SRA OSS,INC.

次期バージョン PostgreSQL 10 の新機能とその後の方向性

SRA OSS, Inc. 日本支社
マーケティング部　PostgreSQL 技術グループ
長田 悠吾

db tech showcase OSS 2017
2017-06-16

自己紹介

- 長田 悠吾（ながた ゆうご）
- 所属
  - SRA OSS, Inc. 日本支社
    マーケティング部　PostgreSQL 技術グループ
- 業務
  - PostgreSQL の技術サポート、ときどきコンサルタント
  - PostgreSQL 関連の調査・研究・開発

- 最近
  - PGCon 2017 に参加してきました

PGCon 2017

- 世界最大のPostgreSQL のカンファレンス
  - 日程: 5/23 (Tue) – 5/26 (Fri)
  - 場所: カナダのオタワ大学

- PostgreSQL に関する様々なセッション
  - PostgreSQL 10 の新機能
  - Hacking
  - パフォーマンス
  - 運用
  - 事例
  - ・・・

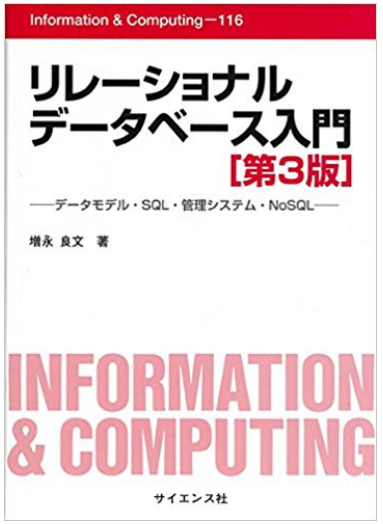
発表もしてきました

Extending View Updatability by a Novel Theory
- Prototype Implementation on PostgreSQL -

Yugo Nagata, Yosifumi Masunaga

ビューの更新可能性を拡張する理論
& PostgreSQL でのプロトタイプ実装

お茶の水大学名誉教授
増永良文先生との共同研究



本日の内容

- PostgreSQL 10 の新機能
- PostgreSQL のその後の方向性について
  - PGCon 2017 の話題
  - 開発コミュニティで提案中・開発中の機能

PostgreSQL

- 代表的なオープンソースのRDBMSの1つ
  - カリフォルニア大学で開発された研究用RDBMSのIngres(1970)を先祖に持つ
- オーナー企業を持たず、コミュニティによる開発が続けられている
  - 年1回のメジャーバージョンアップ
- PostgreSQLライセンスで配布
  - BSDタイプの緩いライセンス

PostgreSQL 10

2017/5/18 に、beta 1 がリリース
（まだ開発中）

バージョン番号について

- 今までは x.y.z 形式
  - 9.6.3
    - 9.6 の部分がメジャーバージョン
      - 機能追加や仕様変更など
    - .3 の部分がマイナーバージョン
      - 不具合修正など
- 今後は x.y 形式
  - 10.0
    - 10 の部分がメジャーバージョン番号
    - .0 の部分がマイナーバージョン番号

PostgreSQL 10 の新機能

- ロジカルレプリケーション
- 宣言的パーティショニング
- パラレルクエリ改善

- その他

ロジカルレプリケーション

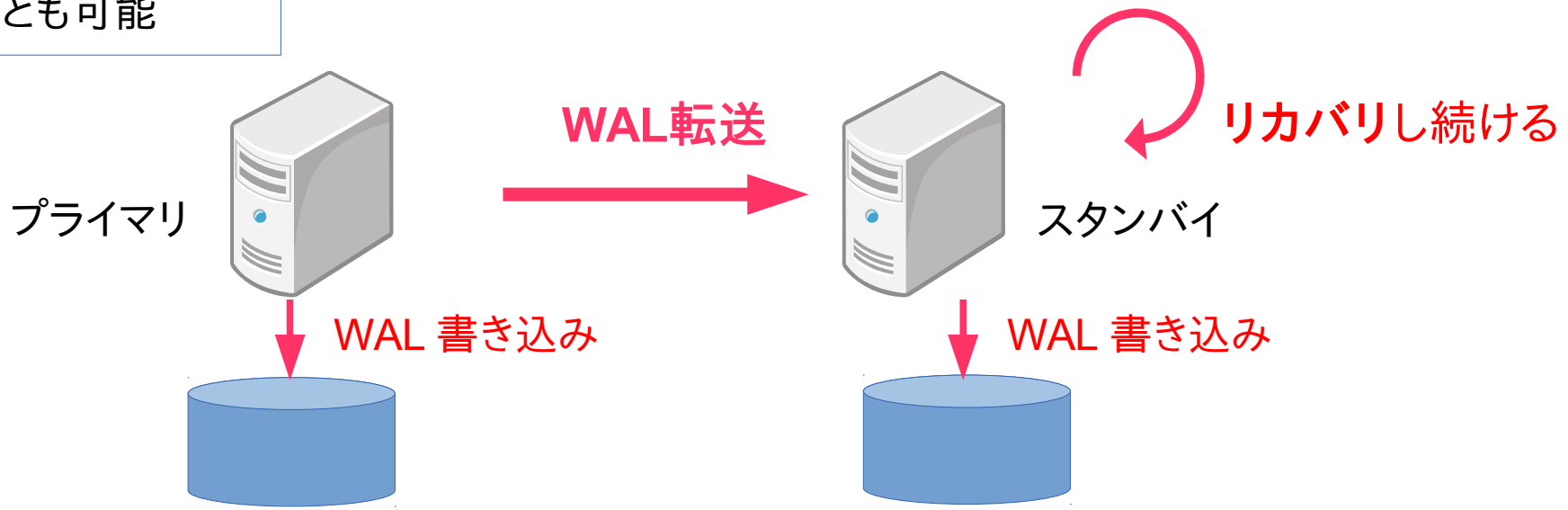
PGCon2017
Logical Replication in PostgreSQL 10
Peter Eisentraut

PostgreSQL のレプリケーション

- ストリーミングレプリケーション（PostgreSQL 9.0 ～）
  - マスタからスレーブにトランザクションログ（WAL）を転送することによりデータの複製を実現
  - 対象はデータベース全体
  - 非同期

スレーブは参照クエリを受け付けることができる（Hot Standby）

マスタへは更新・参照の両方とも可能



ロジカル（論理）vs 物理？

- 従来のストリーミングレプリケーション
  - トランザクションログ（WAL）をスタンバイに転送＆適用
    - ログの内容は「ファイルの変更内容」
    - スタンバイサーバはマスタサーバのビット単位の複製となる

＝　「物理レプリケーション」

- 完全な複製
- スタンバイには一切の物理的変更ができない

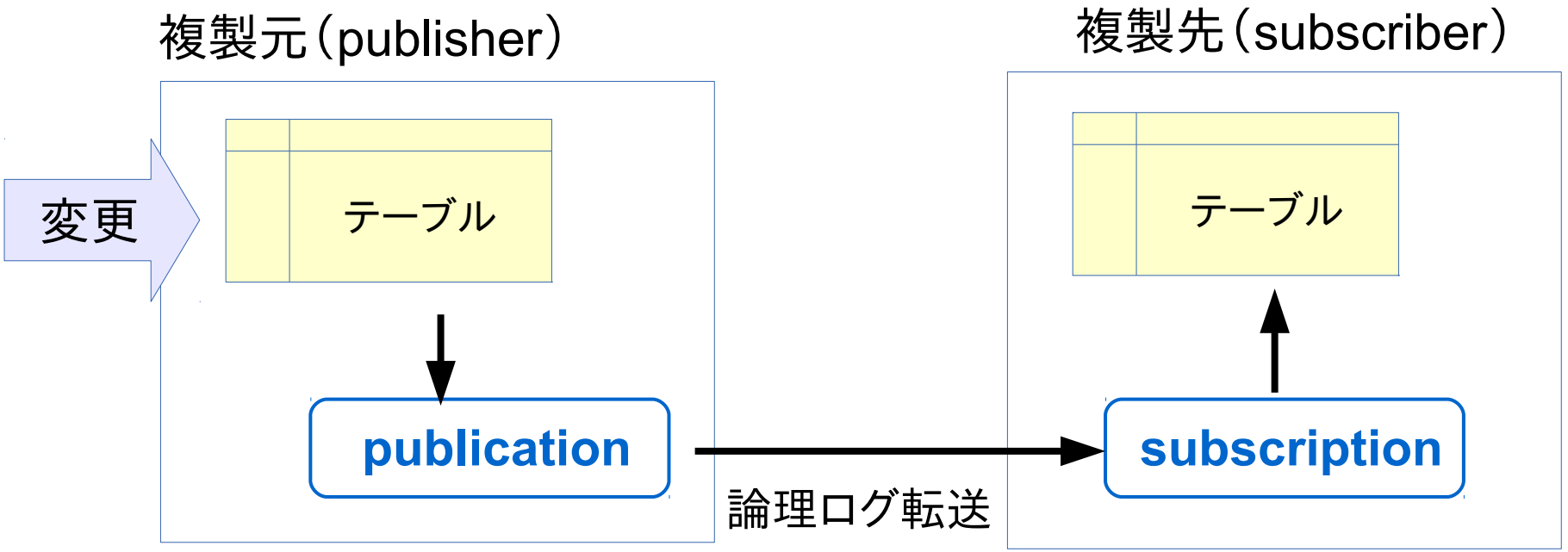
論理レプリケーション

- トランザクションログを「論理的な変更内容」として転送する
  - 例)「テーブル T に、行 (1, 'abc') が挿入されました」

- この形式のログだと
  - 必要なものだけを転送できたり
    - テーブル単位のレプリケーション
    - 挿入だけをレプリケーション
  - 複製先データベースに変更を加えたり
    - 一時テーブルの作成や、インデックスの追加
      → データの処理や解析に使える
  - 複数のサーバのデータを1箇所に集めてみたり
  - 異なるバージョン間での複製をしてみたり

ロジカルレプリケーション概要

- publish/subscribe（出版/購読型）モデル
  - 出版側は特定の購読者を想定せずにメッセージを送信



  - Publication は転送する変更を選択可能（INSERT, UPDATE, DELETE）
  - UPDATE, DELETE のためにはテーブルの行を特定する情報が必要
    - デフォルトではプライマリキーが使用される

publication 作成（複製元）

- 指定のテーブルに publication を作成
  - テーブルのオーナである必要がある

```
CREATE PUBLICATION mypublication
 FOR TABLE users, departments;
```

- 「全てのテーブル」を指定することもできる
  - スーパユーザ権限が必要

```
CREATE PUBLICATION alltables FOR ALL TABLES;;
```

- 転送される変更内容の指定

```
CREATE PUBLICATION insert_only FOR TABLE mydata
    WITH (publish = 'insert');
```

subscription 作成（複製先）

- Publication 名のリストと接続文字列を指定して、subscription を作成
  - スーパユーザ権限が必要
  - 接続ユーザは、接続先に replication 権限が必要

```
CREATE SUBSCRIPTION mysub
 CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb'
 PUBLICATION mypublication, insert_only;
```

- CREATE SUBSCRIPTION 文がコミットされると、テーブルの同期とレプリケーションが開始される
- デフォルトでは、自動的に publisher 側にレプリケーションスロットが作成される
  - Subscription を削除すると、レプリケーションスロットも削除される
  - これらを回避するオプションもある。その場合は手動で作成/削除が必要。

レプリケーションの衝突

- ロジカルレプリケーションでは、複製先でもテーブルの更新が可能
  - 制約の衝突が起こる可能性がある
    - テーブルにプライマリキーか、UNIQUEキーが存在する場合 (UPDATE, DELETE の複製時)
- 衝突が発生するとレプリケーションが停止してしまう
  - 手動で解消が必要
    - 複製先の問題の行を削除する
    - または、複製元のトランザクションをスキップする
      - subscriber 側で pg_replication_origin_advance() という関数を使う

今のロジカルレプリケーションでは できないこと

- CREATE TABLE などの DDL のレプリケーション
  - 対応しているのは INSERT, DELETE, UPDATE
  - TRUNCATE も非対応

- テーブル以外のレプリケーション
  - シーケンスは非対応

- 相互レプリケーション
  - テーブル毎に publisher / subscriber が異なるなら可能
  - 同じテーブルをマルチマスターにはできない

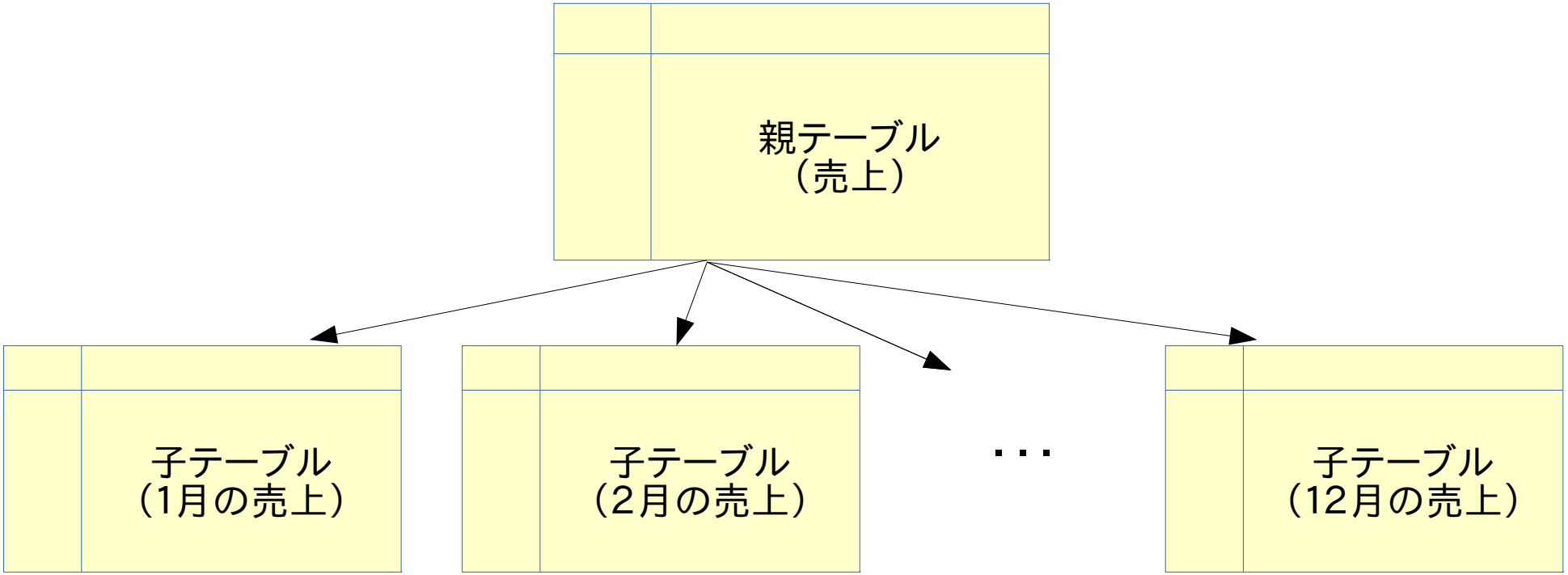
・・・ 将来に期待？

宣言的パーティショニング

PGCon2017
Partition and conquer large data with PostgreSQL 10
Declarative partitioning comes to PostgreSQL
Amit Langote Ashutosh Bapata

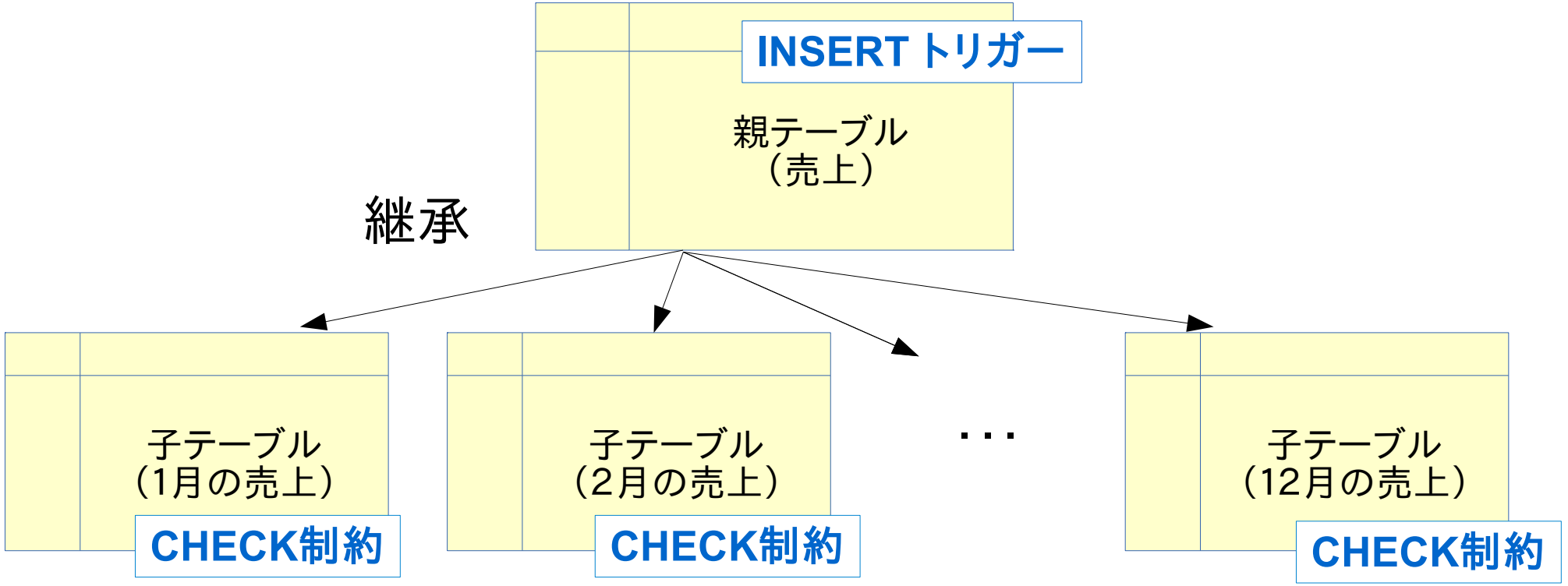
パーティショニング

- 巨大な親テーブルを複数の子テーブルに分割
  - 親テーブルへの検索時、必要な子テーブルのみを検索
  - 挿入時には適切な子テーブルに振り分け



パーティショニング：従来の方法

- PostgreSQL のテーブル継承と制約を使って実現
  - 子テーブルは親テーブルを継承して作成
  - 検索時は「制約による除外」により、必要な子テーブルのみが検索される
  - 挿入時はトリガー関数でレコードを振り分ける



パーティショニング：従来の方法

- 親テーブルを作成して
- 親テーブルを継承する子テーブルを作成して
- 子テーブル全部に CHECK 制約を作成して
- 親テーブルに INSERT トリガーを作成して

・・・・ 面倒くさい

宣言的パーティショニング

- CREATE TABLE 文でパーティショニングが構築可能に
  - 振り分けルールは2種類
    - リスト:　子テーブルにパーティションキーのリストを指定
    - レンジ:　子テーブルにパーティションキーの範囲を指定

- ALTER TABLE 文で、子テーブルの追加／除去が可能
- 階層的なパーティショニングも可能
- 子テーブルに外部サーバのテーブル(FDW)を使用可能
- INSERT の高速化

- テーブル継承の機能を使用して実装されている

パーティショニング構築（レンジパーティショニング）

- 親テーブル
  - パーティションキーには複数のカラムも指定可能
  - もし NOT NULL 制約がなかったら、自動付与される

```
CREATE TABLE measurement (
    logdate         date not null,
    peaktemp        int,
    unitsales       int
) PARTITION BY RANGE (logdate);
```

- 子テーブル
  - 範囲が重なるパーティションが存在していてはいけない

```
CREATE TABLE measurement_y2016m07
    PARTITION OF measurement (
    unitsales DEFAULT 0
) FOR VALUES FROM ('2016-07-01') TO ('2016-08-01');
```

パーティショニング構築（リストパーティショニング）

- 親テーブル
  - パーティションキーは1つのカラムまたは式

```
CREATE TABLE cities (
    city_id      bigserial not null,
    name         text not null,
    population   bigint
) PARTITION BY LIST (left(lower(name), 1));
```

- 子テーブル
  - 格納するキー値のリストを指定する
  - キーが NULL のデータを格納できる子テーブルは1つのみ

```
CREATE TABLE cities_ab
    PARTITION OF cities (
    CONSTRAINT city_id_nonzero CHECK (city_id != 0)
) FOR VALUES IN ('a', 'b');
```

パーティションの追加／削除

- パーティションの追加 (ATTACH)
  - あらかじめ子テーブルは作成しておく
  - キー値の条件に合わないデータが入っているとエラー

```
ALTER TABLE cities
    ATTACH PARTITION cities_ab FOR VALUES IN ('a', 'b')
```

- パーティションの削除 (DETACH)

```
ALTER TABLE measurement
    DETACH PARTITION measurement_y2015m12;
```

今はできないこと

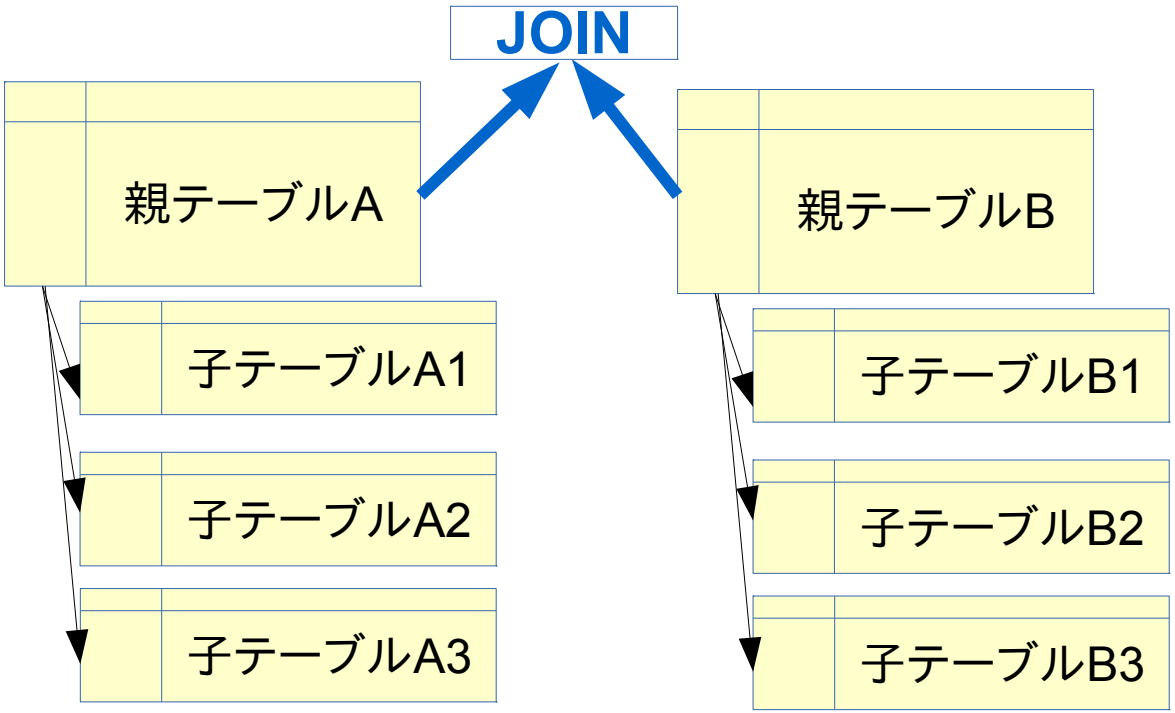
- 親テーブルへのインデックス付与
  - 子テーブルへの伝搬
- デフォルトパーティション
  - 振り分け条件に合うパーティションが「無かった」場合の振り分け先
- UPDATE でパーティションキーが変更されたときの、パーティション間自動移動
- 外部テーブルへの振り分け
- 事後のパーティションの分割およびマージ
  - 子テーブルを親テーブルから切り離し、新しいパーティションを作ってから、データの再投入、が必要

パーティショニング その後の可能性

- ハッシュパーティショニング
  - パーティショニングキーをハッシュ関数で変換
  - その値を元にデータを振り分ける方法

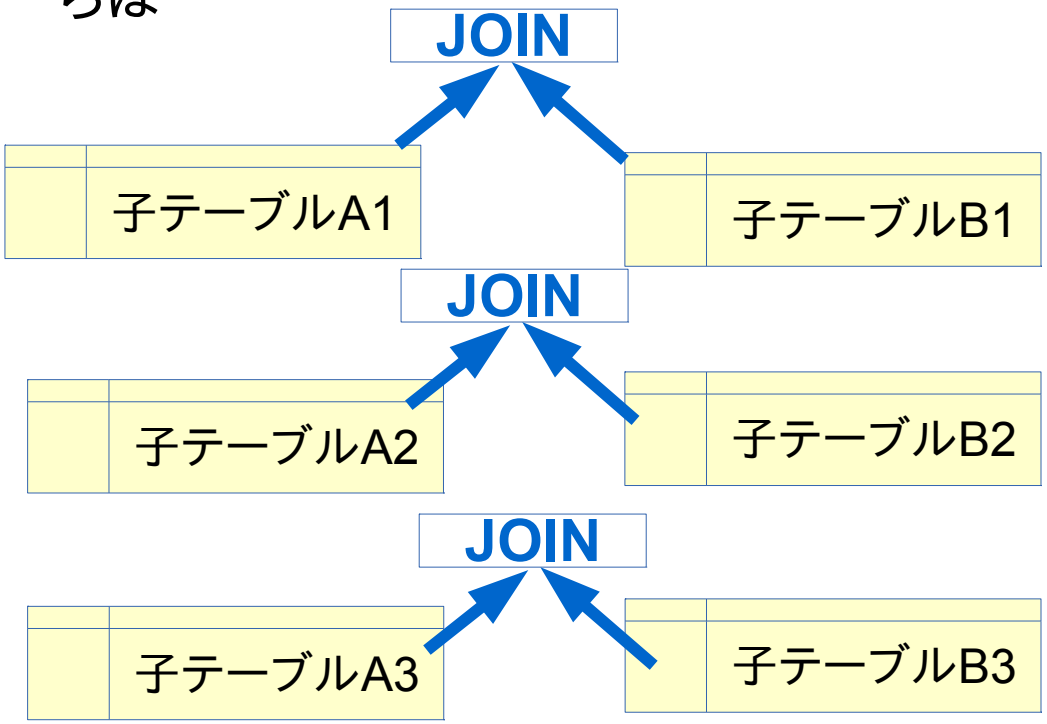
パーティショニング その後の可能性

- パーティション単位の処理
  - 巨大テーブルに対する処理を、パーティション単位で処理に分割
  - 例えば、パーティション単位の JOIN、集約、ソート、・・・・



パーティショニング その後の可能性

- パーティション単位のJOIN
  - JOIN の結合キーと、パーティションのキーが同じで、
  - 子テーブルAx と 子テーブル Bx のパーティションキーの範囲が同じならば



例えば、A1 と B2 はキーの範囲が違うので結合する必要なし

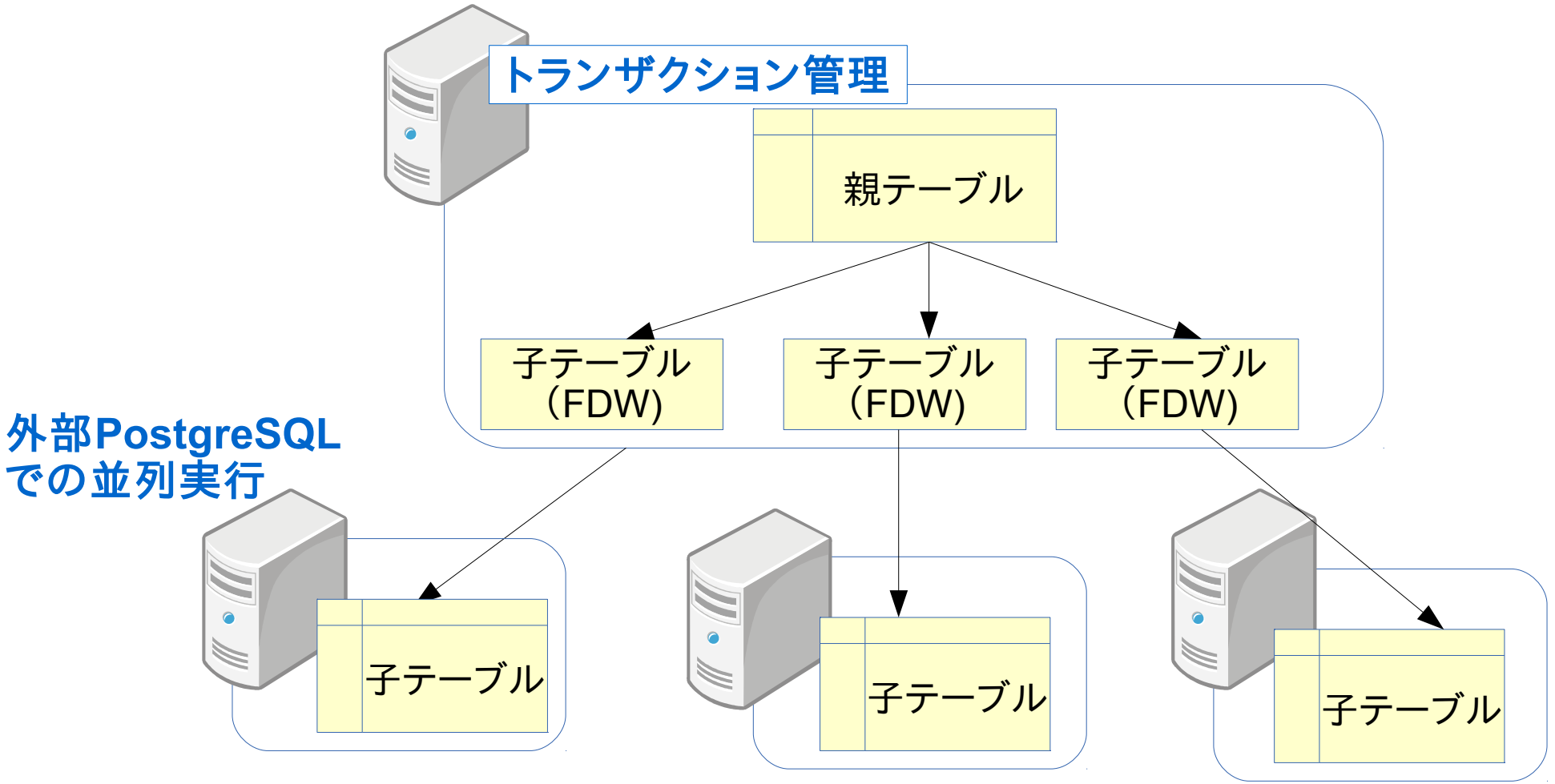
パーティショニング その後の可能性

- PostgreSQL でシャーディングを実現するには？
  - パーティショニングを考慮したクエリの最適化
    - パーティション単位のJOIN、集約、・・・
  - FDW(外部データラッパ)の非同期的実行
    - 外部の PostgreSQL で並列にクエリ処理
  - 分散トランザクション管理
    - 複数の PostgreSQL にまたがるトランザクションの管理

PGCon2017
Towards Built-in Sharding in Community PostgreSQL
Amit Langote, Etsuro Fujita, Kyotaro Horiguchi, Masahiko Sawada

パーティショニング その後の可能性

- PostgreSQL でシャーディングを実現するには？

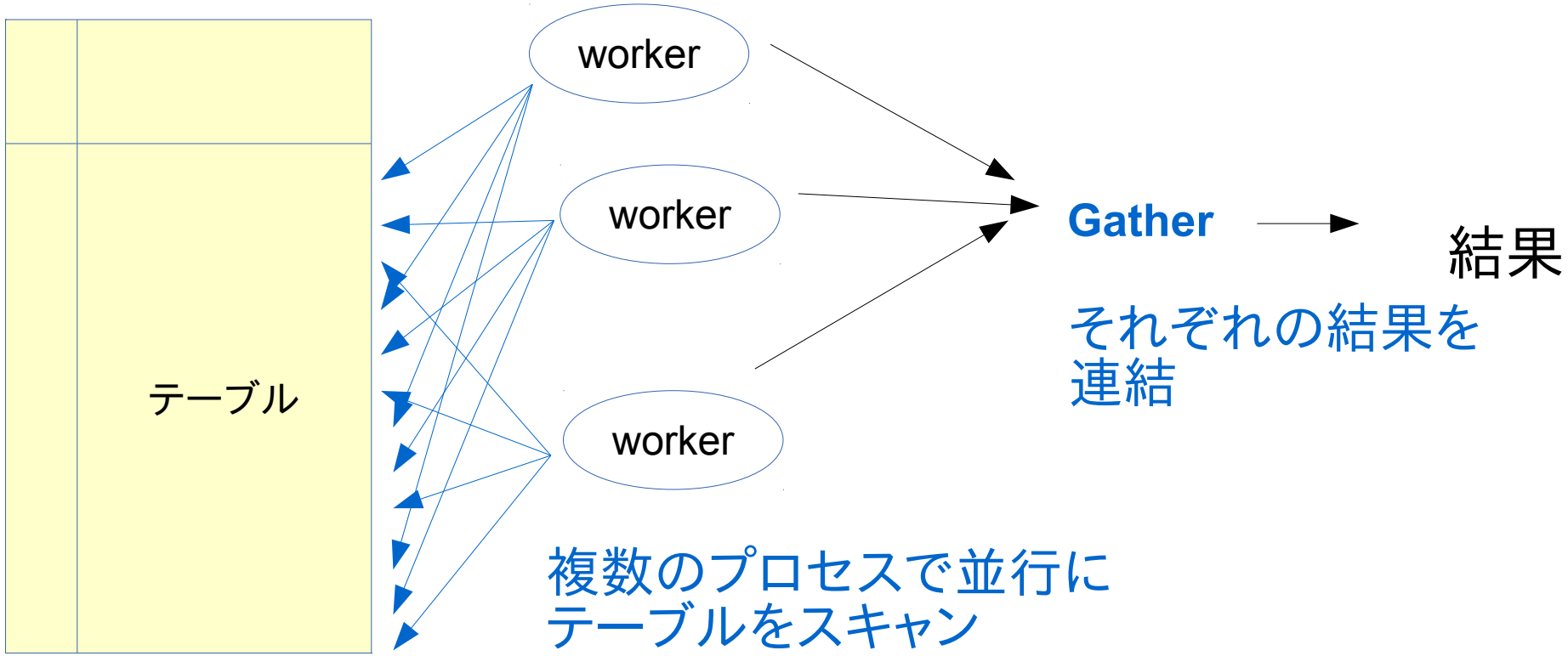


パラレルクエリ改善

PGCon2017
Parallel Query v2
The herd of elephants we unleashed claims new territory
Rafia Sabih, Robert Haas

パラレルクエリ (PostgreSQL 9.6～)

- 大きなテーブルに対するクエリを複数のCPUで実行することでパフォーマンス向上



PostgreSQL 9.6 までのパラレルクエリ

- Scan
  - Sequential scan
- JOIN
  - Nested Loop Join
  - Hash Join
- 集約

```
Gather
  Workers Planned: 4
  Workers Launched: 4
  ->  Parallel Seq Scan
```

```
Finalize HashAggregate
  ->  Gather
        Workers Planned: 4
        ->  Partial HashAggregate
              ->  Parallel Seq Scan
```

PostgreSQL 10 のパラレルクエリ

- Scan
  - Sequential scan
  - btree Index scan
  - Index only scan
  - Bitmap Heap Scan
    - Bitmap Index Scan はまだ
- Join
  - Nested Loop join
  - Hash join
  - Merge join

```
Finalize Aggregate
  ->  Gather
        Workers Planned: 4
        ->  Partial Aggregate
              ->  Parallel Index Scan
```

9.6 では未対応だったクエリもパラレルクエリの恩恵を得られるように

PostgreSQL 10 のパラレルクエリ

- Gather Merge
  - 複数のワーカが生成したソート済の行集合を、1つのソート済集合にまとめる
    - 大きなソートを小さいソートに分割し並列実行
    - Parallel btree Index Scan, Parallel Merge Join と相性がよい

```
Finalize GroupAggregate
  ->  Gather Merge
        Workers Planned: 4
        ->  Partial GroupAggregate
              ->  Sort
                    ->  Parallel Seq Scan on tenk1
```

- 非相関サブクエリのパラレル実行
- 手続き言語内でのパラレルクエリ

9.6 では未対応だったクエリもパラレルクエリの恩恵を得られるように

パラレルクエリの更なる改善

- Parallel Hash Join の改良
  - 現在
    - ワーカは外表の行を手分けして処理している
    - しかし、それぞれのワーカが内表の全行を処理!
      - CPUとメモリの無駄
  - Parallel Shared Hash
    - 1つのハッシュ表を複数プロセスで構築・共有

その他の新機能や話題

その他の PostgreSQL 10 新機能

- SCRAM認証
  - md5 より一層セキュアなパスワード認証方法
- Hash index
  - WAL ログ出力するようになった
    - 永続的、レプリケーション可能
  - 性能改善、サイズ抑制
- FDW で外部サーバに集約を push down
  - 可能な限り、外部サーバ側で集約計算させてから行を取得する
- Transition Table（遷移表）
  - AFTER TRIGGER の中で「クエリに影響を受けた全ての行」にアクセスできるようになった

複数カラムの統計情報

- プランナによる取得行数の推定
  - 今までは、それぞれのカラムで値の分布が独立であることを想定
  - しかし、カラム値に間に相関のある場合には、推定を誤って悪い計画になりがちだった

```
CREATE TABLE t (a INT, b INT);
INSERT INTO t
  SELECT i % 100, i % 100 FROM generate_series(1, 10000) s(i);
ANALYZE t;
```

```
EXPLAIN (ANALYZE, TIMING OFF) SELECT * FROM t WHERE a = 1 AND b = 1;
                                QUERY PLAN
-------------------------------------------------------------------------
 Seq Scan on t  (cost=0.00..195.00 rows=1 width=8) (actual rows=100 loops=1)
   Filter: ((a = 1) AND (b = 1))
   Rows Removed by Filter: 9900
```

複数カラムの統計情報

- ANALYZE で複数カラムに関する統計情報を取得可能になった
  - カラム間の相関係数（関数従属性）
  - 「カラムの組み合わせ」に現れる個別値の数
- 調べるカラムは「拡張統計（extended statistics)」を作成するときに指定
  - 以下は、関数従属性（dependencies）の統計情報を取得する例

```
CREATE STATISTICS stts (dependencies) ON a, b FROM t;
ANALYZE t;
EXPLAIN (ANALYZE, TIMING OFF) SELECT * FROM t WHERE a = 1 AND b = 1;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on t  (cost=0.00..195.00 rows=100 width=8) (actual rows=100 loops=1)
   Filter: ((a = 1) AND (b = 1))
   Rows Removed by Filter: 9900
```

プランナの未来？

- PGCon 2017 にて、プランナの改良に関する実験的な発表が2件
  - 機械学習を用いた適応的なクエリプランナ
    - カラム間に相関がある場合の行数推定
  - 実行時間だけを目的としない「多目的」最適化
    - 実行時間とメモリ使用量のトレードオフを考慮
    - 両者が丁度良く最適化される実行計画を生成

※ あくまで実験的提案であり PostgreSQL 10 の新機能ではありません

Write amplification 対策

- UPDATE 実行時の性能問題
- 開発コミュニティで議論中
  - テーブル更新時に、不要なインデックス更新を抑える仕組みが、2通り提案されている
    - Indirect indexes
      - そのテーブルの primary key の値を返すインデックス
    - WARM (Write Amplification Reduction Method)
      - 従来の仕組み（HOT: Heap Only Tuple) の改良版
  - Pluggable storage
    - 追記型MVCC以外にも、様々なストレージを使えるようにするしくみ

おわりに

- PostgreSQL 10
  - ロジカルレプリケーション
    - まだまだ登場したばかり
    - これからの運用・応用しだいで可能性は広がる
  - パーティショニング
    - 構文ができて構築が楽になった・・・だけではなく
    - 巨大データ処理の基盤の1つとして他の機能と連携
  - パラレルクエリ改善
    - この先も、並列化できる領域がどんどん拡張
- さらなる進化に向けた議論も継続中

参考資料

- PostgreSQL 10 beta 1 Documentation
  - https://www.postgresql.org/docs/10/static/index.html
- PGCon 2017
  - http://www.pgcon.org/2017/
- Commitfest
  - https://commitfest.postgresql.org/
- PostgreSQL 10 がやってくる!（ぬこ@横浜さん）
  - https://www.slideshare.net/toshiharada/jpug-studypostgre-sql10pub
- 篠田の虎の巻 7（篠田典良さん）
  - http://h50146.www5.hpe.com/products/software/oe/linux/mainstream/support/lcc/pdf/PostgreSQL_10_New_Features_ja_20170522-1.pdf
- Robert Haas blog
  - http://rhaas.blogspot.jp/
- 2ndQuadrant blog
  - https://blog.2ndquadrant.com
- PostgreSQL 技術情報 (SRA OSS, Inc. 日本支社）
  - https://www.sraoss.co.jp/technology/postgresql/

Thank you