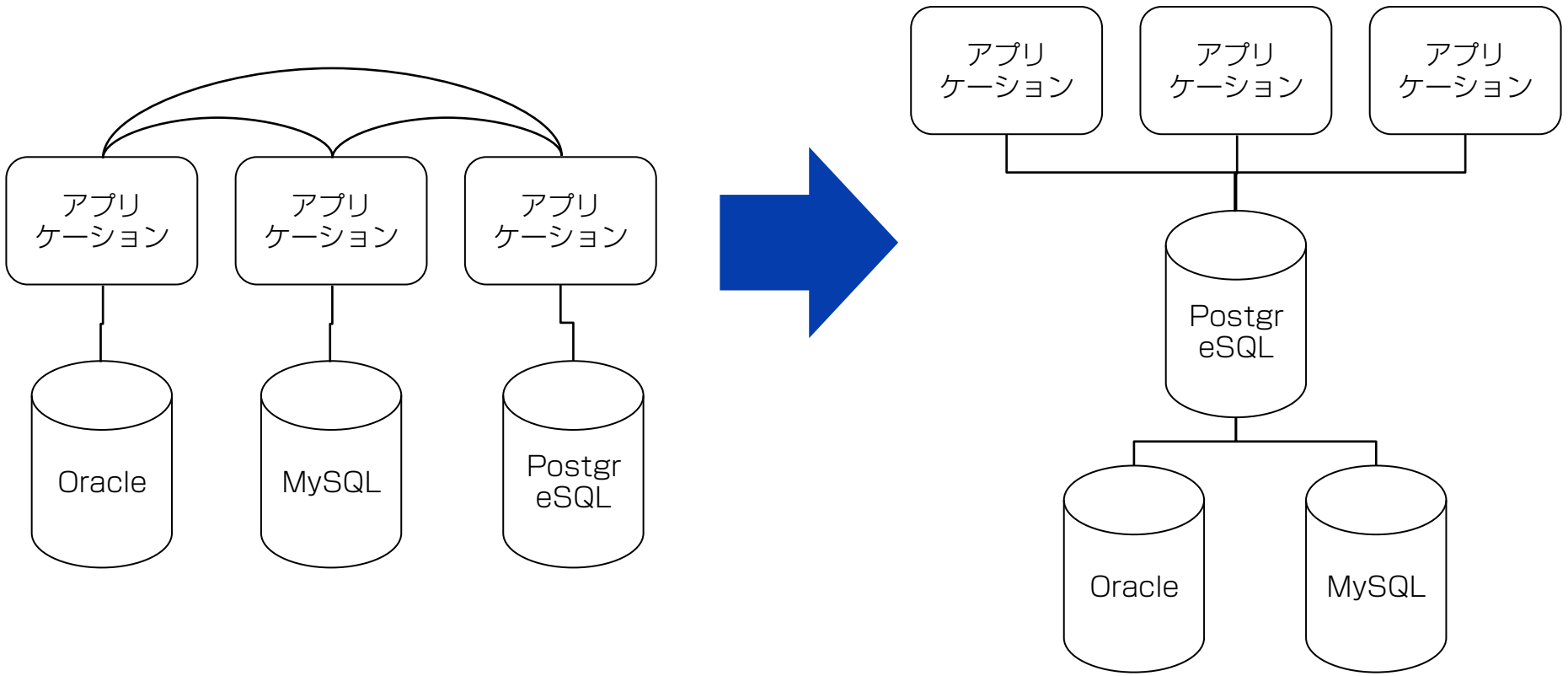
URL : http://pgxn.org/tag/fdw/
2011/12/03 現在
couchdb_fwd 0.1.0
ldap_fwd 0.0.2
mysql_fwd 1.0.0
odbc_fwd 0.1.0
oracle_fwd 0.9.1(testing)
redis_fwd 1.0.0
s3_fwd 0.2.0(unstable)
twitter_fwd 1.0.0
www_fwd 0.1.4

なぜか、PostgreSQL用はまだない。



SQL/MED (Management of External Data)

- こんなのが、こうなるかも。。。



エスケープでの注意点　(互換性の問題)

- エスケープシーケンスの作法
  - standard_conforming_string=on がデフォルト

  誤　‘xxxx\’xxxxx’

  正　E‘xxxx\’xxxxx’

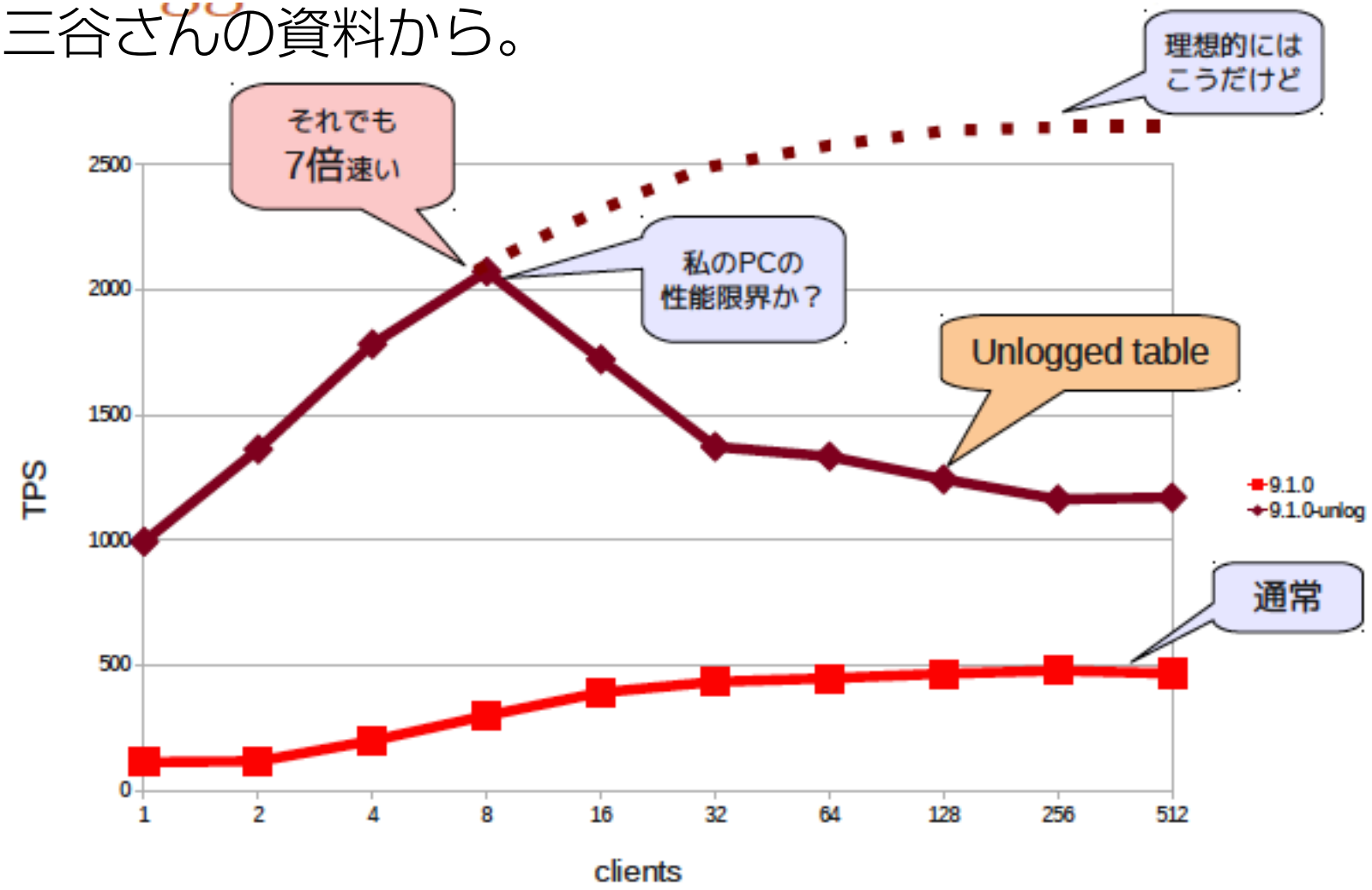
  - 変更の理由。標準SQLで規定されてるから。。。。
  - offで互換モードでの運用をお奨め。（゜Д゜)y―┛~~
- キャストや配列の型チェックの厳格化

UNLOGGEDテーブル

- トランザクションログを書かないテーブル
  - データはハードディスクに書き込む
  - 普通に使う分には問題ない
  - 停電やクラッシュした場合は消えるかも
- ログが書かれないのでレプリケーション対象外
- 更新速度重視のテーブル
- 使い所
  - 速度重視で crash safeでないテーブル。
  - in-memory / global temporary 風。

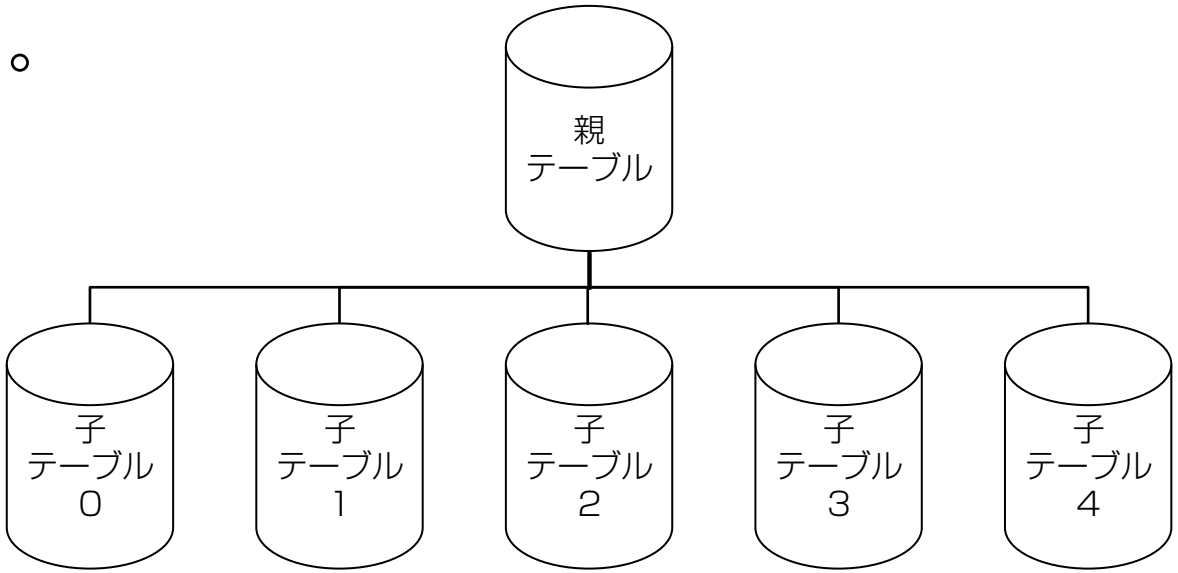
UNLOGGEDテーブル

- 三谷さんの資料から。



継承テーブルでのINDEX利用

- 親の「テーブル定義」が子に継承される。
- 継承テーブルはテーブル分割(パーティショニング)でよく使われる。



一般的なパーティショニングテーブル構造
月別にテーブルをつくるとか。

継承テーブルでのINDEX利用

- ソートを行うときにも子テーブルにあるインデックスを使用。

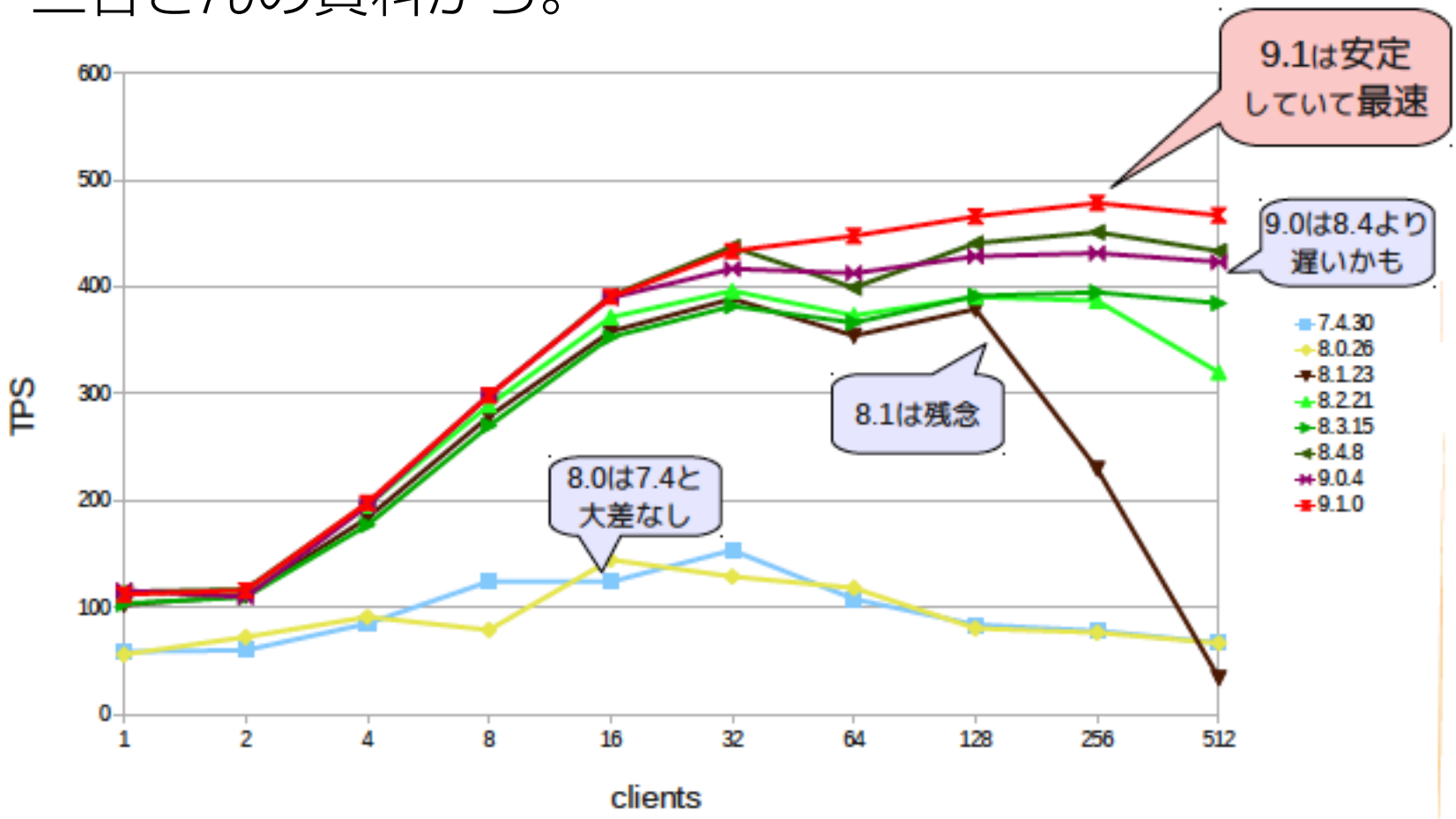
```
<9.0.4>
test=# explain select max(sales) from parent where area_id = 0;
                              QUERY PLAN
-----------------------------------------------------------------------------
Aggregate (cost=228.77..228.78 rows=1 width=4)
  -> Append (cost=0.00..203.75 rows=10006 width=4)

<9.1.0>
test=# explain select max(sales) from parent where area_id = 0;
                              QUERY PLAN
-----------------------------------------------------------------------------
Result (cost=0.06..0.07 rows=1 width=0)
InitPlan 1 (returns $0)
  -> Limit (cost=0.01..0.06 rows=1 width=4)
```

パフォーマンス

- FULL OUTER JOIN でhash joinがつかえる。
- fsync要求の統合化。
  まとめて呼ぶようにした。
- commit_siblingsの性能改善。
  commit_delay パラメータによるWALファイルへの書き出し遅延をするために必要な同時にオープンされているトランザクションの最少数。WALファイルへの書き出しを遅延させても、同時書き出す対象のトランザクションが無い可能性が高い場合に、無駄な遅延を発生させないことを目的。

パフォーマンス

- 三谷さんの資料から。



PostgreSQL9.1で早くなったの？

- デフォルト設定では、早くない。
  初期値のままでは、8.3あたりが最速？？
- 状況がハマるとき、しかるべく設定したときに性能アップの効果がある。
  ちゃんとチューニングしましょうね。

賢いCLUSTERコマンド

- テーブル統計情報をみて、最適な再編成の方式を自動的に選択
  - 新方式は「Seq Scan + Sort」
  - 従来は「Index Full Scan」
- 速い、ただし、ほぼクラスタ済みデータ(断片化が少ないデータ)なら旧方式有利
- 新方式は一時ファイルを大きく使う。

contrib/sepgsql

- データベースオブジェクトに強制アクセス制御
  アクセスの可否のチェックをSE-Linuxと統合
  各SQLについてSE Linuxに内部的に問い合わせるので
  SELinuxのポリシーとして設定を行う
- SE-Linux同様のラベルベース強制アクセス制御。
  - SE-Linuxになれている。アクセス制御は1カ所でやりたい
  という人にお奨めかも。
  - configer optionに –with-selinux

さまざまな文字列関数の追加 format()等。

- format('%s and %s', 'abc', '123')
  - printf タイプのフォーマッタ
  - SQLリテラル、SQL識別子のエスケープに対応
- concat(a,b,c, ..)、concat_ws()
  - || 演算子と同じだけど、某DBとの互換性的に
- reverse()
- left(),right()
  - 右から何文字、左から何文字

更新を行えるWITH句 MERGEの代用ができる。

- SQLの結果を一時的に保存しておく機能
- RETURNINGとの組み合わせで、複数の処理が連続してできる。
  - DELETEした内容を別テーブルへINSERT
  - データがあれば、UPDATEなければINSERT

```
WITH tmp_table (id, val) AS (DELETE FROM t RETURING *); // 削除した結果をtmp_tableに一時保存
INSERT INTO archive_table SELECT * FROM tmp_table ; // tmp_tableの内容をarchive_tableに保存
```

```
WITH val AS (SELECT 100 as id,  'AAA'  as v),
        upd AS(UPDATE t SET v = val.v FROM val WHERE t.id = val.id RETURING t.id)
INSERT INTO t SELECT * FROM val WHEREid NOT IN (SELECT id FROM upd);
```

カラム単位のロケール指定

- 検索毎、カラム毎でCOLLATE(言語を考慮した文字比較) 設定ができる
  - 8.3までは、データベースクラスタ単位
  - 8.4データベース単位
- glibcの日本語ロケールは例によって使えないので Cのままで

Contrib/ pg_trgmの拡張

- 類似文字列の高速検索及び、トリグラム一致に基くテキスト類似度の決定に関する関数と演算子も提供。
- LIKE検索で中間一致にもインデックス検索
  - Tri-gram 「hello」-> 「hel」 「ell」 「llo」 「lo」
  - 9.1でLIKE,ILIKEに対応
- 日本語に優しくない
  - マルチバイト対応にはソース上のフラグ変更が必要
  - 2文字検索が上手くいかない？文化？(in of by on …)

```
CREATE INDEX ON docs USING gin (doc gin_trgm_ops); //gin or gist index にgin_trgm_opsオプション
SELECT * FROM docs WHERE doc LIKE　‘%foo%’;
```

KNN GiST インデックス

- K-NN(k Nearest Neighbor)検索：空間上に指定された地点に近接するオブジェクトを、空間データベースの中からｋ個求める。
- GiSTインデックスを距離が近いものトップリストを出す検索につかえる
  - POINT型、Geonetry型など
  - 家から近い歯医者さん10件とかすぐだせる。

```
SELECT coordinates, (coordinates <-> '5.0,5.0' :: point) AS dist FROM spots ORDER BY dist ASC
LIMIT 10;
```

CREATE EXTENSION

- 拡張モジュールをインストールする命令
- 所定の作法を守って書かれた拡張モジュールを、SQLコマンドで管理できる。
  - ビルドと dll / .so のインストールはあらかじめ必要。
- 拡張モジュールのバージョンアップに対応。

```
CREATE EXTENSION hstore ;
DROP  EXTENSION hstore ;
```

ちょこっとだけ9.2　2012年、7～9月リリース予定

- http://www.postgresql.org/docs/devel/static/index.html

  PostgreSQLは、開発とともにドキュメントの更新をしている。
- レプリケーションのカスケード対応　予定
- recovery.conf とpostgresql.confの統合　予定
- Index Only Scan　予定
- Range Types : 範囲型
- pg_export_snapshot スナップショット