PostgreSQL最新情報

OSC 2020 Online/Nagoya

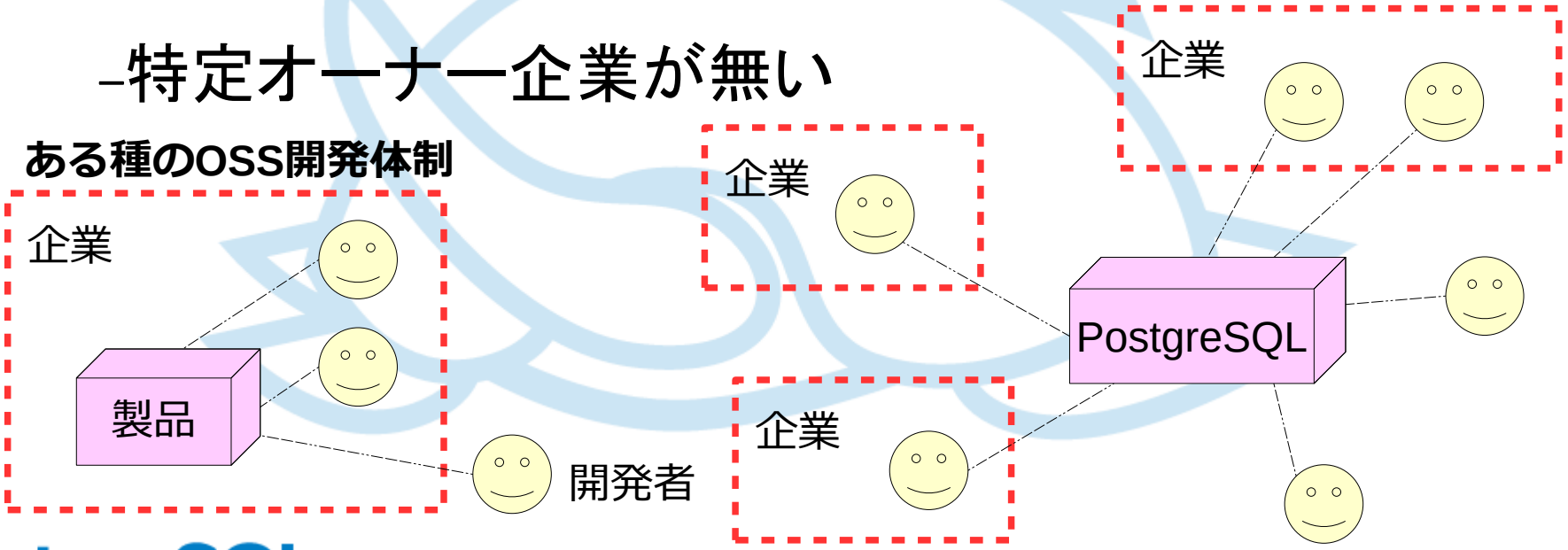
2020-05-30

日本PostgreSQLユーザ会

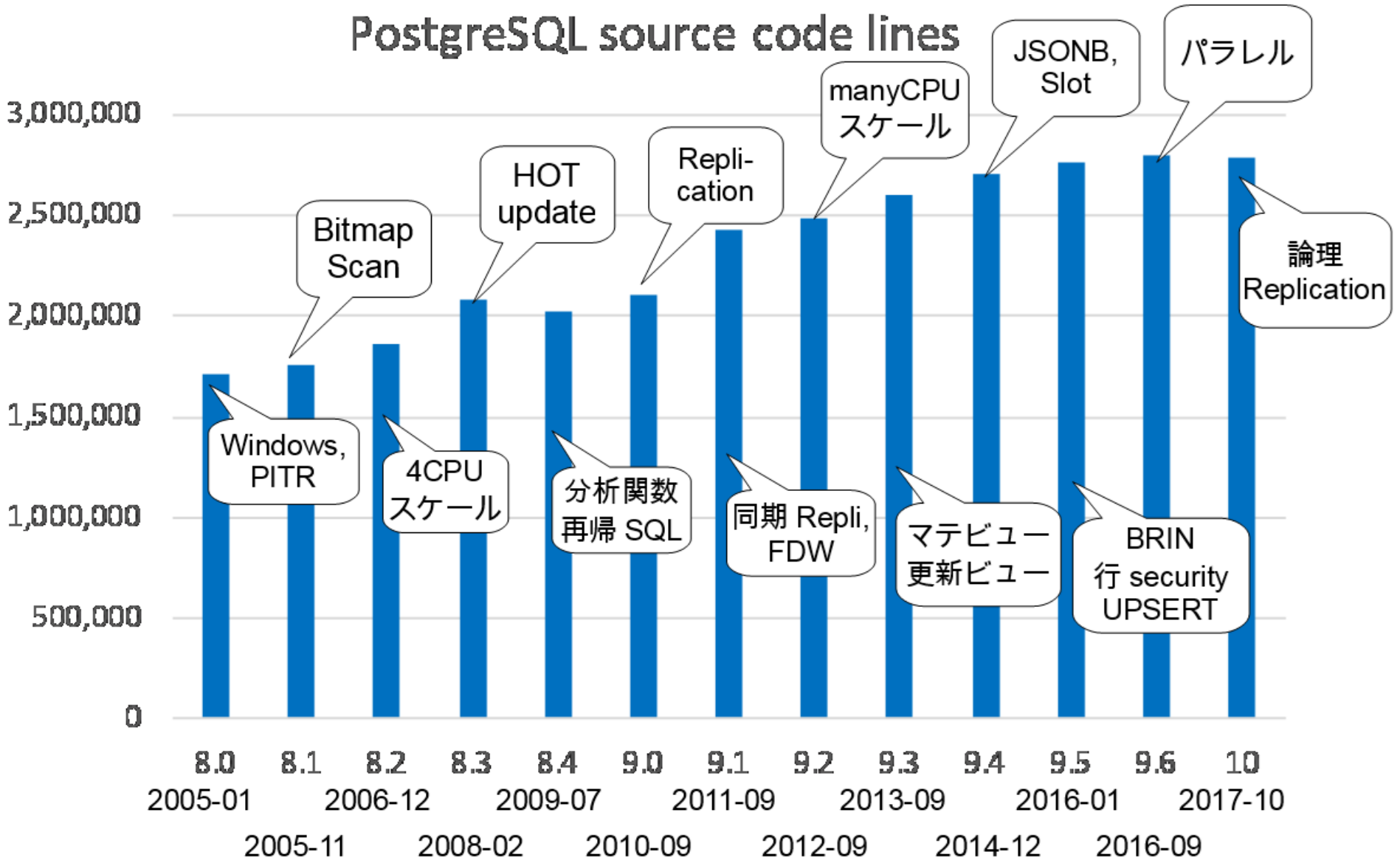
名古屋支部 福島 克輝

PostgreSQLとは

- 多機能、高性能、かつオープンソースの
リレーショナルデータベース管理システム
  - INGRES(‘70),POSTGRES(’80)由来の歴史
（カリフォルニア大学バークレイ）
  - BSDライセンス
  - 特定オーナー企業が無い



安定したリリース



PostgreSQL
http://www.postgresql.jp

最近のリリース

2020年5月14日（バグ修正リリース）

- 12.3リリース
- 11.8リリース
- 10.13リリース
- 9.6.18リリース
- 9.5.22リリース
- 9.4系は前回リリースでメンテナンス終了

2020年5月21日　PostgreSQL 13 Beta1　リリース！

バグ修正リリースの内容

- セキュリティ上の問題

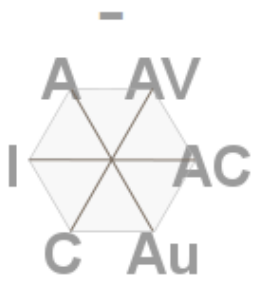
CVE-2020-10733: Windows インストーラが、制御されていないディレクトリから実行ファイルを実行する。影響を受けるバージョン: 9.5 - 12. セキュリティチームはサポートされていないバージョンをテストしていませんが、この問題はPostgreSQL 9.5以前にも存在していました。

PostgreSQLのWindowsインストーラは、完全修飾パスを持たないシステム提供の実行ファイルを起動します。インストーラがロードするディレクトリや現在の作業ディレクトリにある実行ファイルが、意図した実行ファイルよりも優先されます。これらのディレクトリのいずれかにファイルを追加する権限を持つ攻撃者は、これを利用してインストーラの管理者権限で任意のコードを実行することができます。

Windows 版 PostgreSQL のインストーラの処理に特権を奪われる問題

ID	35347	登録日	2020/05/15	最終更新日	---
危険度	H	影響範囲	ローカル	影響プログラム名	PostgreSQL
CVE	CVE-2020-10733				

CVSS



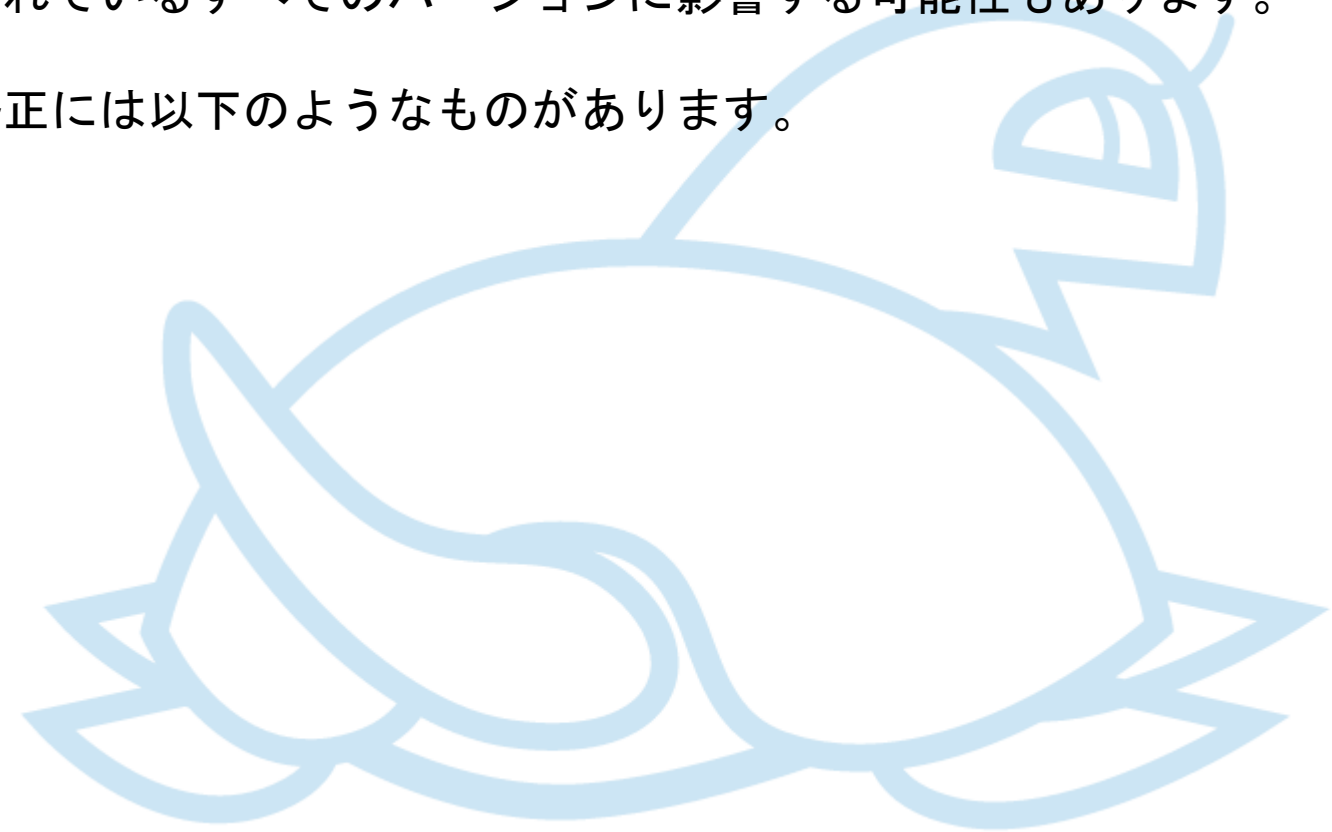
攻撃元 (AV)	ローカル (L)	隣接 (A)	ネットワーク (N)
攻撃成立条件の難易度 (AC)	難しい (H)	やや難しい (M)	簡単 (L)
攻撃前の認証要否 (Au)	複数回 (M)	一回 (S)	不要 (N)
機密性への影響 (C)	なし (N)	部分的 (P)	全面的 (C)
完全性への影響 (I)	なし (N)	部分的 (P)	全面的 (C)
可用性への影響 (A)	なし (N)	部分的 (P)	全面的 (C)

出典：SIDfm セキュリティホール危険度警告センター

バグ修正リリースの内容（続き）

バグ修正と改善
今回のアップデートでは、過去数ヶ月間に報告されていた75以上のバグも修正されています。これらの問題の中にはバージョン 12 のみに影響するものもありますが、サポートされているすべてのバージョンに影響する可能性もあります。

これらの修正には以下のようなものがあります。



バグ修正リリースの内容（続き）

- 生成された列の出力がテーブル上の物理的な列の正確なコピーである場合、例えば式が自身の入力を返す関数を呼び出した場合など、テーブル内のデータをクラッシュさせたり破損させたりする可能性がある問題を含む、生成された列に関するいくつかの修正。
- SET STORAGEディレクティブがテーブルのインデックスに確実に伝搬されるようにすることを含む、ALTER TABLEのためのいくつかの修正。
- 別のセッションが同じオブジェクトを削除している間にDROP OWNED BYを使用した場合の潜在的な競合状態を修正しました。
- ROW トリガを継承している場合にパーティションを切り離すことができるようになりました。
- REINDEX CONCURRENTLYに関するいくつかの修正、特にREINDEX CONCURRENTLY操作が失敗した場合の問題を修正しました。
- COLLATE がパーティションバインド式の中の照合不可能な型に適用された場合のクラッシュを修正しました。
- 浮動小数点オーバーフロー/アンダーフロー検出における性能低下を修正しました。
- フルテキスト検索、特にフレーズ検索でのいくつかの修正。
- クエリのFROM句で使用されるset-returning関数のメモリリークを修正しました。
- VACUUM VERBOSEの出力に関するいくつかのレポート修正。
- circle型の入力が、ドキュメントで指定されている形式(x,y),rを受け付けるようにしました。

バグ修正リリースの内容（続き）

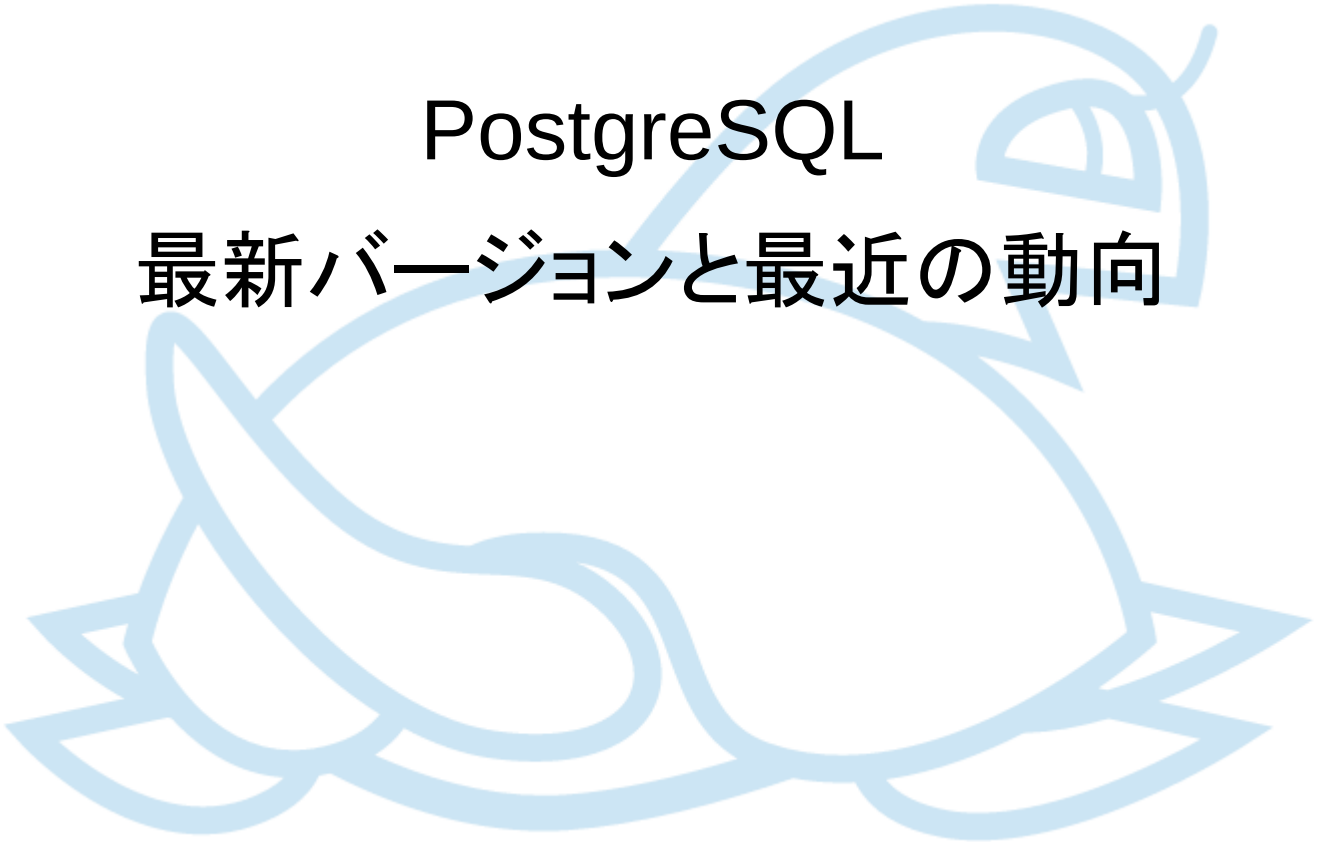
- get_bit()とset_bit()関数が256MB以上のbytea文字列で失敗しないようにしました。
- クラッシュリカバリ中の WAL セグメントのリサイクルが早すぎると、WAL セグメントがアーカイブされる前にリサイクルされてしまう可能性があります。
- リカバリ中にアーカイブストレージから存在しない WAL ファイルを取得しようとすると、関連性のないタイムラインをスキップしてしまうことを回避しました。
- 論理レプリケーションとレプリケーションスロットに関するいくつかの修正。
- synchronous_standby_names設定の変更時に発生したものを含む、同期スタンバイ管理におけるいくつかの競合状態を修正しました。
- GSSAPIサポートに関するいくつかの修正。GSSAPI暗号化を使用する際に発生したメモリリークの修正を含みます。
- pg_read_all_statsロールのメンバがすべての統計情報ビューを読めるようにしました。
- information_schema.triggersビューのパフォーマンス低下を修正しました。
- sslmode=verify-fullを使用した場合のlibpqのメモリリークを修正しました。
- 失敗した接続を再確立しようとしたときの psql のクラッシュを修正しました。
- psqlで、ファイル名の引数のタブ補完を許可しました。
- ALTER ...に対するpg_dumpのサポートを追加しました。
- pg_dumpに対して、RLSポリシーのコメントをダンプしたり、イベントトリガーのリストアを最後まで延期したりするなど、他にもいくつかの修正を行いました。

バグ修正リリースの内容（続き）

- pg_basebackupが有効なtarファイルを生成するようになりました。
- pg_checksumsが、異なるPostgreSQLのメジャーバージョンに属するテーブル空間サブディレクトリをスキップします。
- いくつかの Windows 互換性の修正この更新には、モロッコとカナダのユーコンにおける DST 法の変更に関する tzdata release 2020a と上海の歴史的な修正も含まれています。アメリカ/ゴッドタブゾーンは、現在の英語の使用法を反映するために、アメリカ/ヌークに改名されました。これはまた、initdbの既知のWindowsタイムゾーン名のリストを、最近追加されたものを含むように更新しました。

www.DeepL.com/Translator（無料版）で翻訳しました。

PostgreSQL
最新バージョンと最近の動向



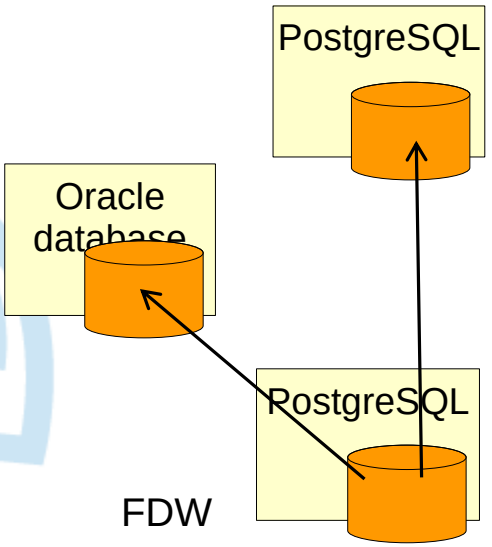
現在のPostgreSQLと周辺

SQL機能的には： .ANSI SQL:2011コア 概ね準拠 .各種の組み込み言語 .地理情報システム .JSON型 .他DB連携（FDW）	クラスタ構成： .ストリーミングレプリケーション .論理レプリケーション .HAクラスタ .MPPクラスタ(shared nothing) .RAC型(shared disk)は不可
性能的には： .OLTPの多CPUスケール 参照→96コアでもスケール 更新→48コアでもスケール .パラレルクエリ対応拡大 .パーティショニング	運用支援など： .ログ/状態の集積分析ツール pg_statsinfo / pg_badger .クライアントツール PgAdmin4 Etc. .AWS / Azure で対応 .Aurora

現在のPostgreSQLと周辺

SQL機能的には：
- ANSI SQL:2011コア おおむね準拠
- 各種の組み込み言語
- 地理情報システム（PostGIS）
- JSON型
- 他DB連携（FDW）

PL/pgSQL
PL/Perl
PL/Python
PL/Tcl
PL/v8
PL/R




```
db=# SELECT * FROM t1;
 id |                    info
----+-----------------------------------------
  1 | {"DBMS": "PostgreSQL", "Version": "10.1"}
(1 row)

db=# SELECT * FROM t1,
      LATERAL jsonb_to_record(t1.info) AS r ("DBMS" text, "Version" text);
 id |                    info                     |    DBMS    | Version
----+---------------------------------------------+------------+---------
  1 | {"DBMS": "PostgreSQL", "Version": "10.1"} | PostgreSQL | 10.1
(1 row)
```

現在のPostgreSQLと周辺

性能的には：
.OLTPの多CPUスケール
参照→96コアでもスケール
更新→48コアでもスケール
.OLAPむけに
.パラレルクエリ対応拡大、
.パーティショニング

対応範囲	9.6	10
シーケンシャルスキャン	○	○
入れ子ループ結合	○	○
ソート	○	○
インデックススキャン	×	○
マージ結合	×	○
ハッシュ結合	×	○
サブクエリ対応	×	○

11 でアペンドなども対応
JIT対応

CPUスケール（参照系）

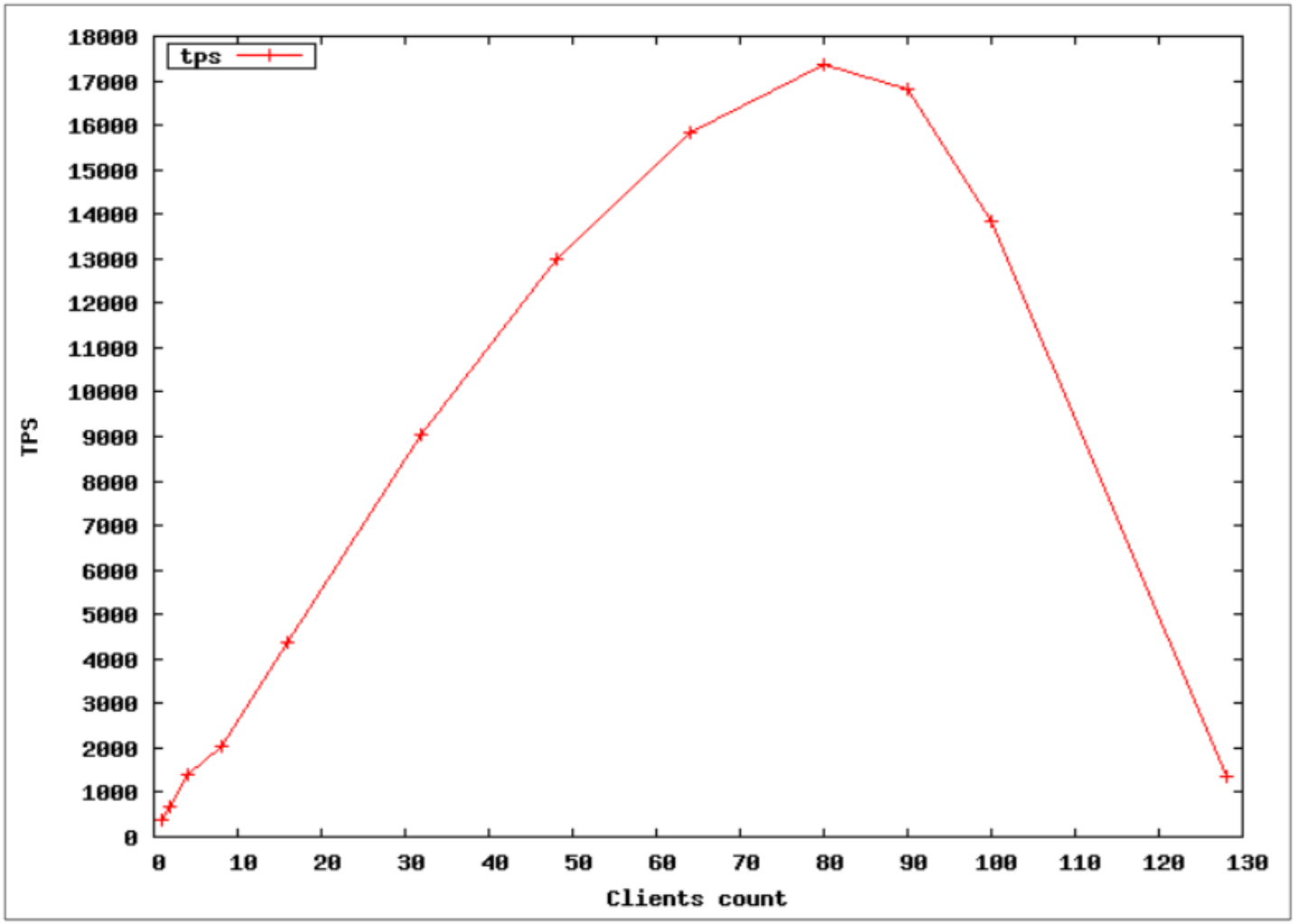
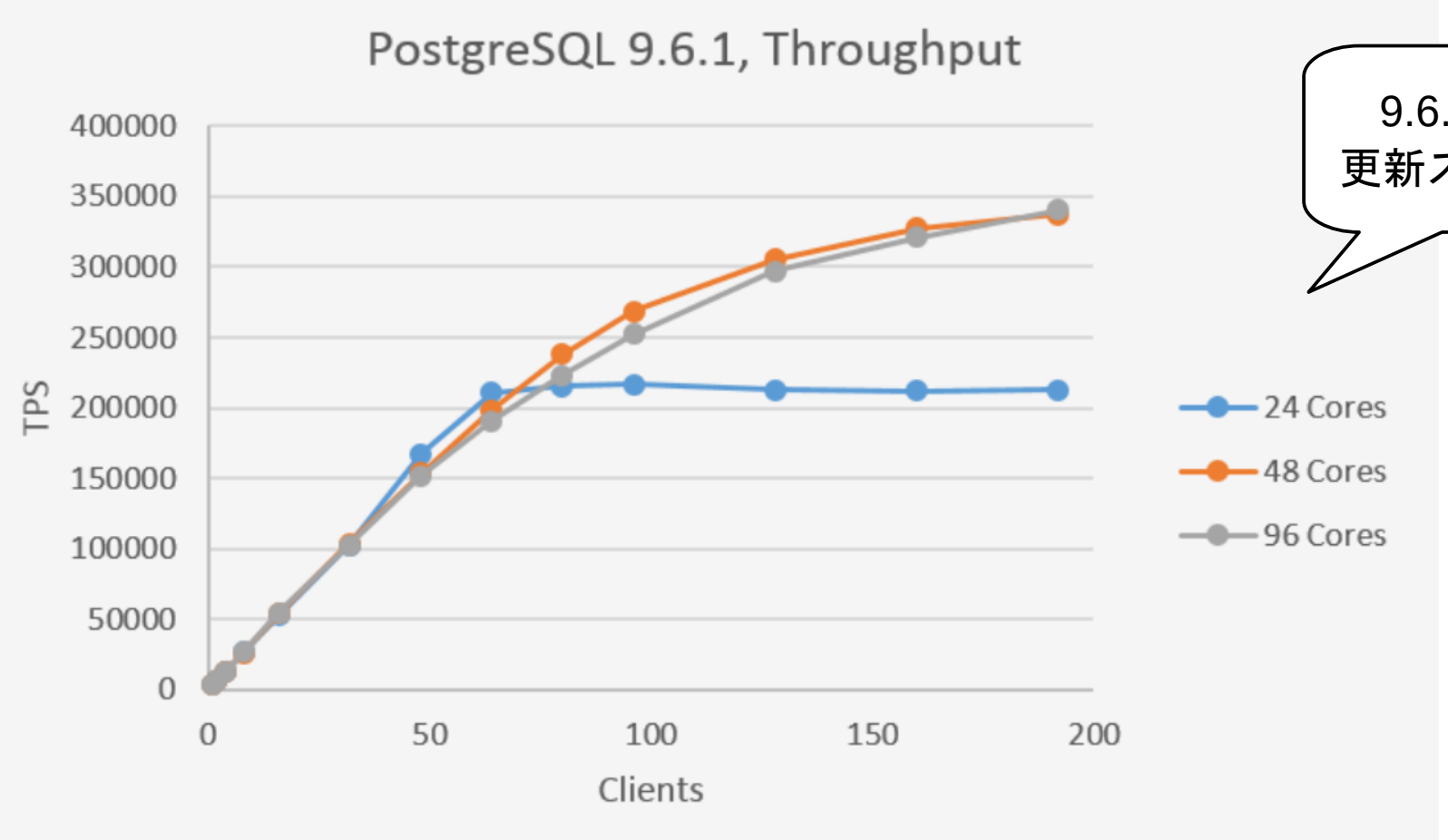


図 2.1: 負荷の高い参照系スクリプトでクライアント数を変動させたときの結果 tps

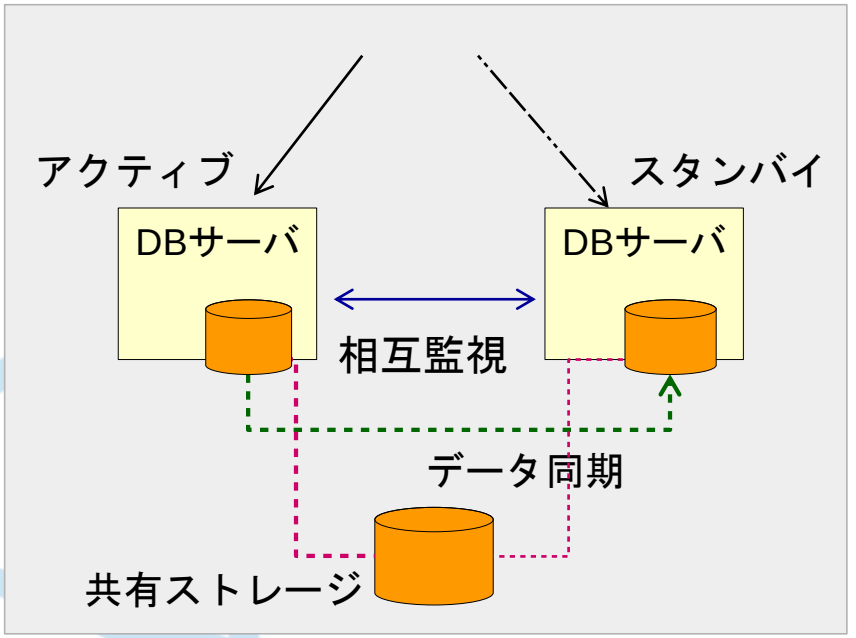
9.2.xの段階で
参照はスケール

CPUスケール（更新系）

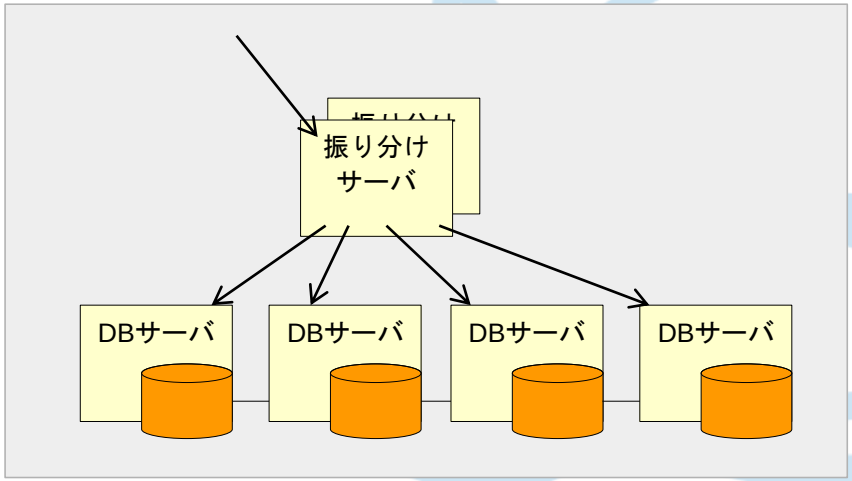


2016年度PGECons WG1成果資料より

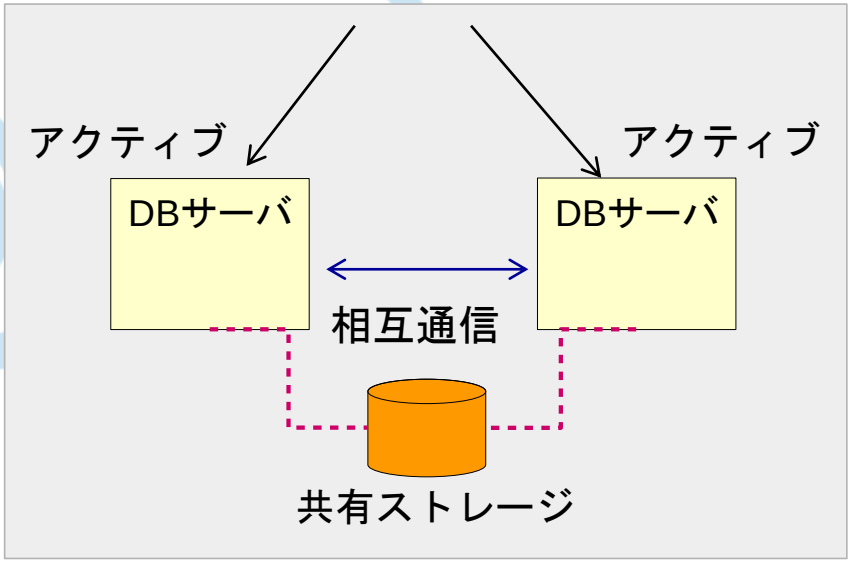
クラスタ構成：
- ストリーミングレプリケーション
- 論理レプリケーション
- HAクラスタ
- MPPクラスタ(shared nothing)
- RAC型(共有ストレージ)は不可



HA（高可用性）クラスタ

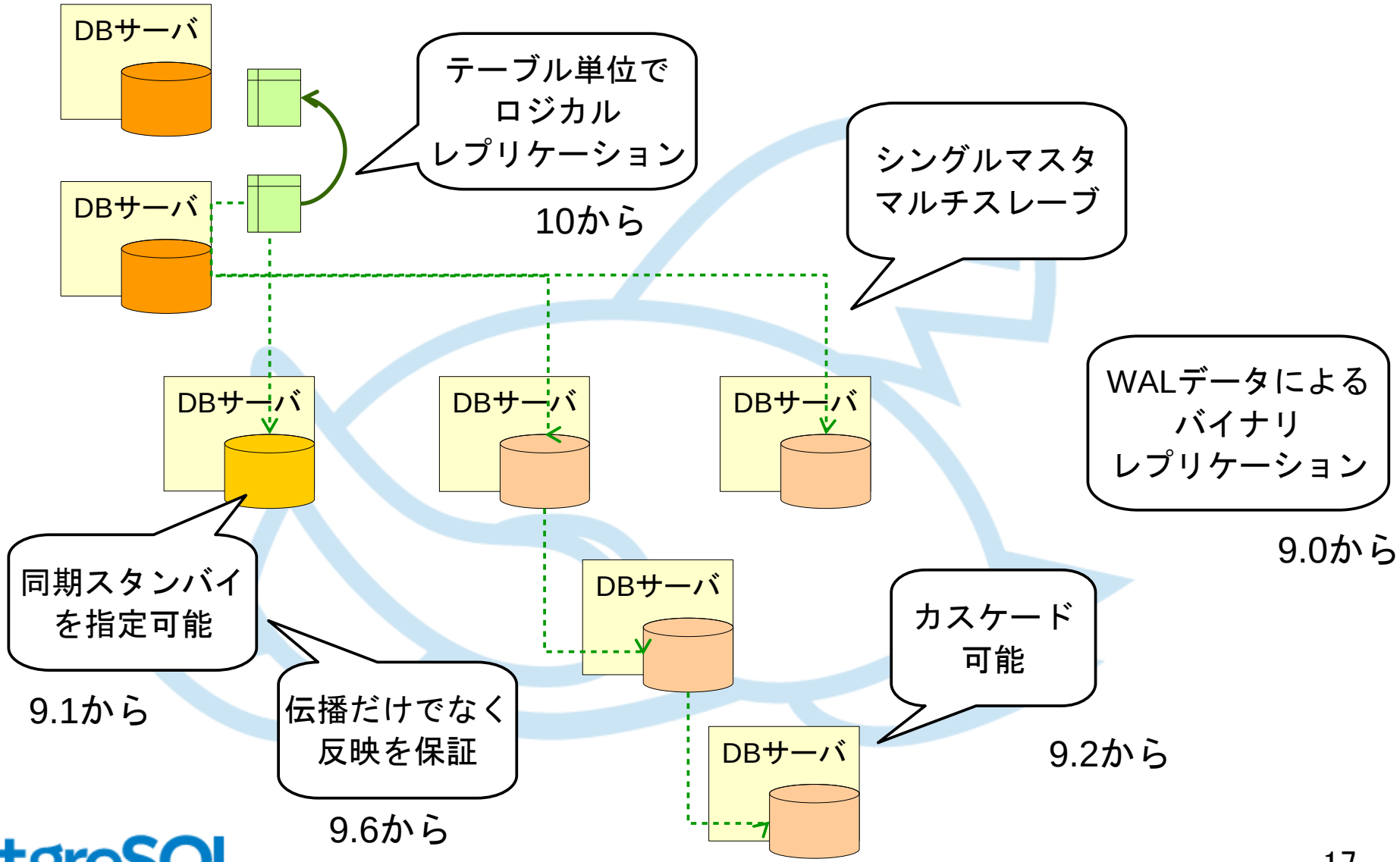


MPPクラスタ



RAC型

レプリケーション機能の進展



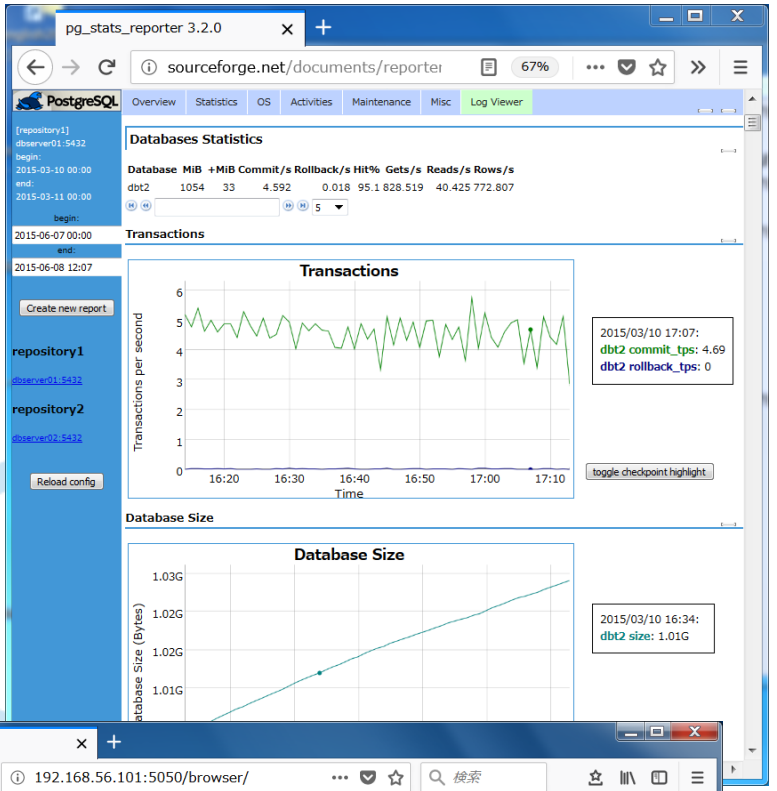
PostgreSQL
http://www.postgresql.jp

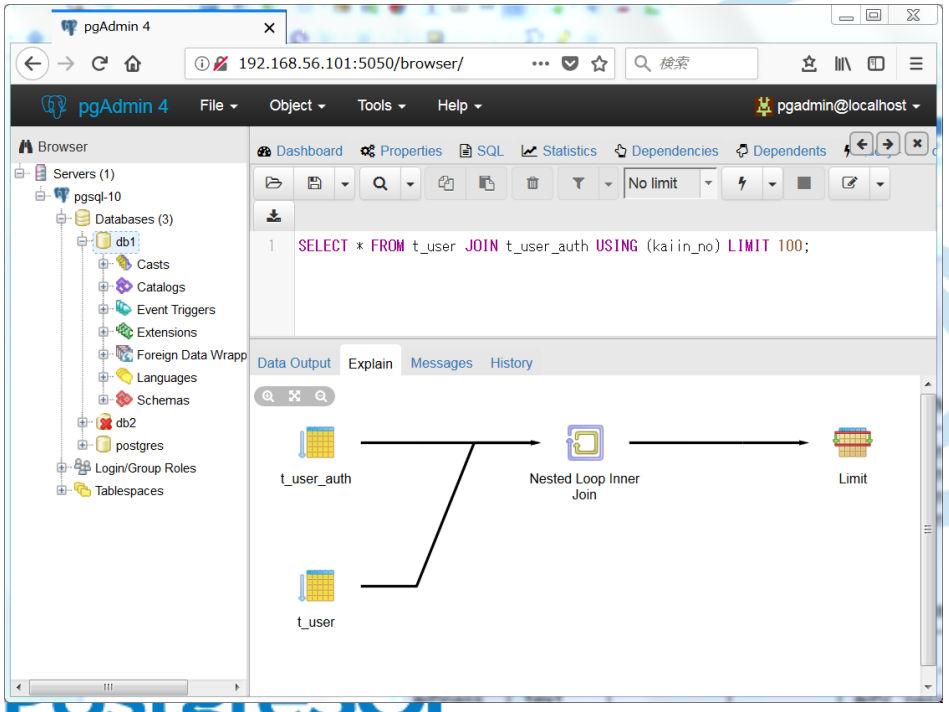
運用支援など：

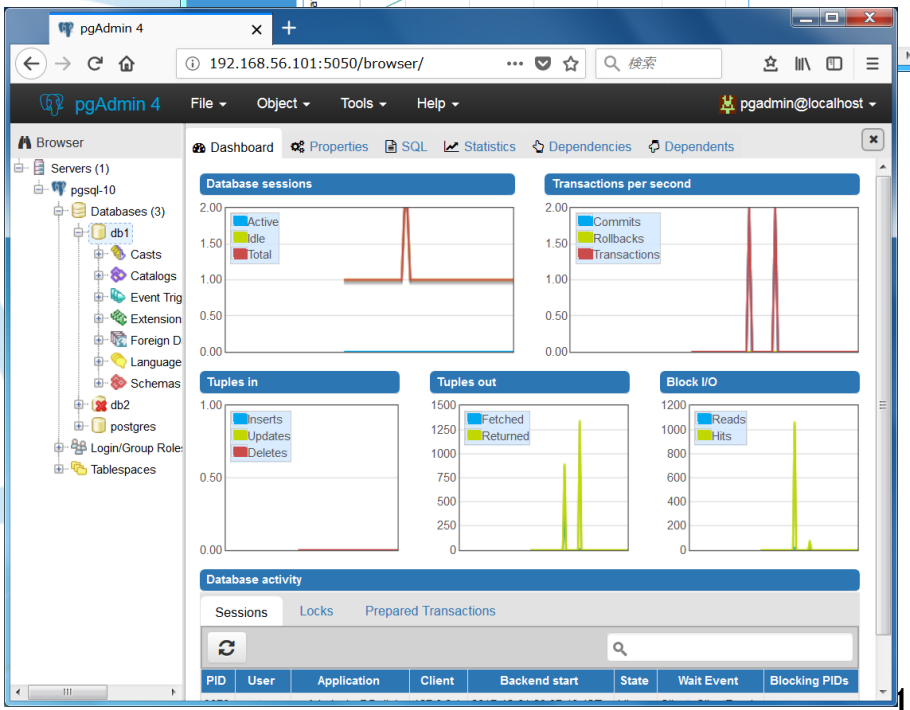
- ログ/状態の集積分析ツール

pg_badger / pg_statsinfo

- クライアントツール
- PgAdmin4 / SI Object Browser
- AWS / Azure で対応
- Aurora







PostgreSQL 10 以前の拡張

- レプリケーション
  - Streaming Replication(9.0標準装備～)
  - Logical Replication(9.4基盤機能、10標準装備)
- JSONデータ型 (9.2～)
  - JSONB +内部要素検索インデックス(9.4～)
- Foreign Data Wrapper (9.1～)
- 書き込み対応(9.2～)、各種プラン対応改良、
  - パーティションに組み込み(9.4～)
- パラレルクエリ (9.6～)
- 多CPUコアにおける性能スケール(9.2、9.5)

PostgreSQL 10の主な新機能

- **ロジカルレプリケーション**

テーブル単位のレプリケーションが可能に、DBのアップグレードにも利用可能です。

- **ネイティブテーブルパーティショニング**

パーティショニング用の構文が用意されるようになりました。INSERTの子テーブルへの振り分けをトリガで行う必要が無くなり、INSERTのパフォーマンスが改善しました。

- **パラレルクエリの改善**

インデックススキャン、ビットマップスキャン、マージ結合、ハッシュ結合に対応しました。

- **同期レプリケーションのクォーラルコミット**

コミット時、いずれかN台のスタンバイに反映したら同期完了という条件を与えることができます。

PostgreSQL 11の主な新機能その１

- **JITコンパイル**

SQL実行において特定の問い合わせ式の実行を高速化するJust In Time コンパイルに対応しました。大量データに対する分析の問い合わせで性能アップが期待できます。WHERE句、ターゲットリスト、集約、投影、一部の内部操作の実行を高速化します。

- **PROCEDURE**

SELECT で呼び出すFUNCTIONの他に、CALL コマンドで実行することができる、戻り値を持たないPROCEDUREに対応します。PROCEDUREではトランザクション制御も可能です。 PROCEDUREはCREATE PRODEDUREコマンドにて作成されます。

PostgreSQL 11の主な新機能その２

- **パーティショニングの改善**

様々な点でパーティショニングが強化されました。いままでのリストパーティションやレンジパーティションに加えてHASHパーティショニング対応しました。キー列のパーティションを跨ぐ更新、パーティションテーブルに対するインデックスやプライマリキーの作成、パーティション指向のプランナ動作、などが含まれます。その他パーティションキーと一致しないデータを格納するデフォルトパーティション機能が追加されています。

- **パラレルクエリの改善**

これまでのシーケンシャルスキャンやハッシュジョインに加えて、パラレル動作しなかった様々なケース（アペンド等）に対応しました。

PostgreSQL 12の主な新機能その１

- **JSON/SQL Pathをサポート**

JSONから要素を抽出する記述方式 にSQL/JSON Pathを利用できるようになりました。JSON PathはSQL:2016で定められているSQL標準の方式です。以前もJSONの要素を抽出する機能はありましたが、独自の関数・演算子を使うものでした。

- **生成列をサポート**

テーブルに指定した式の値が自動的に格納される列を定義できるようになりました。これを生成列（Generated Column）と呼びます。CREATE TABLEやALTER TABLEの列定義でGENERATEDオプションを使って設定します。

- **各種インデックスとパーティショニングの改善**

Btree、GIN、GiST、SP-GiSTの各インデックスでそれぞれ動作の効率化が実現されています。また、REINDEXコマンドをテーブルの参照・更新処理と並行して実行できるようになったり、GiSTがINCLUDE句に対応したり、SP-GiSTで近傍探索対応する機能拡張も行われました。パーティショニングではデータ投入時の性能改善と外部キー制約被参照側への対応などが加わっています。

PostgreSQL 12の主な新機能その２

- **COLLATE（文字照合）の拡張**

ICUライブラリを使った文字照合で、アクセント記号有無や大文字小文字を同一視する指定が可能になりました。日本語では平仮名・カタカナ同一視ができます。これまで特定の構文や演算子（ILIKEなど）で同様の機能が提供されていましたが、COLLATEとして実現することもできるようになります。

その他、後からのページチェックサム有効/無効の切り替え、複数列にまたがる最頻値プランナ統計情報、WITH句の最適化の改善などが適用されています。

PostgreSQL 13の主な新機能(機能性)

- PostgreSQLの標準インデックスであるB-treeインデックスにおける、重複データの処理方法が改善されました。この機能強化により、特に繰り返し値を含むインデックスのインデックスサイズを縮小し、検索速度を向上させることができます。
- PostgreSQL 13では、増分ソートが追加されました。これは、問い合わせの以前の部分からソートされたデータが既にソートされている場合に、データのソートを高速化するものです。さらに、OR句やIN/ANY定数リストを持つ問い合わせでは、拡張統計情報（CREATE STATISTICSで作成された統計情報）を使用することができます。
- このリリースでは、PostgreSQLのパーティショニング機能にさらに多くの改良が加えられました。パーティショニングされたテーブルがBEFORE行レベルのトリガをサポートするようになり、パーティショニングされたテーブルは、個々のパーティションを公開することなく、論理レプリケーションを介して完全にレプリケートできるようになりました。

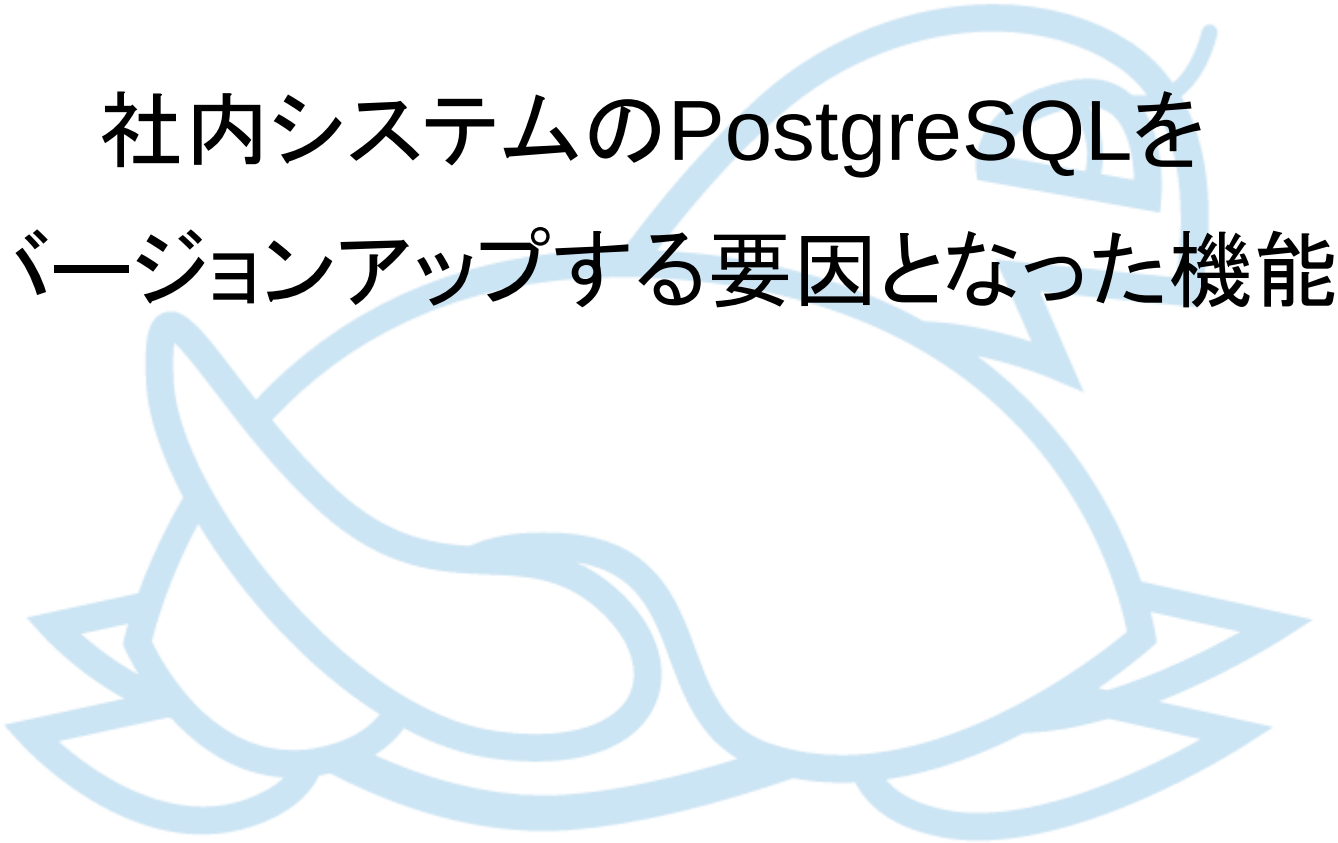
PostgreSQL 13の主な新機能(管理面)

- VACUUMコマンドでインデックスを並列に処理する機能が追加されました。この機能は、VACUUMコマンドの新しいPARALLELオプション（またはvacuumdbの--parallel）を使用してアクセスすることができ、インデックスのバキューム処理に使用する並列ワーカーの数を指定することができます。
- reindexdbコマンドには新しい--jobsフラグによるインデックス再編成の並列性も追加されました。
- PostgreSQL 13では、WALの使用統計やストリーミングベースバックアップの進捗状況、ANALYZEコマンドの進捗状況を追跡できるようになりました。
- pg_dumpの新しいフラグ--includ-foreign-dataが追加され、ダンプ出力に外部データラッパーで参照されるサーバからのデータを含めるようになりました。

PostgreSQL 13の主な新機能(安全面)

- psqlや多くのPostgreSQL接続ドライバを動かす接続ライブラリであるlibpqには、接続の安全性を確保するためのいくつかの新しいパラメータが含まれています。PostgreSQL 13では、channel_binding接続パラメータが導入されました。これにより、クライアントがSCRAMの一部としてチャンネルバインディング機能を必要とすることを指定できます。さらに、パスワードで保護されたTLS証明書を使用しているクライアントは、sslpasswordパラメータを使用してパスワードを指定できるようになりました。PostgreSQL 13では、DER符号化された証明書のサポートも追加されています。
- PostgreSQLの外部データラッパー(postgres_fdw)は、他のPostgreSQLクラスタに接続するために証明書ベースの認証を使用する機能など、接続の安全性を確保するためのいくつかの機能強化を受けました。さらに、非特権アカウントがパスワードを使用せずにpostgres_fdwを介して他のPostgreSQLデータベースに接続できるようになりました。

社内システムのPostgreSQLを
バージョンアップする要因となった機能



PROCEDURE
トランザクション制御対応(PG11)

- 戻り値を返さないPROCEDUREに対応
- PROCEDURE内でのトランザクション制御（COMMIT、ROLLBACK）に対応

```
CREATE [ OR REPLACE ] PROCEDURE
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [,
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
```

パーティションのデフォルトテーブル (PG11)

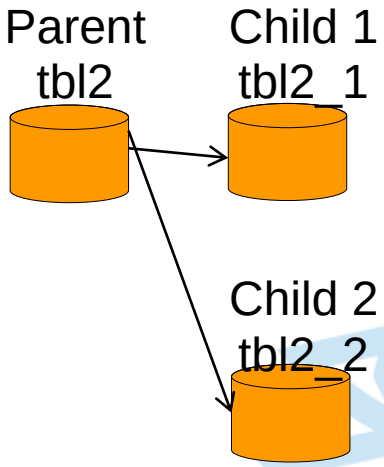
- RANGEやLISTで子テーブルで指定されていない範囲についてはエラーとなる
- デフォルトテーブルは、上記の範囲外のデータを登録するテーブル

Parent	CREATE TABLE tbl2 (id INT, val INT) PARTITION BY RANGE (val);
Child1	CREATE TABLE tbl2_1 PARTITION OF tbl2 FOR VALUES FROM (0) TO (500);
Child2	CREATE TABLE tbl2_2 PARTITION OF tbl2 FOR VALUES FROM (500) TO (1000);
Default	**CREATE TABLE tbl2_default PARTITION OF tbl2 DEFAULT;**

パーティションキー列 UPDATE対応 (PG11)

- パーティションキーに指定したフィールドの、パーティションをまたがる更新に対応

Parent　CREATE TABLE tbl2 (id INT, val INT) PARTITION BY RANGE (val);
Child1　CREATE TABLE tbl2_1 PARTITION OF tbl2 FOR VALUES FROM (0) TO (500);
Child2　CREATE TABLE tbl2_2 PARTITION OF tbl2 FOR VALUES FROM (500) TO (1000);



ID	VAL
1	134
3	445
4	311

ID	VAL
2	512
5	675

ID	VAL
1	134
4	411

ID	VAL
2	512
3	812
5	675

```
UPDATE tbl2 SET val = 812 WHERE ID = 3
```

PostgreSQL
http://www.postgresql.jp

パーティション全体への ユニークインデックス (PG11)

- 親テーブルにALTER TABLEでPRIMARY KEYを設定することが可能（パーティションキーを含む）
- 上記により、パーティション全体でのユニークインデックスが可能

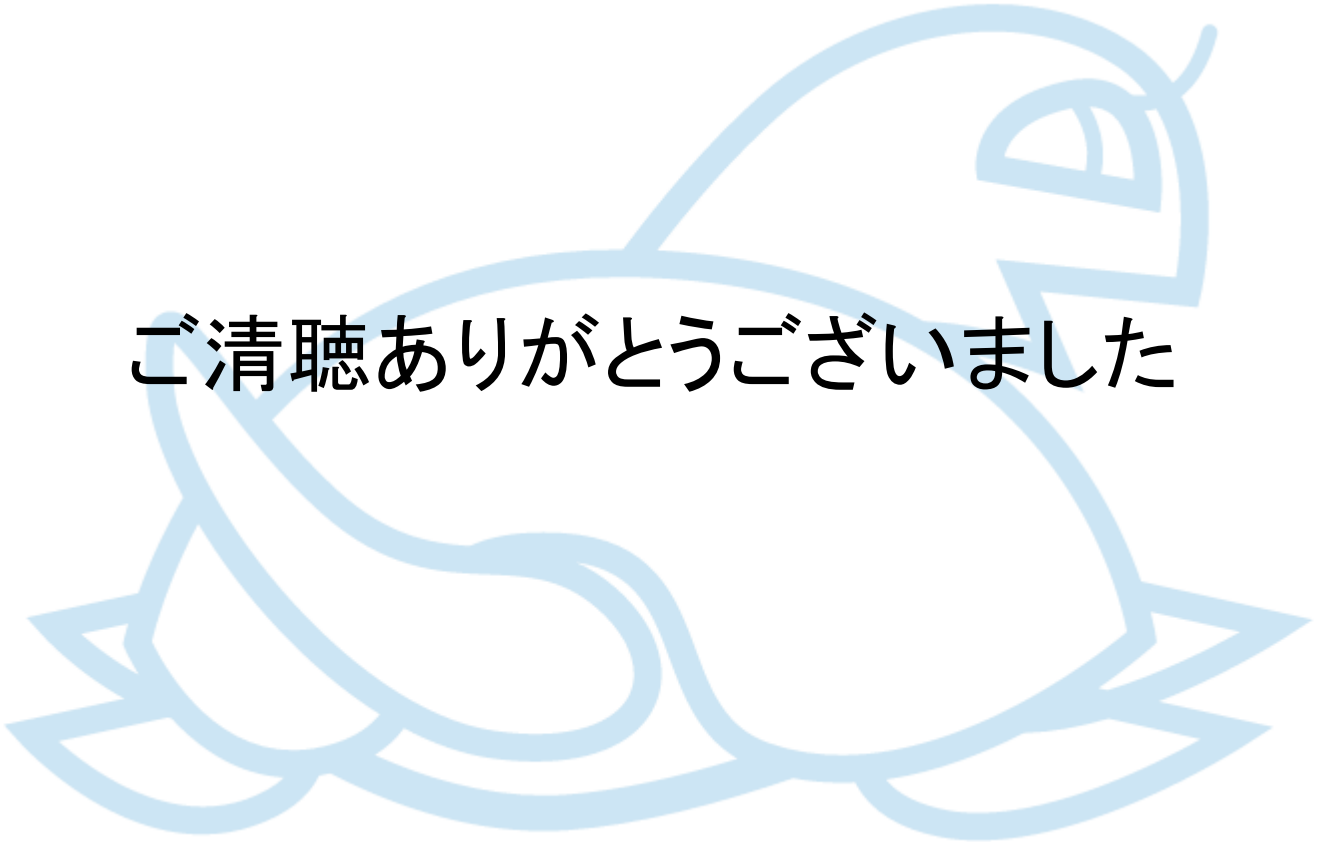
```
Parent    CREATE TABLE tbl2 (id INT, val INT) PARTITION BY RANGE (val);
Child1    CREATE TABLE tbl2_1 PARTITION OF tbl2 FOR VALUES FROM (0) TO (500);
Child2    CREATE TABLE tbl2_2 PARTITION OF tbl2 FOR VALUES FROM (500) TO (1000);

ALTER TABLE tbl2 ADD CONSTRAINT tbl2_pkey  PRIMARY KEY (id, val);
```

以下もOK

```
CREATE TABLE tbl2 (id INT, val INT,
      CONSTRAINT tbl2_pkey PRIMARY KEY (id, val)) PARTITION BY RANGE (val);
```

おしまい



ご清聴ありがとうございました